The hierarchical model for load balancing on two machines *

Orion Chassid † Leah Epstein‡

Abstract

Following previous work, we consider the hierarchical load balancing model on two machines of possibly different speeds. We first focus on maximizing the minimum machine load and show that no competitive algorithm exists for this problem. We overcome this barrier in two ways, both related to previously known models. The first one is fractional assignment, where each job can be arbitrarily split between the machines. The second one is a semi-online model where the sum of jobs is known in advance. We design algorithms of best possible competitive ratios for both these cases. Furthermore, we show that the combination of the two models leads to the existence of an optimal algorithm (i.e., an algorithm of competitive ratio 1). This algorithm is clearly optimal for the makespan minimization problem as well. For the latter problem, we consider the fractional assignment model and design an algorithm of best possible competitive ratio for it.

1 Introduction

We study load balancing on two machines for cases where the two processors or machines do not have the same capabilities.

We consider online algorithms. For an algorithm $\mathcal{A}$, we denote its cost by $\mathcal{A}$ as well. The cost of an optimal offline algorithm that knows the complete sequence of jobs is denoted by OPT. In this paper we measure the performance of algorithms using the (absolute) competitive ratio. For minimization problems, the competitive ratio of $\mathcal{A}$ is the infimum $\mathcal{R} \geq 1$ such that for any input, $\mathcal{A} \leq \mathcal{R} \cdot \text{OPT}$. For maximization problems, the competitive ratio of $\mathcal{A}$ is the infimum $\mathcal{R} \geq 1$ such that for any input, $\text{OPT} \leq \mathcal{R} \cdot \mathcal{A}$.

If the competitive ratio of an online algorithm is at most $\mathcal{C}$ we say that it is $\mathcal{C}$-competitive. If no $\mathcal{R}$ satisfying the inequality exists, we say that the competitive ratio is unbounded or ∞.

The most general non-preemptive online scheduling model assumes m machines $1, \ldots, m$ and n jobs, arriving one by one. The information associated with a job j is a vector of p_j of length m, where p_j^i is the processing time or size of job j if it is completely assigned to machine i. Each job is to be assigned to a machine before the arrival of the next job. The load of a machine i is the

*This work was submitted as the M.Sc. thesis of the first author.

†Department of Mathematics, University of Haifa, 31905 Haifa, Israel. orion.chassid@gmail.com.

‡Department of Mathematics, University of Haifa, 31905 Haifa, Israel. lea@math.haifa.ac.il.

sum of the processing times on machine i of jobs assigned to this machine. The typical goal is to minimize the maximum load of any machine but other goals such as maximizing the minimum load of any machine [5, 3] and minimizing the ℓ_p norm of the load vector [2] were studied as well.

This model is known as *unrelated machines* [1]. Many simplified models were defined, both in order to allow the design of algorithms with good performance (which is often difficult, or even impossible, for unrelated machines), and to make the studied model more similar to reality. In the sequel we describe a few models which are relevant to our study.

Uniformly related machines [1, 7] are machines with speeds associated with them, thus machine i has speed s_i and the information that a job j needs to provide upon its arrival is just its size, or processing time on a unit speed machine, which is denoted by p_j. Then we have $p_j^i = p_j/s_i$. If all speeds are equal, we get *identical machines* [11].

Restricted assignment [4] is a model where each job may be run only on a subset of the machines. A job j is associated with a processing time p_j which is the time to run it on any of its permitted machines M_j. Thus if $i \in M_j$ we have $p_j^i = p_j$ and otherwise $p_j^i = \infty$. The hierarchical model represents a situation where there is a clear order between the strength of machines, in terms of the jobs they are capable of performing. Thus, the set M_j is a prefix of the machines for any j.

In this paper we consider the *restricted related hierarchical model*, where machine i has speed s_i, job j has a processing time of p_j on a unit speed machine, and may run on machines $1, \ldots, g_j$. Therefore, $p_j^i = \frac{p_j}{s_i}$ if $i \leq g_j$ and otherwise $p_j^i = \infty$.

We study the case of two machines. In this case, it is reasonable to assume that the machine that is capable of running any job is faster, since this is a stronger machine. However, the opposite case can occur in real life as well, when the machine that cannot run any job, is more specialized, and works faster when it is running the jobs that it is capable of running.

Since we consider the case of two machines, we have $g_j = 1$, if the job may run only on the first machine, and $g_j = 2$ if it can run on both. We denote by D the sum of jobs j such that $g_j = 1$. Throughout the paper, we denote the speed of the first machine by q and assume that the second machine has unit speed. Since the machines are different, we need to consider both cases $q \geq 1$ and $q < 1$, together or separately. The jobs are denoted by $j_1, j_2, \ldots$, in the order of arrival and have respective processing times $p_1, p_2, \ldots$. Their hierarchies are denoted by $g_1, g_2, \ldots$. We denote the final loads of the two machines by L_1 and L_2, and the total processing time assigned to the machines by P_1 and P_2. We have $P_2 = L_2$ and $P_1 = q \cdot L_1$. We denote the sum of all processing times by Σ, and if the value of Σ is known in advance, we assume $\Sigma = 1$ (which can be achieved by scaling).

Previous results. The hierarchical model for general m was studied in [6] (see also [8]). They designed a non-preemptive $e + 1 \approx 3.718$-competitive algorithm.

Park, Chang and Lee [13] and independently Jiang, He and Tang [12] studied the problem on two identical speed hierarchical machines. They both designed $\frac{5}{3}$-competitive algorithms and showed that this ratio is best possible. In [13], the semi-online variant where the sum of job processing times is known in advance was studied as well. A $\frac{3}{2}$-competitive algorithm was designed. The

authors further showed that this is best possible.

The paper [12] considered a restricted type of a preemptive model in which idle time is not allowed. They designed a $\frac{3}{2}$-competitive algorithm and showed this is best possible. Additional work on preemptive models can be found in [9].

Another model where each job can be arbitrarily split between the machines, and parts of the same job can run on different machines, possibly in parallel, was studied in the restricted assignment model [4]. It was shown there that this option does not change the order of growth of the best possible competitive ratio, which is $\Theta(\log n)$. We refer to this model as fractional assignment and to the model where each job is assigned completely to one machines as integral assignment.

Our results. We provide a complete solution for several problems, for which we consider a setting of two hierarchical machines, and all possible speed combinations.

The max-min model is the one where the goal is to maximize the load of the least loaded machine. The min-max model is the one where the goal is to minimize the load of the most loaded machine.

In this paper we first focus on the max-min model. We show that no competitive algorithm exists for this problem. To overcome this barrier, we study two more specific previously known models. The first one is fractional assignment, defined above. The second one is a semi-online model where the sum of jobs is known in advance. We design algorithms of best possible competitive ratios for both these cases. Furthermore, we show that the combination of the two models leads to the existence of an optimal algorithm (i.e., an algorithm of competitive ratio 1). Note that the overall competitive ratio for fractional assignment (i.e., the supremum competitive ratio for any speed) turns out to be 2, whereas the overall competitive ratio for our semi-online model is unbounded. The values of these ratios for $q = 1$, i.e., for identical speed machines are $\frac{3}{2}$ and 2, respectively.

The optimal algorithm for the combined model is clearly optimal for the makespan minimization problem as well. For this latter problem, we consider the fractional assignment model and design an algorithm of best possible competitive ratio for it. The overall competitive ratio of this algorithm is $\frac{4}{3}$, and this value is achieved for $q = 1$. Note that this is the only case we find in this paper, where the competitive ratio is the same for two machines of speed ratio q, no matter which one of them is the stronger machine in terms of hierarchy.

We summarize our results in table 1. All bounds are tight, thus there is a single entry for each problem.

Note that for two uniformly related machines of speed ratio q, the best competitive ratio for the max-min model is $q + 1$, which is achieved by a greedy algorithm [10].

2 Standard assignment in the max-min model

In this section we show that no algorithm with bounded competitive ratio exists in this model, and thus more specific models need to be studied. The next theorem is valid for any $q \in (0, \infty)$.

Scheduling Problem	$q < 1$	$q \geq 1$
Integral online assignment, max-min	∞	∞
Fractional online assignment, max-min	$\frac{2q+1}{q+1}$	$\frac{2q+1}{q+1}$
Integral semi-online assignment, max-min	$1+\frac{1}{q}$	$q+\frac{1}{q}$
Fractional semi-online assignment (both max-min and min-max)	1	1
Fractional online assignment, min-max	$\frac{(q+1)^2}{q^2+q+1}$	$\frac{(q+1)^2}{q^2+q+1}$

Table 1: Table of results

Theorem 1 *Any load balancing algorithm for max-min optimization has an unbounded competitive ratio.*

Proof Consider an algorithm $\mathcal{A}$ and the following sequence of jobs. The first job has the properties $p_1 = 1, g_1 = 2$ and we must assign it to the second machine, else a second job with $p_2 = q$, $g_2 = 1$ arrives and after it is assigned (to machine 1) we get $\text{OPT} = 1$ and $\mathcal{A} = 0$. This implies $R = \infty$.

Otherwise, let M be a sufficiently large number. The second job has $p_2 = qM, g_2 = 2$ and we must assign it to machine 1 else we get $\mathcal{A} = 0$ and $\text{OPT} \geq 1$ and $R = \infty$ again.

Finally, a third job with $p_3 = q(qM + 1)$ and $g_3 = 1$ arrives. We must assign it to machine 1 and get $\mathcal{A} = 1$, since this is the load on the second machine. However, $\text{OPT} = qM + 1$ which implies that $R \to \infty$ as M grows, for any fixed q.

□

3 Fractional assignment in the max-min model

In this section we design an algorithm for fractional assignment. The next algorithm is defined for any $q \in (0, \infty)$.

Algorithm 1

Upon arrival of a new job j,

1. If $g_j = 1$, assign j completely to the first machine.
2. If $g_j = 2$, split it into two parts in the ratio $q : q + 1$. The first machine receives a part of size $\frac{qp_j}{2q+1}$ of the job and the second machine receives a part of size $\frac{(q+1)p_j}{2q+1}$ of the job.

Theorem 2 *Algorithm 1 has a competitive ratio of at most $\frac{2q+1}{q+1}$, which is best possible.*

Proof Denote by ALG the cost of Algorithm 1. If no job has $g_j = 2$ then $\text{OPT} = 0$ and $\text{ALG} = 0$, thus the competitive ratio is 1. Otherwise we have $L_2 = P_2 = \frac{q+1}{2q+1}(\Sigma - D)$ and $P_1 = D + \frac{q}{2q+1}(\Sigma - D) = \frac{q+1}{2q+1}D + \frac{q}{2q+1}\Sigma$ and so $L_1 = \frac{q+1}{q(2q+1)}D + \frac{1}{2q+1}\Sigma$.

We use the following bounds on OPT. Due to the sum of processing times, $\text{OPT} \le \frac{\Sigma}{q+1}$. Since the second machine can receive jobs of a total processing time of at most $\Sigma - D$, $\text{OPT} \le \Sigma - D$.

If $D \le \frac{q\Sigma}{q+1}$ and the minimum load is on machine 1, using $\text{OPT} \le \frac{\Sigma}{q+1}$, $\frac{\text{OPT}}{\text{ALG}} \le \frac{\frac{\Sigma}{q+1}}{\frac{q+1}{q(2q+1)}D + \frac{1}{2q+1}\Sigma} \le \frac{2q+1}{q+1}$, since $D \ge 0$.

If $D \le \frac{q\Sigma}{q+1}$ and the minimum load is on machine 2, using $\text{OPT} \le \frac{\Sigma}{q+1}$, $\frac{\text{OPT}}{\text{ALG}} \le \frac{\frac{\Sigma}{q+1}}{\frac{q+1}{2q+1}(\Sigma - D)} \le \frac{\frac{\Sigma}{q+1}}{\frac{q+1}{2q+1}(\Sigma - \frac{q\Sigma}{q+1})} = \frac{\frac{\Sigma}{q+1}}{\frac{(q+1)}{2q+1}\frac{\Sigma}{q+1}} \le \frac{2q+1}{q+1}$.

If $D \in (\frac{q\Sigma}{q+1}, \Sigma]$, the minimum load is on machine 2 (since $(\Sigma - D)\frac{q+1}{2q+1} < \Sigma - D < \frac{\Sigma}{q+1}$), $\text{ALG} = \frac{q+1}{2q+1}(\Sigma - D)$, using $\text{OPT} \le \Sigma - D$, $\frac{\text{OPT}}{\text{ALG}} \le \frac{\Sigma - D}{\frac{q+1}{2q+1}(\Sigma - D)} = \frac{2q+1}{q+1}$.

To prove a lower bound, we consider an algorithm $\mathcal{A}$ and the following sequence of jobs. The first job has $g_1 = 2$, $p_1 = 1$. Let δ be the part of this job assigned by $\mathcal{A}$ to the second machine. At this time the best way to split the job is into parts of size $\frac{q}{q+1}$ and $\frac{1}{q+1}$ and thus $\text{OPT} = \frac{1}{q+1}$.

If $\delta \ge \frac{q+1}{2q+1}$, then the loads of the machines are $L_2 = \delta$ and $L_1 \le \frac{1}{2q+1} < \delta$. Thus the minimum load is on machine 1 which leads to a competitive ratio of at least $\frac{2q+1}{q+1}$. Otherwise, the sequence is continued by an additional job with $g_2 = 1$, $p_2 = q$. After this job we have $\text{OPT} = 1$. Clearly, this job must run on the first machine, and the minimum load, which is $\delta < 1$, is on machine 2, the competitive ratio is now $\frac{1}{\delta} > \frac{2q+1}{q+1}$. □

4 Semi-online assignment in the max-min model

In this section we show algorithms for the cases $q \ge 1$ and $q < 1$. The two cases use the same basic algorithm with different choices of parameter.

Note that the competitive ratios for $q_1 > 1$ and $q_2 = \frac{1}{q_1} < 1$, i.e., two cases where the speed ratio is the same, we get different competitive ratios.

In the definitions of algorithms, we denote by ℓ_2 the current load of the second machine at each time. Recall that the total sum of processing times is 1. Let $\mathcal{R}_q = \max\{q + \frac{1}{q}, 1 + \frac{1}{q}\} \ge 2$

Algorithm 2

Upon arrival of a new job j,

1. If $g_j = 1$, assign j to the first machine.
2. If $g_j = 2$,
   (a) If $\ell_2 + p_j \le 1 - \frac{q}{(q+1)\mathcal{R}_q}$, assign j to machine 2.
   (b) Otherwise, if $q \cdot \mathcal{R}_q \cdot \ell_2 + p_j > 1$, assign j to machine 1.
   (c) Otherwise assign j to machine 2.

Theorem 3 *Algorithm 2 has a competitive ratio of at most $\mathcal{R}_q$, which is best possible.*

Proof Denote by ALG the cost of Algorithm 2. If there is no job with $g_j = 2$ then OPT $= 0$ and ALG $= 0$ and the competitive ratio is 1. Otherwise if $D > \frac{q}{q+1}$, then all the jobs such that $g_j = 2$ will be assigned to machine 2 by step 2(a). We can show that all jobs with $g_j = 2$ are assigned to this machine, and thus its load will be $1 - D \leq \frac{1}{q+1}$. The inequality $\frac{1}{q+1} \leq 1 - \frac{q}{(q+1)\mathcal{R}_q}$ is equivalent to $\mathcal{R}_q \geq 1$, and thus all jobs with $g_j = 2$ are assigned to machine 2 by step 2(a). As a result, the minimum load will be on machine 2, so OPT $= 1 - D$ and ALG $\geq 1 - D$, which gives a competitive ratio of 1 as well.

If $\frac{1}{q+1} < L_2 \leq 1 - \frac{q}{(q+1)\mathcal{R}_q}$, then the minimum load will be on machine 1, and OPT $\leq \frac{1}{q+1}$, ALG $\geq \frac{1}{(q+1)\mathcal{R}_q}$, and thus $\frac{\text{OPT}}{\text{ALG}} \leq \mathcal{R}_q$.

If $\frac{1}{(q+1)\mathcal{R}_q} \leq L_2 \leq \frac{1}{q+1}$, the minimum load will be on machine 2, then ALG $\geq \frac{1}{(q+1)\mathcal{R}_q}$, OPT $\leq \frac{1}{q+1}$. This implies that $\frac{\text{OPT}}{\text{ALG}} \leq \mathcal{R}_q$.

If $L_2 < \frac{1}{(q+1)\mathcal{R}_q}$, then the minimum load is on machine 2.

There are two cases that it can be happen. $D > \frac{q}{q+1}$, which we already considered, and else, some job j such that $g_j = 2$ was assigned to machine 1 by step 2(b) of the algorithm, then by the definition of the algorithm, the following conditions hold at the time of arrival of this job j.

1. $\ell_2 + p_j > 1 - \frac{q}{(q+1)\mathcal{R}_q}$, and thus since $\ell_2 \leq L_2 < \frac{1}{(q+1)\mathcal{R}_q}$ we have $p_j > 1 - \frac{q}{(q+1)\mathcal{R}_q} - \frac{1}{(q+1)\mathcal{R}_q} = 1 - \frac{1}{\mathcal{R}_q}$, since job j was not assigned by step 2(a).

2. $q\mathcal{R}_q\ell_2 + p_j > 1$ and so $L_2 \geq \ell_2 > \frac{1-p_j}{q\mathcal{R}_q}$, since the job was assigned by step 2(b).

We argue that this is the only job such that $g_j = 2$ on machine 1. If there are at least two such jobs i, j then $p_i + p_j > 2(1 - \frac{1}{\mathcal{R}_q}) \geq 1$ since $\mathcal{R}_q \geq 2$ for every value of q, in contradiction to $\Sigma = 1$. Therefore all the future jobs j' such that $g_{j'} = 2$ (if such jobs exist) will be assigned to machine 2. If OPT assigns the job j to machine 1, then using the load of machine 2, OPT $\leq 1 - p_j - D$. Since j is the only job assigned by the algorithm to machine 1 among jobs that machine 2 can run, we have ALG $= 1 - p_j - D$. In this case, the competitive ratio is 1. Otherwise if OPT assigns this job j to machine 2, by the load of the first machine OPT $\leq \frac{1-p_j}{q}$. ALG $\geq \frac{1-p_j}{q\mathcal{R}_q}$ so we get $\frac{OPT}{ALG} \leq \mathcal{R}_q$.

Finally, we are left with the case $L_2 > 1 - \frac{q}{(q+1)\mathcal{R}_q}$. This means that we did the last assignment on machine 2 at step 2(c). In this case we have that $L_2 > \frac{1}{q+1}$ so the minimum load is on machine 1.

We show that such an assignment can only happen once. If it happens for a job j, then after job j is assigned we have a load of more than $1 - \frac{q}{(q+1)\mathcal{R}_q}$ on machine 2, and for every $i > j$ for which the algorithm reaches step 2(b), $q\mathcal{R}_q\ell_2 + p_i > q\mathcal{R}_q(1 - \frac{q}{(q+1)\mathcal{R}_q}) = q\mathcal{R}_q - \frac{q^2}{q+1}$. If $q \geq 1$, we have $q\mathcal{R}_q - \frac{q^2}{q+1} \geq 2q - q = q \geq 1$ since $\mathcal{R}_q \geq 2$, $\frac{q^2}{q+1} \leq q$ and $q \geq 1$. If $q < 1$, we have $q\mathcal{R}_q - \frac{q^2}{q+1} \geq q + 1 - q = 1$ by the definition of $\mathcal{R}_q$ and again by $\frac{q^2}{q+1} \leq q$. Therefore each such job i is assigned to machine 1. If step 2(b) is not reached, since no further job can be assigned by step 2(a), this means that job i is assigned by step 1, and thus it is assigned to machine 1.

Denote by S_2 the load of machine 2 before the arrival of job j. Job j was not assigned by step 2(a) and we have $q\mathcal{R}_q S_2 + p_j \leq 1$, since j was not assigned by step 2(b) when this step was reached.

The smaller load is on the first machine, so ALG $\geq \frac{1-S_2-p_j}{q} \geq \frac{1-p_j}{q}(1-\frac{1}{q\mathcal{R}_q})$. Since the job p_j must be assigned to one of the machines, OPT $\leq \max\{1-p_j, \frac{1-p_j}{q}\}$.

If $q \geq 1$, $\max\{1-p_j, \frac{1-p_j}{q}\} = 1-p_j$ and ALG $\geq \frac{1-p_j}{q}(1-\frac{1}{q^2+1})$ using the definition of $\mathcal{R}_q$. This implies that $\frac{OPT}{ALG} \leq \frac{q^2+1}{q} = q+\frac{1}{q}$.

If $q < 1$, $\max\{1-p_j, \frac{1-p_j}{q}\} = \frac{1-p_j}{q}$ and ALG $\geq \frac{1-p_j}{q}(1-\frac{1}{q+1})$ using the definition of $\mathcal{R}_q$. This implies that $\frac{OPT}{ALG} \leq \frac{q+1}{q} = 1+\frac{1}{q}$.

To prove a lower bound for $q \geq 1$, we consider an algorithm $\mathcal{A}$ and the following sequence of jobs. The first job has $p_1 = \frac{1}{(q^2+1)(q+1)}$, $g_1 = 2$. If it is assigned to machine 1, a second job with $g_2 = 1$ and size $p_2 = \frac{q}{q+1} + \frac{q^2}{(q^2+1)(q+1)}$ arrives. It must be assigned to machine 1. If the sequence of the jobs terminates here it leads to an infinite competitive ratio since $\mathcal{A} = 0$ and OPT $\geq p_1$. Thus the first job must be assigned to machine 2. A second job arrives, where $p_2 = \frac{q}{q+1}$, $g_2 = 2$. If it is assigned to machine 2, a third job arrives with $p_3 = \frac{q^2}{(q^2+1)(q+1)}$, $g_3 = 2$, and it must be assigned to machine 1 (else we get zero load on the first machine and an infinite competitive ratio). For the sequence of three jobs, the optimal assignment is to assign the second job to the first machine and the other jobs to the second machine. This gives OPT $= \frac{1}{q+1}$, ALG $\leq \frac{q}{(q^2+1)(q+1)}$ and $\frac{\text{OPT}}{\text{ALG}} \geq q+\frac{1}{q}$. Otherwise if the second job is assigned to machine 1, a third job, where $p_3 = \frac{q^2}{(q^2+1)(q+1)}$, $g_3 = 1$ arrives, which must be assigned to machine 1. We terminate the sequence of jobs, and get OPT $\geq \frac{1}{q(q+1)}$ (by assigning the second job to machine 2 and the other jobs to machine 1) and ALG $= \frac{1}{(q^2+1)(q+1)}$. Then $\frac{\text{OPT}}{\text{ALG}} \geq q+\frac{1}{q}$.

To prove a lower bound for $q < 1$, we consider an algorithm $\mathcal{A}$ and the following sequence of jobs. The first job has $p_1 = \frac{q}{(q+1)^2}$, $g_1 = 2$. If it is assigned to machine 1, a second job with $g_2 = 1$ and size $p_2 = \frac{1}{q+1} + \frac{q^2}{(q+1)^2}$ arrives. It must be assigned to machine 1. If the sequence of the jobs terminates here it leads to an infinite competitive ratio since $\mathcal{A} = 0$ and OPT $> p_1$. Thus the first job must be assigned to machine 2. A second and third jobs arrive, where $p_2 = \frac{1}{q+1}$, $g_2 = 2$, $p_3 = \frac{q^2}{(q+1)^2}$, $g_3 = 1$. For the sequence of three jobs, the optimal assignment is to assign the second job to the second machine and the other jobs to the first machine. This gives OPT $= \frac{1}{q+1}$. If both jobs are assigned to the first machine we have ALG $\leq p_1$ and otherwise, ALG $\leq \frac{p_3}{q}$. In both cases, ALG $\leq \frac{q}{(q+1)^2}$ and $\frac{\text{OPT}}{\text{ALG}} \geq 1+\frac{1}{q}$. □

5 Fractional Semi-online assignment for both the max-min and the min-max models

In this section we show a simple algorithm which achieves an optimal solution and thus has competitive ratio 1 both for maximization of the minimum load and minimization of the maximum load.

We denote by ℓ_2 the current load of the second machine at each time. Recall that the total sum of processing times is 1. The algorithm below has the invariant that $\ell_2 \leq \frac{1}{q+1}$.

Algorithm 3

Upon arrival of a new job j,

1. If $g_j = 1$, assign j completely to the first machine.
2. If $g_j = 2$,
   (a) If $\ell_2 = \frac{1}{q+1}$, assign j to the first machine.
   (b) Otherwise if $\ell_2 + p_j \leq \frac{1}{q+1}$, assign j to the second machine.
   (c) Otherwise assign a part of j of size $\frac{1}{q+1} - \ell_2$ to machine 2 and the rest to machine 1.

Theorem 4 *Algorithm 3 is an optimal algorithm.*

Proof By definition, machine 2 cannot have a load of more than $\frac{1}{q+1}$. If it has exactly this load and then $L_1 = L_2 = \frac{1}{q+1}$. In this case the algorithm is optimal since the machines are balanced. Otherwise, the load of the second machine is smaller than the load of the first machine. In this case, all jobs with $g_j = 2$ were assigned to machine 2, and machine 1 only got jobs with $g_j = 1$, so this is an optimal schedule as well. □

6 Fractional assignment in the min-max model

In this section we design an algorithm for fractional assignment. The next algorithm is defined for any $q \in (0, \infty)$ and is similar to the algorithm for the max-min model, but uses different parameters.

Algorithm 4

Upon arrival of a new job j,

1. If $g_j = 1$, assign j completely to the first machine.
2. If $g_j = 2$, split it into two parts in the ratio $q^2 : q + 1$. The first machine receives a part of size $\frac{q^2 p_j}{q^2+q+1}$ of the job and the second machine receives a part of size $\frac{(q+1)p_j}{q^2+q+1}$ of the job.

Theorem 5 *Algorithm 4 has a competitive ratio of at most $\frac{(q+1)^2}{q^2+q+1}$, which is best possible.*

Proof Denote by ALG the cost of Algorithm 4. We have $L_2 = P_2 = \frac{q+1}{q^2+q+1}(\Sigma - D)$ and $P_1 = D + \frac{q^2}{q^2+q+1}(\Sigma - D) = \frac{q^2}{q^2+q+1}\Sigma + \frac{q+1}{q^2+q+1}D$ and so $L_1 = \frac{q}{q^2+q+1}\Sigma + \frac{q+1}{q(q^2+q+1)}D$.

We use the following bounds on OPT. Due to the sum of processing times, $\text{OPT} \geq \frac{\Sigma}{q+1}$. Since the first machine must receive jobs of a total processing time at least D, $\text{OPT} \geq \frac{D}{q}$.

If $D \leq \frac{q\Sigma}{q+1}$ and the maximum load is on machine 1, using $\text{OPT} \geq \frac{\Sigma}{q+1}$, $\frac{\text{ALG}}{\text{OPT}} \leq \frac{\frac{q}{q^2+q+1}\Sigma + \frac{q+1}{q(q^2+q+1)}D}{\frac{\Sigma}{q+1}} \leq \frac{\frac{q}{q^2+q+1}\Sigma + \frac{1}{q^2+q+1}\Sigma}{\frac{\Sigma}{q+1}} \leq \frac{(q+1)^2}{q^2+q+1}$.

If $D \leq \frac{q\Sigma}{q+1}$ and the maximum load is on machine 2, using $\text{OPT} \geq \frac{\Sigma}{q+1}$ and $D \geq 0$, $\frac{\text{ALG}}{\text{OPT}} \leq \frac{\frac{q+1}{q^2+q+1}(\Sigma - D)}{\frac{\Sigma}{q+1}} \leq \frac{\frac{q+1}{q^2+q+1}\Sigma}{\frac{\Sigma}{q+1}} \leq \frac{(q+1)^2}{q^2+q+1}$.

If $D \in (\frac{q\Sigma}{q+1}, \Sigma]$, the maximum load is on machine 1 (since $L_1 = \frac{q}{q^2+q+1}\Sigma + \frac{q+1}{q(q^2+q+1)}D > \Sigma\frac{q+1}{q^2+q+1} > \frac{\Sigma}{q+1}$), using $\text{OPT} \geq \frac{D}{q}$ and $\Sigma \leq \frac{q+1}{q}D$, $\frac{\text{ALG}}{\text{OPT}} \leq \frac{L_1}{D/q} = \frac{\frac{q^2}{q^2+q+1}\Sigma + \frac{q+1}{q^2+q+1}D}{D} \leq \frac{q(q+1)+q+1}{q^2+q+1} = \frac{(q+1)^2}{q^2+q+1}$.

To prove a lower bound, we consider an algorithm $\mathcal{A}$ and the following sequence of jobs. The first job has $g_1 = 2$, $p_1 = 1$. Let δ be the part of this job assigned by $\mathcal{A}$ to the second machine. At this time the best way to split the job is into parts of size $\frac{q}{q+1}$ and $\frac{1}{q+1}$ and thus $\text{OPT} = \frac{1}{q+1}$.

If $\delta \geq \frac{q+1}{q^2+q+1}$, then the loads of the second machine is $L_2 = \delta$ and the competitive ratio is at least $\frac{\delta}{1/(q+1)} \geq \frac{(q+1)^2}{q^2+q+1}$. Otherwise, the sequence is continued by an additional job with $g_2 = 1$, $p_2 = q$. After this job we have $\text{OPT} = 1$. Clearly, this job must run on the first machine, and we have $L_1 = \frac{1-\delta+q}{q} > \frac{\frac{q^2}{q^2+q+1}+q}{q} = \frac{(q+1)^2}{q^2+q+1}$ which is also the competitive ratio. □

References

[1] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts. On-line load balancing with applications to machine scheduling and virtual circuit routing. *J. ACM*, 44:486–504, 1997.

[2] B. Awerbuch, Y. Azar, E. F. Grove, M.-Y. Kao, P. Krishnan, and J. S. Vitter. Load balancing in the l_p norm. In *Proc. 36th Symp. Foundations of Computer Science (FOCS)*, pages 383–391. IEEE, 1995.

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

[4] Y. Azar, J. Naor, and R. Rom. The competitiveness of on-line assignments. *J. Algorithms*, 18:221–237, 1995.

[5] N. Bansal and M. Sviridenko. The santa claus problem. In *38th ACM Symposium on Theory of Computing (STOC2006)*, pages 31–40, 2006.

[6] A. Bar-Noy, A. Freund, and J. Naor. On-line load balancing in a hierarchical server topology. *SIAM J. Comput.*, 31:527–549, 2001.

[7] P. Berman, M. Charikar, and M. Karpinski. On-line load balancing for related machines. *J. Algorithms*, 35:108–121, 2000.

[8] P. Crescenzi, G. Gambosi, and P. Penna. On-line algorithms for the channel assignment problem in cellular networks. *Discrete Applied Mathematics*, 137(3):237–266, 2004.

[9] G. Dósa and L. Epstein. Preemptive scheduling on a small number of hierarchical machines. manuscript, 2006.

[10] L. Epstein. Tight bounds for online bandwidth allocation on two links. *Discrete Applied Math.*, 148(2):181–188, 2005.

[11] R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical J.*, 45:1563–1581, 1966.

[12] Y.-W. Jiang, Y. He, and C.-M. Tang. Optimal online algorithms for scheduling on two identical machines under a grade of service. *Journal of Zhejiang University SCIENCE A*, 7(3):309–314, 2006.

[13] J. Park, S. Y. Chang, and K. Lee. Online and semi-online scheduling of two machines under a grade of service provision. *Operations Research Letters*, 34(6):692–696, 2006.