Tight Bounds for Online Bandwidth Allocation on Two Links *

LEAH EPSTEIN
School of Computer Science, The Interdisciplinary Center, Herzliya, Israel.

Abstract

We study on-line bandwidth allocation on two parallel links. Motivated by issues of Quality of Service and fair resource allocation, the goal is to maximize the load of the least loaded link. We analyze several models of different and of identical links and give tight bounds on the competitive ratio in those models.

1 Introduction

We consider networks consisting of two parallel links (or paths) between a source and a destination. The links have capacities C_1 and C_2 ($C_2 \leq C_1$). Requests for bandwidth arrive on-line one by one. Each request j is specified by a size p_j. Each arriving request is to be assigned to one of the links. The assignment is final and is done before seeing the next request. The request causes load of p_j/C_i to link i. The load of each link is the sum of loads caused to it by all requests assigned to it. In order to improve the quality of service, we would like the loads on the links to be balanced. We measure the balance by maximizing the minimum load on any link.

A second application arises from seeing the problem as on-line resource allocation. The links require various resources to be able to keep going. Those resources may be physical as electricity or physical links, or logical as computational power. Each arriving "request" is a resource which needs to be assigned to one of the links. The goal is to assign the resources in a fair way so that no link will suffer from starvation and both links get a fair amount of resources.

The problem can be easily formulated as (non-preemptive) scheduling on two uniformly related machines. In this case the sizes of requests are their running times, the capacity becomes speed and the goal is to maximize the minimum completion time. This application is useful in systems where the complete system relies on keeping both machines alive. The system fails to work when one machine or both stop running. Here again the requests can be seen as resources that arrive

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

to the system. Being consistent with previous work on similar problems, we adopt the scheduling terminology for the rest of this paper. We use the terms speed, size (of a request), load (the running time of a request on a specific machine), and completion time (or load) of a machine. Let $q = C_1/C_2$ be the *speed ratio* of the machines. Without loss of generality we assume that the speeds are 1 and $1/q$. We refer to the general case ($q \geq 1$) as the model of related machines, and to the special case $q = 1$ as the case of identical machines. Using this terminology the problem is called "Machine Covering".

We measure the algorithms by the *competitive ratio* . The *competitive ratio* compares an on-line algorithm to an optimal off-line algorithm OPT that knows the input sequence in advance. We define the competitive ratio for maximization problems. For an algorithm $\mathcal{B}$, let $\mathcal{B}(\sigma)$ denote its value on an input sequence σ. When σ is clear from the context, we slightly abuse the notation and use $\mathcal{B}$ for the cost of the algorithm $\mathcal{B}$. The competitive ratio $\mathcal{R}(\mathcal{A})$ of an on-line algorithm $\mathcal{A}$ is the infimum $\mathcal{R}$ such that for every sequence σ,

$$\mathrm{OPT}(\sigma) \leq \mathcal{R} \cdot \mathcal{A}(\sigma)\ . \tag{1}$$

Among other models we study several restricted models where the cost of the optimal off-line algorithm is known in advance. In our main application this means that the required load on the links is known even before the arrival of any requests. In those cases we may assume (without loss of generality) that OPT = 1. Where OPT is known in advance our goal is to stretch our loads to get loads close to OPT on both machines. In this case we refer to the minimum load achieved also as the stretch factor.

Previous Results: The problem was previously studied for identical machines and for related machines. The off-line scheduling problem of maximizing the load of the least loaded machine among two machines is known to be NP-complete (for any speed ratio q).

Woeginger [5] showed that the greedy algorithm is m competitive for m identical machines. It is well known that no deterministic algorithm can achieve a better competitive ratio, this means that the best competitive ratio for 2 identical machines is 2. Deuermeyer, Friesen and Langston [4] studied the LPT-heuristic which orders the jobs by non increasing weights and assigns each job to the least loaded machine at the moment. It is shown in [4] that the competitive ratio of this heuristic is at most $\frac{4}{3}$. The tight ratio for this heuristic, $(4m-2)/(3m-1)$, is given by Csirik, Kellerer and Woeginger [3]. This model can be seen as a semi on-line problem on identical machines. The above papers give only upper bounds but it can be easily shown that for 2 and 3 machines LPT is the best possible semi-online algorithm. This gives the tight competitive ratio 1.2 for two machines.

In [1, 2] the model where OPT is known in advance is investigated. An algorithm of competitive ratio $2 - 1/m$ for m identical machines is given. It is shown that this algorithm has optimal competitive ratio for $2,3$ and 4 machines. Specifically the tight competitive ratio for 2 machines is 1.5.

For related machines [1] give several algorithms and lower bounds for m related machines. The competitive ratios are in this paper are given only as a function of m and not as a function of the speeds. It is shown that no algorithm can have a constant or a bounded competitive ratio already for two machines (the sequence which is used to show it gives a lower bound of q). If the jobs are arriving sorted in a non-increasing order of size, or OPT is known in advance, the tight competitive ratio is shown to be m. For the case that both holds, and algorithm of competitive ratio 2 is given.

Our Results: We show the tight competitive ratio for several models.

For the related machines case we investigate the competitive ratio as a function of q. we show that the exact competitive ratio is $q+1$ for two machines of speed ratio q. This competitive ratio grows linearly with q. The competitive ratio 2 which is known for 2 identical machines [5] is a special case of our result for the case $q=1$. To overcome the linearly growing competitive ratio we study the alternative model which restricts the input to sequences where OPT is known in advance. In this case we also get a tight analysis of the competitive ratio function. The function is split into 3 intervals (see Section 3), first decreasing and then increasing and its value is bounded between $\sqrt{2}$ and 2. The result for the special case $q=1$ (competitive ratio $3/2$) is given in [1, 2]. Note that the algorithm for the case of known OPT which is given in [1] (of competitive ratio 2) has optimal competitive ratio for $q \to \infty$.

Next we consider sequences where all jobs are bounded by OPT/k (for an integer k) on two identical machines. We get an exact competitive ratio of $2k/(2k-1)$. For the case where OPT is known in advance we get an exact competitive ratio of $(2k+1)/(2k)$. In those cases the improvement in the competitive ratio when OPT is known in advance becomes relatively small as k grows. The reason for this is that the knowledge that the jobs in the sequence are bounded by a fraction of OPT already gives a large amount of information. Note that the special case $k=1$ gives the previously known competitive ratios for 2 identical machines.

Notations: The input is a sequence of jobs. The jobs are denoted by $J_1, J_2, \ldots$. 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^ℓ ($\ell \geq 1$ is an index of the sequence to which the jobs belongs). Their sizes are p_i' and p_i^ℓ respectively. If a single job j is considered, we simply denote its size by p_j. The load of a machine is the sum of the sizes of jobs assigned to it divided by its speed. For identical machines we denote the machines by M_1 and M_2 and their loads by P_1 and P_2 (respectively) whereas for related machines we denote the machines by M_1 (fast) and M_q (slow) and the loads by P_1 (of the fast machine) and P_q (of the slow machine). Note that using the scheduling terminology, the time to run a job of size p on a machine of speed $1/q$ is pq. When it is known that OPT $= 1$, the total size of all jobs is at least $1+1/q$.

2 The Basic Model

In this section we analyze a non-restricted model. We show that the competitive ratio for this model grows with the speed ratio (it is $q+1$). This gives motivation to the study of more restricted models.

We use the simple algorithm **GREEDY** which assigns a job to the least loaded machine. Note that for related machines it is more common to use a different version of GREEDY (sometimes called post GREEDY) that checks the two loads after a possible assignment and not before the assignment. However for our model the simple GREEDY gives optimal competitive ratio (while post GREEDY can be arbitrarily bad. In order to be competitive, the first two jobs must be assigned to different machines, which is not always done by post GREEDY).

Lemma 1 *The competitive ratio of the on-line algorithm GREEDY for two machines of speed ratio q is at most $q+1$.*

Proof. We consider the structure of a general assignment built by GREEDY. Let j be the job of the maximum completion time and p_j be its size. Let x be the total size of jobs assigned to the same machine not including j. Let y the total size assigned to the other machine. We can increase the size of j into a large constant M without changing the value of the on-line algorithm. The optimal value might increase but cannot decrease. Now consider the optimal schedule. The best that can be done is assigning all jobs but j to the slow machine. This gives OPT $\leq q(x+y)$.
We consider two cases according to which machine has the job j in the on-line schedule.
If the fast machine has j then GREEDY $= qy$ and $qy \geq x$ (as j is assigned to the least loaded machine). We get OPT $\leq qy+q^2y = (q+1)$GREEDY.
If the slow machine has j then GREEDY $= y$ and $y \geq qx$. We get OPT $\leq qy+y = (q+1)$GREEDY. □

We show that the competitive ratio of GREEDY is the best possible:

Lemma 2 *The competitive ratio of any on-line algorithm for machine covering for two machines and speed ratio $q \geq 1$ is at least $q+1$.*

Proof. Consider an on-line algorithm ONL and the following sequence. The first job J_1 has size $p_1 = 1$. If it is assigned to M_1 then $p_2 = 1/q$ and otherwise $p'_2 = q$.

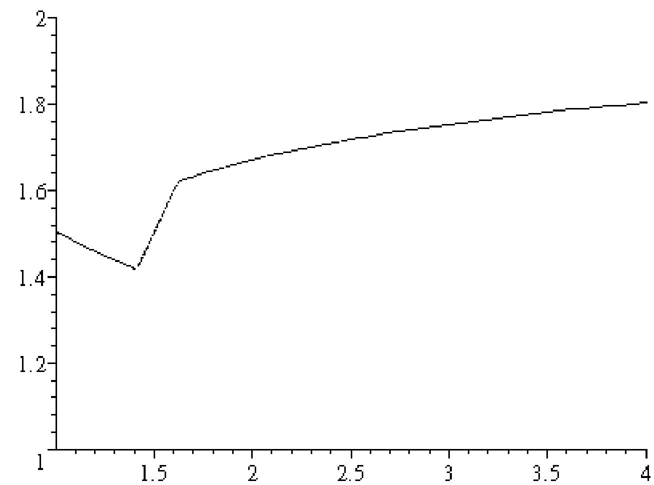

Figure 1: The competitive ratio as a function of q

In order to be competitive at this point each machine has exactly one job. One additional job arrives and has the size $p_3 = q+1$ in the first case and $p_3' = q^2+q$ in the second case. This gives $\mathrm{OPT} = q+1$ in the first case and $\mathrm{OPT} = q^2+q$ in the second. However the third job is assigned to only one of the machines and the other machine remains loaded with a single job and so $\mathrm{ONL} = 1$ in the first case and $\mathrm{ONL} = q$ in the second. This gives the competitive ratio $q+1$ in both cases. □

3 Known Optimal Value

In this section we assume that the optimal off-line value is known in advance. Recall that without loss of generality we let $\mathrm{OPT} = 1$. This reduction does not influence the competitive ratio . We show that the tight competitive ratio function $r(q)$ is as follows (see also figure 1):

$$r(q) = \begin{cases} \dfrac{q+2}{q+1} & \text{for } q \in \left[1, \sqrt{2} \approx 1.41421\right] \\ q & \text{for } q \in \left[\sqrt{2}, \phi = (\sqrt{5}+1)/2 \approx 1.61803\right] \\ \dfrac{2q+1}{q+1} & \text{for } q \geq \phi \end{cases}$$

3.1 Algorithms

We define two algorithms. An algorithm SFILL which gives preference to the slow machine (unless a relatively large job arrives), and an algorithm FFILL which gives preference to the fast machine.
We use SFILL for $q \geq \phi$. The algorithm uses a parameter $0 < \beta < 1$ which is fixed later.

> **Algorithm SFILL:**
> On arrival of a new job j of size p:
> 1. If $P_q \geq \beta/q$ and $P_1 \geq \beta$, assign the new job to an arbitrary machine.
> 2. If $P_q \geq \beta/q$ but $P_1 < \beta$, assign the new job to machine M_1.
> 3. If $P_q < \beta/q$ but $P_1 \geq \beta$, assign the new job to machine M_q.
> 4. Otherwise if $p \geq \beta$ assign j to M_1 and else to M_q.

We use the algorithm FFILL for the case where the slow machine is relatively fast ($q \leq \phi$). Here again $0 < \beta < 1$.

> **Algorithm FFILL:**
> On arrival of a new job j of size p:
> If the situation is one of the 3 first cases in algorithm SFILL,
> then follow those cases.
> Otherwise (case 4) if $p \geq \beta/q$ assign j to M_q and else to M_1.

Lemma 3 *The competitive ratio of the on-line algorithm SFILL for two machines of speed ratio $q \geq \phi$ using $\beta = (q+1)/(2q+1)$ is at most $(2q+1)/(q+1)$.*

Proof. It is clear that if the algorithm ever reaches the first case, we are done. We show that the algorithm cannot terminate before that.

If the algorithm terminates when it is supposed to use case 2 we are in the following situation. All jobs that arrived so far are smaller than β (otherwise such a job is assigned to M_1 and case 2 can never happen any more). Hence $P_q < \beta/q + \beta$ since once P_q reaches β/q or more, the other jobs are assigned to M_1. The total size of jobs is at least $1 + 1/q$ and so $M_1 \geq 1 + 1/q - \beta(1 + 1/q)$. Using $\beta = (q+1)/(2q+1)$ we get $P_1 > \beta$.

If the algorithm terminates when it is supposed to use case 3 we have that M_1 has a single job. A job could be assigned to M_1 while being in case 4 only if its size is at least β, and this moves the algorithm to case 3. Then no other jobs may be assigned to M_1 until moving to case 1. The job on M_1 is assigned to one of the machines in the optimal off-line schedule. The other machine has jobs of total size of at least $1/q$ and so the algorithm cannot terminate in this situation if $\beta/q \leq 1/q$ which holds for any $\beta \leq 1$.

If the algorithm terminates when it is supposed to use case 4 we get that the total size of jobs is less than $\beta(1 + 1/q)$ which can not hold for $\beta \leq 1$. □

Lemma 4 *The competitive ratio of the on-line algorithm FFILL for two machines* *of speed ratio* $q \le \phi$ *using* $\beta = (q+1)/(q+2)$ *is at most* $\max\{q, (q+2)/(q+1)\}$.

Proof. Again it is clear that if the algorithm ever reaches the first case, we are done. We show that if the algorithm terminates before that, its stretch factor is at least $1/q$.

If the algorithm terminates when it is supposed to use case 2 we are in the following situation. All jobs that arrived so far are smaller than β/q (otherwise such a job is assigned to M_q). Hence $P_1 < \beta + \beta/q$ since once P_1 reaches β or more, the other jobs are assigned to M_q. The total size of jobs is at least $1 + 1/q$ and so $M_q \ge 1 + 1/q - \beta(1 + 1/q)$. Using $\beta = (q+1)/(q+2)$ we get $P_q > \beta/q$.

If the algorithm terminates when it is supposed to use case 3 we have that M_q has a single job. A job could be assigned to M_q while being in case 4 only if its size is at least β/q and this moves the algorithm to case 3. Then no other jobs may be assigned to M_1 until moving to case 1. The job on M_1 is assigned to one of the machines in the optimal off-line schedule. The other machine has jobs of total size of at least $1/q$ and so the algorithm can terminate in this situation but $P_1 \ge 1/q$.

If the algorithm terminates when it is supposed to use case 4 we get that the total size of jobs is less than $\beta(1 + 1/q)$ which can not hold for $\beta \le 1$. □

3.2 Lower Bounds

We show that both algorithms SFILL and FFILL have optimal competitive ratios in the intervals where they are defined. We consider each one of the three intervals separately.

Lemma 5 *The competitive ratio of any on-line algorithms for machine covering and known optimal value for two machines of speed ratio* $q \ge \phi$ *is at least* $(2q+1)/(q+1)$.

Proof. Let $\alpha = (q+1)/(2q+1)$. The first job J_1 has size $p_1 = \alpha/q$. We consider two cases.

Case 1: J_1 is assigned to M_1. The sequence continues with two jobs J_2' and J_3' of sizes $p_2' = 1$ and $p_3' = (1-\alpha)/q = 1/(2q+1)$. As $p_1 + p_3' = 1/q$, OPT can assign J_2' to M_1 and the other jobs to M_q and to get OPT $= 1$. In order to be competitive, the on-line algorithm must assign at least one job to M_q. If J_2' is assigned to M_q then $P_1 \le p_1 + p_3' = 1/q \le (q+1)/(2q+1)$ (holds for $q \ge \phi$). Otherwise $P_q = p_3'$ and so the stretch factor is $q/(2q+1) < (q+1)/(2q+1)$. In both cases the stretch factor is too low.

Case 2: J_1 is assigned to M_q. We continue with a job J_2 of size $p_2 = \alpha$.

Sub-case 2.1: J_2 is also assigned to M_q. We have 2 additional jobs of sizes $p_3 = 1/q$ and $p_4 = 1 - \alpha - \alpha/q = (q^2 - q - 1)/(2q^2 + q))$. Note that for $q \ge \phi$, $p_4 \ge 0$

holds. As $p_1+p_2+p_4=1$, OPT $=1$. However $P_1 \le p_3+p_4=(q+1)/(2q+1)$, and hence the stretch factor is at most α.

Sub-case 2.2: J_2 is assigned to M_1. We have one additional job of size $p_3''=1-\alpha/q$. As $p_2=\alpha \ge 1/q$ and $p_1+p_3''=1$, again OPT $=1$. (Any other assignment of the same jobs yields a lower value of OPT.) However J_3'' can be assigned to only one of the two machines and so either $P_q=\alpha/q$ or $P_1=\alpha$. In both case the stretch factor is at most α. □

Lemma 6 *The competitive ratio of any on-line algorithms for machine covering and known optimal value for two machines of speed ratio* $\sqrt{2} \le q \le \phi$ *is at least* q.

Proof. The first job J_1 has size $p_1=1/q^2$. We consider two cases. If J_1 is assigned to M_1 the sequence continues by two jobs J_2' and J_3' of sizes $p_2'=1$ and $p_3'=1/q-1/q^2$. As $p_1+p_3'=1/q$, OPT can assign J_2' to M_1 and the other jobs to M_q and to get OPT $=1$. In order to be competitive, the on-line algorithm must assign at least one job to M_q. If J_2' is assigned to M_1 then $P_q=p_3'=(q-1)/q^2 \le 1/q^2$ (for $q \le 2$). Otherwise (J_2' is assigned to M_q) then $P_1 \le p_1+p_3'=1/q$ and so the stretch factor is at most $1/q$ in both cases.

Otherwise (J_1 is assigned to M_q) we continue with a two jobs J_2 and J_3 of sizes $p_2=1/q$ and $p_3=1-1/q^2$. Again the optimal off-line schedule is easily obtained since $p_1+p_3=1$ and $p_2=1/q$. If both J_2 and J_3 are assigned to M_1, $P_q=1/q^2$ which gives stretch factor $1/q$. Otherwise $P_1 \le \max\{1/q, 1-1/q^2\}$. For $q \le \phi$ we have $1/q \ge (q^2-1)/q^2$ and so the stretch factor is at most $1/q$. □

Lemma 7 *The competitive ratio of any on-line algorithms for machine covering and known optimal value for two machines of speed ratio* $1 \le q \le \sqrt{2}$ *is at least* $(q+2)/(q+1)$.

Proof. Let $\alpha=(q+1)/(q^2+2q)$. The first job J_1 has size $p_1=1/(q^2+2q)$.

We consider two cases.

Case 1: J_1 is assigned to M_q. The sequence continues with a job J_2 of size $p_2=1/(q+2)$.

Sub-case 1.1: J_2 is assigned to the other machine (M_1). We conclude with two jobs of sizes $p_3^1=1/q-\alpha=1/(q^2+2q)$ and $p_4^1=1$. If J_4^1 is assigned to M_1 we get $P_q \le p_1+p_3^1=2/(q^2+2q)=2\alpha/(q+1) \le \alpha$ which holds for any $q \ge 1$. If J_4^1 is assigned to M_q we get $P_1 \le p_2+p_3^1=\alpha \le q\alpha$. We can see that OPT $=1$ as $p_1+p_2+p_3^1=1/q$ and $P_4^1=1$.

Sub-case 1.2: Both first jobs are assigned to M_q. We continue by another job J_3 of size $p_3=\alpha$. If the new job is assigned to M_1 we finish by one job of size $p_4^2=(q+1)/(q+2)$. Otherwise we finish by two jobs of sizes $p_4^3=1-\alpha=$

$(q^2+q-1)/(q^2+2q)$ and $p_5^3 = 1/q-\alpha = 1/(q^2+2q)$. In the second case, as $p_1+p_2+p_4^3 = 1$ and $p_3+p_5^3 = 1/q$, hence OPT can assign the jobs so that OPT $=$ 1. In the first case $p_1+p_3 = 1/q$ and $p_2+p_4^2 = 1$. In the first case it is clear that one machine will be left with total size α and so its stretch factor is at most $q\alpha = (q+1)/(q+2)$. In the second case $P_1 \le p_4^3+p_5^3 = (q+1)/(q+2)$.

Case 2: J_1 is assigned to M_1. We continue with a job J_2' of size $p_2' = q\alpha - 1/(q^2+2q) = (q^2+q-1)/(q^2+2q)$.

Sub-case 2.1: J_2' is assigned to M_q. We give two last jobs of sizes $p_3^4 = 1$ and $p_4^4 = 1/q-q\alpha = (2-q^2)/(q^2+2q) \ge 0$. Again OPT $= 1$ since $p_1+p_2'+p_4^4 = 1/q$ and $p_3^4 = 1$. If J_3^4 is assigned to M_q we get $P_1 \le 1/(q^2+2q)+1/q-q\alpha = (3-q^2)/(q^2+2q) \le (q+1)/(q+2)$ for $q \ge 1$. Otherwise $P_q \le q\alpha - 1/(q^2+2q)+1/q-q\alpha = \alpha$.

Sub-case 2.2: Both first jobs were assigned to M_1. We continue with a job J_3 of size $p_3 = \alpha$. If the new job is assigned to M_q we finish by one job of size $p_4^5 = \alpha$. Otherwise we finish by two jobs of sizes $p_4^6 = 1-q\alpha = 1/(q+2)$ and $p_5^6 = 1/q-\alpha = 1/(q^2+2q)$. To show that in both cases OPT $= 1$ we can see that $p_2'+p_3 = 1$ and $p_1+p_4^5 = 1/q$, and that $p_1+p_2'+p_4^6 = 1$ and $p_3+p_5^6 = 1/q$. In the first case it is clear that we have either $P_q = \alpha$ or $P_1 = q\alpha$, and the stretch factor is $(q+1)/(q+2)$. In the second case $P_q \le 1/(q+2)+1/(q^2+2q) = \alpha$ and the stretch factor is the same. □

4 Bounded Items and Identical Machines

In this section we consider the case where there is a bound on the size of jobs in terms of OPT. Let k be a positive integer. We say that the sequence is k bounded if all the jobs have size at most OPT$/k$. In the previous sections we can say that jobs were 1 bounded since there is no difference between a job of size OPT and a larger job. If OPT $= 1$ we can say that in a k bounded sequence, all items are of size at most $1/k$. If OPT is not known in advance, we can still use this definition. When a job of size 1 arrives, and the sequence is k bounded it means that we can already know that OPT $\ge k$, even if it does not hold for the subsequence that arrived so far. We consider two identical machines. In this case the bound is uniform for both machines. For uniformly related machines and large values of q there is big difference between the size of jobs assigned by OPT on the two machines, hence the analysis of k bounded sequences is natural mainly for identical machines.

4.1 The General Model

We show the simple analysis of GREEDY in this case.

Lemma 8 *The competitive ratio of GREEDY for k bounded sequences and two* *identical machines is* $\frac{2k}{2k-1}$.

Proof. Let P be the total size of jobs. And let $p \leq \mathrm{OPT}/k$ be the size of the last job. The total size on the less loaded machine is at least $(P-p)/2$ We use $\mathrm{OPT} \leq P/2$ to get $\mathrm{ONL} \geq P/2 - p/2 \geq \mathrm{OPT} - \mathrm{OPT}/(2k) = (2k-1)/(2k)\mathrm{OPT}$. □

Note that if we consider two related machines of speed ratio q, a very similar analysis gives the competitive ratio $(kq+k)/(kq+k-q)$. For $k=1$ we already saw that this ratio tend to ∞ linearly. However for $k>1$ this ratio is bounded by $k/(k-1)$. This gives constant competitive ratio for any $k>1$ (even for non-integer k).

We show that GREEDY has optimal competitive ratio for k bounded sequences.

Lemma 9 *The competitive ratio of any on-line algorithm for k bounded sequences and two identical machines is at least* $\frac{2k}{2k-1}$.

Proof. The sequence starts by $2k$ jobs of size 1. At this point $\mathrm{OPT}=k$, so if they are not assigned evenly we have one machine with load at most $k-1$ which give competitive ratio $k/(k-1) < (2k)/(2k-1)$. Otherwise there are additional $2k-1$ jobs of size $2k$. This gives $\mathrm{OPT} = 2k^2$. However there is one on-line machine with load of at most $(k-1)(2k)+k = 2k^2-k$ which gives the desired competitive ratio. □

4.2 Known Optimal Value

We adapt the algorithm we used for 1 bounded sequences to k bounded sequences in the following way. The algorithm uses two parameters $0 < \beta < 1$ and integer $k \geq 1$. It fills one machine with small jobs and the other with larger jobs, until one of them reaches β.

Algorithm kFILL:
On arrival of a new job j of size p:
1. If $P_2 \geq \beta$ and $P_1 \geq \beta$, assign the new job to an arbitrary machine.
2. If $P_2 \geq \beta$ but $P_1 < \beta$, assign the new job to machine M_1.
3. If $P_2 < \beta$ but $P_1 \geq \beta$, assign the new job to machine M_2.
4. Otherwise if $p \geq \beta/k$ assign j to M_1 and else to M_2.

Lemma 10 *The competitive ratio of kFILL for k bounded sequences using* $\beta = \frac{2k}{2k+1}$*, two identical machines and* $OPT = 1$ *is* $\frac{2k+1}{2k}$.

Proof. It is clear that if the algorithm ever reaches the first case, we are done. We show that the algorithm cannot terminate before that.

If the algorithm terminates when it is supposed to use case 2 we are in the following situation. All jobs that arrived so far and were assigned to M_2 are smaller than β/k. Jobs of size β/k or larger were assigned to M_1 (though M_1 could receive other jobs as well being in case 2). P_2 cannot exceed $\beta+\beta/k$; when the machine reaches at least β, the algorithm moves to case 2 and no more jobs are assigned to it until it moves to case 1. The last job has size less than β/k. The total size of jobs is at least 2 and so we get $2 \geq P_1+P_2 < \beta+\beta/k+\beta$. Using $\beta = (2k)/(2k+1)$ we get $2 < \beta(2k+1)/k = 2$ which is a contradiction.

If the algorithm terminates when it is supposed to use case 3 we have that M_1 has several jobs. A job could be assigned to M_1 while being in case 4 only if its size is at least β/k. When it gets k such jobs (or less) the algorithm moves to case 3. Then no other jobs may be assigned to M_1 until moving to case 1. In this way we get a bound of 1 on the total size assigned to M_1. The total size of k jobs does not exceed 1 but is at least β. Hence moving to case 3 is done when $\beta \leq P_1 \leq 1$. When the algorithm stops we get that $P_2 \geq 1$. Hence both machines reach at least β.

If the algorithm terminates when it is supposed to use case 4 we get that the total size of jobs is less than 2β which can not hold for $\beta \leq 1$. □

We show that kFILL has optimal competitive ratio for k bounded sequences.

Lemma 11 *The competitive ratio of any on-line algorithm for k bounded sequences and two identical machines is at least* $\frac{2k+1}{2k}$.

Proof. We start with two jobs of size $a_1 = 1/(2k+1)$ each. If they are assigned to the same machine we continue with $2k$ jobs of size $a_2 = 2/(2k+1)$ and otherwise with $2k-1$ jobs of size $a_3 = 1/k$ and a single job of size $a_4 = 1/(2k^2+k)$. In the first case OPT has one a_1 and k jobs of size a_2 on each machine. In the second case OPT has both a_1 jobs, the a_4 job and $k-1$ of the a_3 jobs on one machine, and k of the a_3 jobs on the other.

Consider now the on-line algorithm in the two cases. In the first case since $a_2 = 2a_1$ we basically have $2k+1$ jobs of type a_2. No matter how such jobs are assigned, one machine has load of at most $ka_2 = 2k/(2k+1)$. In the second case the most even schedule can be reached by assigning the same number of jobs to one machine. The less loaded machine will have load of $a_1+(k-1)a_3+a_4 = 1/(2k+1)+(k-1)/k+1/(2k^2+k) = (2k)/(2k+1)$. □

5 Conclusions

We saw that for all somewhat restricted models constant competitive ratios can be achieved. In this paper we did not investigate the semi on-line problem (non-increasing sizes) which is also of interest.

References

[1] Y. Azar and L. Epstein. On-line machine covering. In *5th Annual European Symposium on Algorithms (ESA97)*, pages 23–36, 1997.

[2] Y. Azar and L. Epstein. On-line machine covering. *Journal of Scheduling*, 1:2:67–77, 1998.

[3] J. Csirik, H. Kellerer, and G. Woeginger. The exact LPT-bound for maximizing the minimum completion time. *Operations Research Letters*, 11:281–287, 1992.

[4] B. Deuermeyer, D. Friesen, and M. Langston. Scheduling to maximize the minimum processor finish time in a multiprocessor system. *SIAM J. Discrete Methods*, 3:190–196, 1982.

[5] G. Woeginger. A polynomial time approximation scheme for maximizing the minimum machine completion time. *Operations Research Letters*, 20:149–154, 1997.

Leah Epstein is with the School of Computer Science, The Interdisciplinary Center, Herzliya, Israel. E-mail: lea@idc.ac.il