The cost of selfishness for maximizing the minimum load on uniformly related machines

Leah Epstein* Elena Kleiman† Rob van Stee‡

Abstract

Consider the following scheduling game. A set of jobs, each controlled by a selfish agent, are to be assigned to m uniformly related machines. The cost of a job is defined as the total load of the machine that its job is assigned to. A job is interested in minimizing its cost, while the social objective is maximizing the minimum load (the value of the cover) over the machines. This goal is different from the regular makespan minimization goal, which was extensively studied in a game theoretic context.

We study the price of anarchy (POA) and the price of stability (POS) for uniformly related machines. The results are expressed in terms of s, which is the maximum speed ratio between any two machines. For uniformly related machines, we prove that the POS is unbounded for $s > 2$, and the POA is unbounded for $s \geq 2$. For the remaining cases we show that while the POA grows to infinity as s tends to 2, the POS is at most 2 for any $s \leq 2$.

1 Introduction

Job scheduling games have been widely studied in recent years. In this work, we consider the problem of maximizing the minimum load, seeing jobs as selfish agents who are only interested in their own costs. For applications of this problem and related models see [7, 17].

The model of scheduling on uniformly related machines is defined as follows. A set of n jobs $J = \{1, 2, \ldots, n\}$ is to be assigned to a set of m machines $M = \{1, \ldots, m\}$, where machine i has a speed s_i. If $s_i = 1$ for $i = 1, \ldots, m$, then the machines are called identical. Without loss of generality, we assume $1 = s_1 \leq s_2 \leq \cdots \leq s_m = s$. The value s is called the speed ratio, and it is the maximum speed ratio between any pair of machines. The size of job k (for $1 \leq k \leq n$) is denoted by p_k. An assignment or schedule is a function $\mathcal{A} : J \to M$. Thus $\mathcal{A}(k)$, also denoted by $\mathcal{A}_k$, is the index of the machine that job k is assigned to. The load of machine i, which is also called the delay of this machine, is $L_i(\mathcal{A}) = \sum_{k:\mathcal{A}_k=i} \frac{p_k}{s_i}$. The value, or the *social value* of a schedule $\mathcal{A}$ is the minimum delay of any machine, also known as the value of the *cover*. We denote it by COVER($\mathcal{A}$). We study the problem of maximizing the value of the cover [4]. This problem is dual to the makespan scheduling problem [15].

The non-cooperative machine covering game MC is characterized by a tuple $MC = \langle N, (\mathcal{M}_k)_{k\in N}, (c_k)_{k\in N}\rangle$, where N is the set of atomic players. Each player $k \in N$ controls a single job of size $p_k > 0$ and selects the machine to which it will be assigned. We associate each player with the job it wishes to run, that is, $N = J$. The set of strategies $\mathcal{M}_k$ for each job $k \in N$ is the set M of all machines, i.e., $\mathcal{M}_k = M$. Each job must be assigned to one machine, and preemption is not allowed (i.e., the

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

†Department of Mathematics, University of Haifa, 31905 Haifa, Israel. elena.kleiman@gmail.com.

‡Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany. vanstee@mpi-inf.mpg.de.

set of choices of jobs result in a schedule as it is defined above). The outcome of the game is an assignment $\mathcal{A} = (\mathcal{A}_k)_{k\in N} \in \times_{k\in N}\mathcal{M}_k$ of jobs to the machines, where $\mathcal{A}_k$ for each $1 \leq k \leq n$ is the index of the machine that job k chooses to run on. Let $\mathcal{S}$ denote the set of all possible assignments. The cost function of job $k \in N$ is denoted by $c_k : \mathcal{S} \to \mathbb{R}$. The cost $c_k(\mathcal{A})$ charged from job k running on machine i in a given assignment $\mathcal{A}$ (a job k such that $\mathcal{A}_k = i$) is defined to be the load observed by machine i in this assignment, that is, $c_k(\mathcal{A}) = L_i(\mathcal{A})$. The goal of each (selfish) job is to run on a machine with a load which is as small as possible. An assignment $\mathcal{A}$ is a (pure) Nash equilibrium (NE), if every job k satisfies the following. Let $i = \mathcal{A}_k$. There cannot exist a machine $i' \neq i$ for which $L_{i'}(\mathcal{A}) + \frac{p_k}{s_{i'}} < L_i(\mathcal{A})$. That is, a schedule is an NE, if no job can obtain a lower cost by changing its strategy while all other jobs keep their strategies unchanged. For this selfish goal of players, a pure NE (with deterministic agent choices) always exists [14, 11].

In our scheduling model, the *coordination ratio*, or *price of anarchy (*POA*)* [19] is the worst case ratio between the social value (i.e., the minimum delay of any machine, or the value of the cover) of an optimal schedule, denoted by OPT, and the social value of any NE. If both these values are 0 then we define the POA to be 1. The *price of stability (*POS*)* [1] is the worst case ratio between the social value of an optimal solution, and the social value of the *best* NE. Similarly, if both these values are 0 then we define the POS to be 1. The POA and POS can be defined for one game (a specific instance of jobs and machines), but also for the complete set of games, or for subclasses of games. We let $\text{POA}(m, s)$ and $\text{POS}(m, s)$ denote the POA and POS, respectively, for games with m uniformly related machines where $1 = s_1 \leq s_2 \leq \cdots \leq s_m = s$. We also let $\text{POA}(m) = \text{POA}(m, 1)$, and $\text{POS}(m) = \text{POS}(m, 1)$, that is, the POA and the POS for all games on identical machines, as functions of the number of machines.

The non-selfish version of the problem has been well studied (known by different names such as "machine covering" and the "Santa Claus problem") in the computer science literature (see e.g. [4, 2, 6, 8]). Various game-theoretic aspects of max-min fairness in resource allocation games were considered, but unlike the makespan minimization problem for which the POA and the POS were extensively studied (see [17, 3, 18]), these measures were not previously considered for the uncoordinated machine covering problem in the setting of selfish jobs and uniformly related machines. A different model, where machines are selfish rather than jobs (with the same social goal function) was studied recently in [10, 5, 9].

In [7], we considered the problem for identical machines. We showed that the POS is equal to 1 for every game. Additionally, we showed close bounds on the POA for the complete set of games, namely, that it is at least 1.691 and at most 1.7. For small numbers of machines we showed that $\text{POA}(2) = \text{POA}(3) = \frac{3}{2}$ and $\text{POA}(4) = \frac{13}{8}$.

In contrast to these results, we can show that for uniformly related machines even the POS for the complete set of instances is unbounded. This holds already for $\text{POS}(2, s)$ such that $s > 2$, and $\text{POA}(2, s)$ is unbounded already for $s \geq 2$. The same property holds for m machines and the same speed ratios, that is $\text{POS}(m, s)$ is unbounded for any $m \geq 2$ and $s > 2$, and $\text{POA}(m, s)$ is unbounded for any $m \geq 2$ and $s \geq 2$. We present examples proving these last properties (see also [7]). We show that $\text{POS}(m, s)$ is at most 2 if $s \leq 2$, and $\text{POA}(m, s) = \Theta(\frac{1}{2-s})$ if $s < 2$. These results are very different from the situation for the makespan minimization social goal. For that problem, the POS is 1 for every game. Czumaj and Vöcking [3] showed that the overall POA is $\Theta(\frac{\log m}{\log \log m})$ (see also [12, 17]). Note that for identical machines, the results given in [7] are more similar to the situation for makespan minimization, where the POS is 1 for every game, and the $\text{POA}(m)$ is constant [13, 20].

Kleiman [16] provided additional results for two machines. First, it is shown in this dissertation that we can restrict our analysis of $\text{POA}(2, s)$ to instances involving no more than four jobs. Thus, we only need to consider a small number of cases corresponding to the identity of non-zero sized

jobs, and to the machine which defines the value of the objective function in the worst NE. For each of these cases a linear program (LP) is formulated whose optimal objective function value is equal to $\text{POA}(2, s)$ for any s (or for some subinterval of s, depending on the case) for a suitable subclass of instances. The function $\text{POA}(2, s)$ for each resulting subinterval is the maximum of these values. It is possible to find the tight values by presenting a primal optimal solution and a dual optimal solution for each of these linear programs (and their dual programs). The idea for this analysis is motivated by the classic paper by Graham [15], where a similar approach to analyze the performance of an algorithm for a variant of the makespan minimization problem for m identical machines was used. Kleiman [16] also presented upper bounds for $\text{POS}(2, s)$ showing that it is at most $\frac{3}{2}$ (for $s \leq 2$).

2 The case $s \geq 2$

We show that for sufficiently large speed ratios, $\sup_{s\geq 1} \text{POA}(m, s)$ and $\sup_{s\geq 1} \text{POS}(m, s)$ are unbounded for any $m \geq 2$. Recall that $1 = s_1 \leq \cdots \leq s_m = s$. The following result was already claimed in [7]; here we present a full proof.

Theorem 1. *For any $m \geq 2$ and $s > 2$, $\text{POS}(m, s) = \infty$. For any $m \geq 2$ and $s \geq 2$, $\text{POA}(m, s) = \infty$.*

Proof. Consider a set of uniformly related machines with speed ratio s. We show that if $s > 2$, then the ratio between $\text{COVER}(\text{OPT})$ the the value of the cover of any NE is unbounded, and if $s \geq 2$, then the ratio between $\text{COVER}(\text{OPT})$ the the minimum value of the cover of any NE is unbounded. This is slightly stronger than the statement, since we show the property for every set of m machines rather than for one specific set of m machines.

Assume $s > 2$, and consider an instance that contains m jobs, each of size s. It is not hard to see that for this input $\text{COVER}(\text{OPT}) = 1$. We show that any assignment where each job is assigned to a distinct machine is not an NE. Such assignments are the only ones for which the value of the cover is positive. In fact, in such an assignment, the job assigned to the first machine has cost s, while if it moves to the m-th machine, its load becomes $\frac{2s}{s} = 2 < s$. Thus, any NE assignment has a value of 0 and the claim follows.

Assume $s = 2$, and consider the assignment $\mathcal{A}$ (of the job set defined above) where each machine is assigned a single job, except that the first machine has no jobs, and the m-th machine has two jobs. It is not difficult to see that this is an NE. Each job assigned to the m-th machine will not move to the first machine since it will have the same cost after moving. It would not move to another machine since it would be assigned there together with another job, and this machine is not faster than its current machine. A job assigned to any machine out of $2, 3, \ldots, m-1$ would not move to the first machine, since it is not faster. Moving to a machine which has at least one job assigned to it would result in load of at least 2, while the current load of the machine of this job is at most 2. Thus, we have presented an instance for which $\text{COVER}(\text{OPT}) = 1$ and $\text{COVER}(\mathcal{A}) = 0$, which implies the claim for $s = 2$. □

3 The price of anarchy for $s \leq 2$

Next, we characterize the situation in all other cases, which were not considered in the previous section. We can show that if we let $\varepsilon = 2 - s$, then $\text{POA}(m, s)$ grows with $1/\varepsilon$. For the upper bound, we define a weight function $g(x)$ to be applied on sizes of jobs.

$$g(x) = \begin{cases} 1 & \text{for} \quad x \geq 1 \\ x & \text{for} \quad 0 < x < 1 \end{cases}$$

The next claim is obvious.

Claim 2. *Let I be a set of jobs. The total size of jobs in I is at least 1 if and only if their total weight is at least 1. In addition, if their total weight is strictly larger than 1, then $|I| \geq 2$.*

Theorem 3. *Let $0 < \varepsilon < 1$. Then* $\text{POA}(m, 2-\varepsilon) = \Theta(\frac{1}{\varepsilon})$.

Proof. Consider an instance with $\text{COVER}(\text{OPT}) = 1$, and an arbitrary NE assignment $\mathcal{A}$ for this instance. Let P be the least loaded machine in $\mathcal{A}$, and let W denote the total weight of jobs assigned to P (if no job is assigned to P, then $W = 0$). Recall that s_P denotes the speed of machine P. We use Claim 2 to analyze the input. If $W \geq 1$, then the total size of jobs is at least 1, thus the load of P is at least $\frac{1}{s_P} \geq \frac{1}{2}$.

If $W < 1$, then since every machine in OPT has a load of at least 1 and all speeds are at least 1, each such machine has a total size of jobs of at least 1 and thus weight of at least 1 as well. Therefore, the total weight of all jobs is at least m. Therefore, as $W < 1$, there exists a machine in $\mathcal{A}$ with a total weight of jobs strictly above 1, and thus at least two jobs assigned to it. Denote this machine by Q (and its speed by s_Q).

Let p denote the total size of jobs assigned to P and let q denote the total size of jobs assigned to Q. Since the total weight of jobs assigned to Q is above 1, we have $q \geq 1$. Let a be the size of a smallest size job assigned to Q (and thus $q \geq 2a$). Since the job of size a has no incentive to move to machine P, we have $\frac{p+a}{s_P} \geq \frac{q}{s_Q}$ or equivalently, $\frac{p}{s_P} > \frac{q}{s_Q} - \frac{a}{s_P} \geq \frac{q}{2-\varepsilon} - a$, using $1 \leq s_P$ and $s_Q \leq 2 - \varepsilon$.

If $a \geq \frac{1}{3}$, then using $q \geq 2a$ we have $\frac{q}{2-\varepsilon} - a \geq \frac{2a-2a+a\varepsilon}{2-\varepsilon} \geq \frac{\varepsilon}{6}$. If $a < \frac{1}{3}$, then we have $\frac{q}{2-\varepsilon} - a \geq \frac{1}{6}$, since $q \geq 1$. Thus the load of P is $\Omega(\varepsilon)$, and since $\text{COVER}(\text{OPT}) = 1$, we have a ratio of $O(\frac{1}{\varepsilon})$.

For the lower bound, consider an instance with m identical large jobs, each of size $s = 2 - \varepsilon$, and one small job of size ε. We analyze the schedule which assigns one large job to each machine, except for machine m which receives two such jobs, and machine 1 which receives the small job.

Similar to the proof of Theorem 1, the large jobs assigned to machines $2, 3, \ldots, m-1$ have no incentive to move. Similarly, the jobs assigned to machine m have no incentive to move to any machine, except for possibly the first machine. We have $\frac{2s}{s} = 2 = s + \varepsilon$, and therefore these jobs do not have an incentive to move. Finally, the small job would not move since every machine except for machine 1 is already loaded by at least 1. For this instance, $\text{COVER}(\text{OPT}) \geq 1$, while $\text{COVER}(\mathcal{A}) = \varepsilon$. □

4 The price of stability for $s \leq 2$

In this section we show that $\text{POS}(m, s)$ is finite for any $m \geq 2$ and any $s \leq 2$. For this purpose we use a well-known algorithm for scheduling, called LPT (see [15]). This algorithm sorts the jobs in a non-increasing order of their sizes, and greedily assigns each job to the machine which would have a smaller load (taking the speed into account) as a result of assignment of the job. We use a special case of LPT which acts as follows. In a case of a tie (that is, if according to the assignment rule, a job could be assigned to multiple machines, each of which would have the same load if the job is assigned to it), it prefers a machine with a smaller load before the assignment. If there are multiple machines having the same load before the assignment, then it assigns the job to the machine which has the largest index among the candidate machines (recall that machines are sorted by non-decreasing speed).

Claim 4. *For any sorted input, if there are at least m jobs in the input, then for $s \leq 2$,* LPT *assigns the i-th job (for $1 \leq i \leq m$) to machine $m - i + 1$.*

Proof. Consider a sorted input, where the size of the j-th job is denoted by p_j, and we have $p_1 \geq p_2 \geq \ldots \geq p_m$. We show the claim by induction. We prove that the first job is assigned to the m-th machine. At the time of its arrival all machines have load zero. Thus the algorithm chooses one of the fastest machines, and machine m is one of them. If there is a unique fastest machine, then this is machine m. Otherwise, if there are multiple fastest machines, then the fastest machine of largest index is chosen, and this is machine m as well. Assume now that jobs $1, 2, \ldots, k-1$ ($2 \leq k \leq m$) have been assigned as claimed. Consider the k-th job (of size p_k). If the machine to which this job is assigned is chosen among the machines of indices $1, 2, \ldots, m-k+1$, then (as these machines received no jobs so far) this is a machine of maximum speed out of these machines. This is machine $m-k+1$ is there is a unique such machine, and if there are multiple such machines, then machine $m-k+1$ is one of them, and it has the maximum index among such machines. Otherwise, if the job is assigned to one of the machines $m-k, \ldots, m$, then the resulting load must be strictly smaller than the load that would have resulted from assigning job k to machine $m-k+1$ (since machines with smaller load before the assignment are preferred). Assume by contradiction that the job is assigned to machine $\ell > m-k+1$. By the inductive hypothesis, this machine has exactly one job at this time, which is job $m-\ell+1$. The resulting load would become $\frac{p_{m-\ell+1}+p_k}{s_\ell} \geq p_k$ (since $p_{m-\ell+1} \geq p_k$ and $s_\ell \leq 2$). The load resulting from assigning the job to machine $m-k+1$ is $\frac{p_k}{s_{m-k+1}} \leq p_k$, which is a contradiction. □

It is known that for scheduling games on uniformly related machines with the same selfish goal of the players as here, LPT produces an NE schedule [14]. Given the tie-breaking rule, for a given sorted list of jobs, and a sorted list of machines, this algorithm has exactly one possible output for each input. The value of the POS is upper-bounded by its approximation ratio (for our goal function). This is one of the tools which we use in order to derive an upper bound on the POS.

Theorem 5. *For $m \geq 2$, if $s \leq 2$, then* $\text{POS}(m, s) \leq 2$.

Proof. We will show that for every set of uniformly related machines with speed ratio s, the ratio between the value of the cover in an optimal solution and the value of the cover in the best NE is no larger than 2.

In this proof we will neglect instances for which the value of the optimal cover is zero; for every such instance every NE is an optimal solution. By normalizing all remaining instances, we may assume the optimal cover of an instance has a value of 1. We will call such instances *valid instances*. It is sufficient to analyze the POS on valid instances.

Suppose that at least one valid counterexample instance (ordered sets of jobs and machines, where the speeds of machines are in $[1, 2]$) exists for some $m \geq 2$, for which the value of the cover of *every* NE is strictly below $1/2$. Thus, since LPT always creates an NE schedule, for such an instance LPT creates a schedule in which there is at least one machine with load strictly below $1/2$. We conclude that there exists a valid instance for which LPT creates a schedule with at least one machine of load strictly below $1/2$. We call such an instance *a counterexample for* LPT. In what follows, we only discuss counterexamples for LPT. Let m denote the number of machines in one such counterexample.

Consider all valid instances for m machines (of speeds in $[1, 2]$), for which the value of the cover of the schedule created by LPT is strictly below $1/2$ (as explained above, this set is non-empty). Let T be the infimum total size of jobs among all valid counterexamples for the considered number of machines m. Given the machine speeds and the value of the optimal cover, the total size of the jobs in all counterexamples is at least m, and thus such a positive infimum value must exist. Let I be a counterexample for which the total size of all the jobs is at most $T + 1/(16m)$. We assume that the jobs of I are sorted such that $p_1 \geq p_2 \geq \ldots$, and that LPT is applied on the jobs in this order (of

indices). Consider the NE assignment $\mathcal{A}$ which is defined by the LPT assignment of the jobs in I. We will derive a contradiction from the property that there is a machine of load strictly below $\frac{1}{2}$ in $\mathcal{A}$. This will be done by defining an alternative valid instance I', which is fairly similar to I, but the total size of jobs in it will be strictly below T. We will show that the value of the cover created by applying LPT on the alternative instance is the same as the one for I (that is, the same as in $\mathcal{A}$), while the value of the cover of an optimal solution remains 1. This will contradict the choice of T, since I' is a valid instance for m machines where the value of the cover in the schedule created by LPT is strictly below $\frac{1}{2}$, but the total size of jobs is below T.

Claim 6. *The input I satisfies $0 < p_m < 1$.*

Proof. Since the value of the optimal cover is 1, there are at least m jobs of positive sizes. On the other hand, assume by contradiction $p_m \geq 1$. By Claim 4, LPT assigns a job of an index in $\{1, 2, \ldots, m\}$ to every machine. Specifically, we find that the load of machine i will be at least $\frac{p_{m-i+1}}{s_i} \geq \frac{1}{2}$, since $p_{m-i+1} \geq 1$, and $s_i \leq 2$. This contradicts the assumption that the value of the cover of $\mathcal{A}$ is strictly below 1/2. □

Let P be the least loaded machine in $\mathcal{A}$. By assumption, the load of P is less than 1/2. We show in the next claim that the set of machines with loads at least 1 remains fixed once the first m jobs have been assigned (until LPT terminates), and other machines will have loads below 1 in $\mathcal{A}$.

Claim 7. *During the run of* LPT *on I, the set of machines with load at least 1 is fixed after the first m jobs have been assigned. These machines do not receive further jobs.*

Proof. By the definition of LPT, P received one of the m largest jobs (by Claim 4). After this and until LPT is done, P had load of less than 1/2 (since this is its final load). Jobs of indices $m+1, \ldots$ have sizes no larger than the first job assigned to P, so any additional single job that P could have received after its first job would increase the load of P to less than 1. This means that no assignment of any later job j increases a load of any machine to at least 1 once each machine has a job, because LPT would assign such a job j to P instead. For the same reason, a machine which already has a load of at least 1 cannot receive any later job. □

We next show that there is a machine in $\mathcal{A}$ with load at least $1 + 1/(4m)$. Let Q' be a machine of maximum load in $\mathcal{A}$.

Claim 8. *Q' has load at least $1 + 1/(4m)$, and $Q' > 1$.*

Proof. To prove the first claim, assume by contradiction that it does not hold, and every machine has load less than $1 + 1/(4m)$, whereas P has load less than 1/2. Based on $\mathcal{A}$, the total size of all jobs is at most

$$\sum_{i=1}^{m} s_i \left(1 + \frac{1}{4m}\right) - s_P \left(1 + \frac{1}{4m}\right) + \frac{s_P}{2} < \sum_{i=1}^{m} s_i + \frac{2m}{4m} - \frac{s_P}{2} \leq \sum_{i=1}^{m} s_i,$$

since $s_P \geq 1$, and $s_i \leq 2$ for all i. On the other hand, since COVER(OPT) = 1, the total size of all the jobs must be at least $\sum_{i=1}^{m} s_i$. This is a contradiction.

If $Q' = 1$, then since by Claims 4 and 7 the load of machine 1 is above 1 already after it receives job m, we have $p_m \geq (1 + 1/(4m))s_1 > 1$. This contradicts Claim 6. □

Let $1 \leq Q \leq m-1$ be such that $m-Q+1$ is a machine in $\mathcal{A}$ with load of at least $1+\frac{1}{4m}$. Such a machine exists by Claim 8. Let $x = p_Q \geq (1+\frac{1}{4m}) \cdot s_{m-Q+1} \geq 1+\frac{1}{4m}$. We modify I into an instance I' as follows. The instance contains a job of size $x - \frac{1}{8m}$ instead of job Q, while the other jobs are not modified. The sorted order for I' is based on the ordering for I, with the only change that the job of reduced size being moved to its proper position. If there are at least two possible positions, that is, I contains at least one job of size $x - \frac{1}{8m}$, then the job of reduced size is inserted to be the last job in the ordering out of jobs of size $x - \frac{1}{8m}$. We will show that the value of the optimal cover for I' is at least 1. Thus, since I' results from I by decreasing the size of a job, the value of the optimal cover for I' cannot be above 1, and it will follow that the value of the optimal cover for I' is exactly 1, which will imply that I' is a valid instance.

Lemma 9. *Consider an optimal schedule* OPT *for I (whose value of the cover is 1). There exists a schedule for I' whose value of cover is at least 1.*

Proof. We show how to modify OPT into the required schedule for I'. Seeing this schedule as a schedule for I', since the only difference between I and I' is the size of Q, the only machine of load below 1 may be the machine on which Q is scheduled.

Case 1. Job Q is assigned to a machine $i \in \{1, \ldots, m-Q+1\}$ in OPT.

We use the same schedule for I', and show that the load of machine i is at least 1 even with the modified size of Q. In this schedule the load of i is at least $\frac{x-\frac{1}{8m}}{s_i} \geq \frac{x-\frac{1}{8m}}{s_{m-Q+1}} = \frac{x}{s_{m-Q+1}} - \frac{1}{8m \cdot s_{m-Q+1}} \geq 1 + \frac{1}{4m} - \frac{1}{8m} > 1$, since $s_i \leq s_{m-Q+1}$, $\frac{x}{s_{m-Q+1}} \geq 1+\frac{1}{4m}$, and $s_{m-Q+1} \geq 1$.

Case 2. Job Q is assigned to a machine $i' \in \{m-Q+2, \ldots, m\}$ in OPT, but there exists a job $j < Q$ assigned to a machine $i \in \{1, \ldots, m-Q+1\}$ in OPT.

Create a schedule for I' which is identical to OPT, and swap the locations of Q and j. As in the previous case, the machine i that receives Q will have load above 1 in this schedule. Since $p_j \geq x$, the load of machine i' is at least its load in OPT (for I).

Case 3. Jobs $1, \ldots, Q$ are assigned to machines $m-Q+2, \ldots, m$ in OPT.

The number of these machines is $Q-1$, while the number of jobs is Q. Thus, there exists a machine $i \in \{m-Q+2, \ldots, m\}$ which has at least two jobs out of jobs $1, \ldots, Q$ in OPT.

Case 3.1. Job Q is assigned to i.

We use OPT as a schedule for I'. The load of i in this schedule is at least $\frac{x+(x-\frac{1}{8m})}{s_i} \geq \frac{2+\frac{3}{8m}}{s_i}$, as the size of every job in $\{1, \ldots, Q-1\}$ is at least $x \geq 1+\frac{1}{4m}$. Using $s_i \leq 2$ we find that the load of i is above 1.

Case 3.2. Job Q is assigned to a machine $i' \neq i$, $i' \in \{m-Q+2, \ldots, m\}$.

Machine i has job $j < Q$ (and at least one other job out of $1, \ldots, Q-1$). We create a schedule from OPT by swapping the locations of jobs Q and j. The load of the machine i' which receives j instead of Q is at least its load in OPT (for I), since $p_j \geq x$. Machine i has two jobs of the two sizes $x - \frac{1}{8m}$, and at least x. As in case 3.1, its load is above 1. □

Let $\mathcal{A}'$ be the schedule resulting from running LPT on I'.

Claim 10. *A machine has load of at least 1 in $\mathcal{A}$ if and only if it has load of at least 1 in $\mathcal{A}'$. A machine of load below 1 has the same set of jobs in both $\mathcal{A}$ and $\mathcal{A}'$.*

Proof. By Claim 4 (which holds for any run of LPT), the first m jobs are assigned to m distinct machines for both schedules. Clearly, the job whose size was reduced may be assigned later by LPT (for I' compared to I), as its position in the sorted order may change, and thus this job may be assigned to a machine of a smaller index in $\mathcal{A}'$. Note that in this discussion p_j still refers to the size of job j in I, and the jobs indices are as in I. Let $1 \leq r \leq m$ be such that $p_j \geq x - \frac{1}{8m}$ for $j \leq r$,

and $p_j < x - \frac{1}{8m}$ for $j > r$. We have $r \geq Q$, since for $j \leq Q$, $p_j \geq p_Q > x - \frac{1}{8m}$, and $r < m$ since $x - \frac{1}{8m} \geq 1 + \frac{1}{4m} - \frac{1}{8m} > 1$, while $p_m < 1$ by Claim 6. Thus, in the new ordering of jobs, the job of modified size is still one of the $m-1$ first jobs, but it may be assigned to a machine of lower index (in $\mathcal{A}'$), as explained above. Specifically, in $\mathcal{A}'$, for $1 \leq j \leq Q-1$, machine $m-j+1$ has job j, machine $m-r+1$ has the job of modified size (job Q), for $Q \leq j \leq r-1$, machine $m-j+1$ has job $j+1$, and for $r+1 \leq j \leq m$, machine $m-j+1$ has job j.

By Claim 7, the set of machines of load at least 1 in $\mathcal{A}$ is determined for I by the machines having such a load after the first m jobs were assigned. Consider the assignment of the first m jobs by LPT for I and for I'. We will show that after the first m jobs were scheduled, the machines $m-r+1, \ldots, m-Q+1$ have loads of at least 1 in both schedules. Each remaining machine receives exactly the same job (with the same size) in both schedules. The speed of every machine $m-r+1 \leq i \leq m-Q+1$ satisfies $s_i \leq s_{m-Q+1}$. In both schedules, these machines receive jobs of sizes at least $x - \frac{1}{8m}$; these are jobs $Q, \ldots, r$ (where job Q has a different size in I'). Since $\frac{x}{s_{m-Q+1}} \geq 1 + \frac{1}{4m}$, we find $\frac{x-\frac{1}{8m}}{s_{m-Q+1}} \geq 1 + \frac{1}{4m} - \frac{1/(8m)}{s_{m-Q+1}} > 1$, since $s_{m-Q+1} \geq 1$. Using $s_i \leq s_{m-Q+1}$ for $m-r+1 \leq i \leq m-Q+1$ we find $\frac{x-\frac{1}{8m}}{s_i} > 1$, and thus no matter which job of size at least $x - \frac{1}{8m}$ is assigned to which machine, all these machines have loads above 1.

We prove that the assignment of the remaining jobs (after the first m jobs) by LPT is identical for the two inputs. Assume by contradiction that it is not, and let $k > m$ be the first job assigned differently. Since in $\mathcal{A}$, machine P has load below $\frac{1}{2}$ at termination, this is also the case after the first m jobs have been assigned. Machine P cannot be one of the machines $m-r+1 \leq i \leq m-Q+1$, since they have load above 1 in both schedules after m jobs have been assigned. Thus, P has the same job (with the same size) in both $\mathcal{A}$ and $\mathcal{A}'$ after m jobs have been assigned. Jobs $m+1, \ldots, k-1$ are assigned in $\mathcal{A}'$ as in $\mathcal{A}$, so the load of P remains below $\frac{1}{2}$ just before k is assigned (in both schedules). Similar to the proof of Claim 7, k cannot be assigned to a machine of load of at least 1 (neither for I nor for I'), and thus it is assigned to one of the machines that have exactly the same jobs in both schedules at that time. Since the algorithm has a fixed tie-breaking rule, it must act in the same way for both I and I', which means that k is assigned as in $\mathcal{A}$, which is a contradiction. □

Thus, we find that machine P has load below $\frac{1}{2}$ in $\mathcal{A}'$ as well. A contradiction was reached, and thus we have proved the theorem.

□

We note that unlike $\text{POA}(m,s)$ which continuously grows from constant values to infinite values as a function of s, $\text{POS}(m,s)$ jumps from a constant value for $s=2$ to ∞ for $s>2$.

References

[1] E. Anshelevich, A. Dasgupta, J. M. Kleinberg, É. Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. *SIAM J. Comp.*, 38(4):1602–1623, 2008.

[2] N. Bansal and M. Sviridenko. The Santa Claus problem. In *Proc. 38th ACM Symp. Theo. Comp. (STOC'06)*, pages 31–40, 2006.

[3] A. Czumaj and B. Vöcking. Tight bounds for worst-case equilibria. *ACM Trans. Alg.*, 3(1), 2007.

[4] B. L. Deuermeyer, D. K. Friesen, and M. A. Langston. Scheduling to maximize the minimum processor finish time in a multiprocessor system. *SIAM J. Discr. Math.*, 3(2):190–196, 1982.

[5] P. Dhangwatnotai, S. Dobzinski, S. Dughmi, and T. Roughgarden. Truthful approximation schemes for single-parameter agents. *SIAM J. Comput.*, 40(3):915–933, 2011.

[6] T. Ebenlendr, J. Noga, J. Sgall, and G. J. Woeginger. A note on semi-online machine covering. In *Approx. Online Alg., Third Int. Workshop (WAOA'05)*, pages 110–118, 2005.

[7] L. Epstein, E. Kleiman, and R. van Stee. Maximizing the minimum load: The cost of selfishness. In *Proc. 5th Intl. Workshop Internet Netw. Econ. (WINE'09)*, pages 232–243, 2009.

[8] L. Epstein, A. Levin, and R. van Stee. Max-min online allocations with a reordering buffer. *SIAM J. Discrete Math.*, 25(3):1230–1250, 2011.

[9] L. Epstein, A. Levin, and R. van Stee. A unified approach to truthful scheduling on related machines. *CoRR*, abs/1207.3523, 2012. To appear in Proc. of SODA'13.

[10] L. Epstein and R. van Stee. Maximizing the minimum load for selfish agents. *Theor. Comp. Sci.*, 411(1):44–57, 2010.

[11] E. Even-Dar, A. Kesselman, and Y. Mansour. Convergence time to Nash equilibrium in load balancing. *ACM Trans. Algorithms*, 3(3):32, 2007.

[12] R. Feldmann, M. Gairing, T. Lücking, B. Monien, and M. Rode. Nashification and the coordination ratio for a selfish routing game. In *Proc. 30th Intl. Coll. Aut., Lang. Prog. (ICALP'03)*, pages 514–526, 2003.

[13] G. Finn and E. Horowitz. A linear time approximation algorithm for multiprocessor scheduling. *BIT Numerical Mathematics*, 19(3):312–320, 1979.

[14] D. Fotakis, S. C. Kontogiannis, E. Koutsoupias, M. Mavronicolas, and P. G. Spirakis. The structure and complexity of Nash equilibria for a selfish routing game. *Theor. Comp. Sci.*, 410(36):3305–3326, 2009.

[15] R. Graham. Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math.*, 17(2):416–429, 1969.

[16] E. Kleiman. Packing, scheduling and covering problems in a game-theoretic perspective. *CoRR*, abs/1110.6407, 2011.

[17] E. Koutsoupias and C. H. Papadimitriou. Worst-case equilibria. In *Proc.16th Ann. Symp. Theor. Aspects Comp. Sci. (STACS'99)*, pages 404–413, 1999.

[18] M. Mavronicolas and P. G. Spirakis. The price of selfish routing. *Algorithmica*, 48(1):91–126, 2007.

[19] T. Roughgarden. *Selfish routing and the price of anarchy.* MIT Press, 2005.

[20] P. Schuurman and T. Vredeveld. Performance guarantees of local search for multiprocessor scheduling. *INFORMS J. Comput.*, 19(1):52–63, 2007.