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 on 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 (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 unrelated machines. We consider use the parameter s_{max}, which is the maximum speed ratio between any two machines.

We show that on related machines, POS is unbounded for $s_{max} > 2$, and the POA is unbounded for $s_{max} \geq 2$. We study the remaining cases and show that while the POA tends to grows to infinity as s_{max} tends to 2, the POS is at most 2 for any $s_{max} \leq 2$. Finally, we analyze the POA and POS for $m = 2$.

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. For applications of this problem and related models see [7, 15].

In our scheduling model, the *coordination ratio*, or *price of anarchy (*POA*)* [17] is the worst case ratio between the social value (i.e., minimum delay of any machine, or cover) of an optimal schedule, denoted by OPT, and the value of any Nash equilibrium. 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 value of the *best* Nash equilibrium. Similarly, if both these values are 0 then we define the POS to be 1.

The non-selfish version of the problem has been well studied (known by different names such as "machine covering" and "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 POA and POS of which were extensively studied (see [15, 3, 16]), these measures were not previously considered for the uncoordinated machine covering problem in the setting of selfish jobs with respect to related machines. A different model, where machines are selfish rather than jobs with the same social goal was studied recently in [9, 5].

In [7], we considered the problem for identical machines. We showed that the POS is equal to 1 while the POA and show close bounds on the overall value of the POA is at least 1.691 and at most 1.7 for an arbitrary number of machines. For small numbers of machines, namely, 2,3 and 4 machines, the POA is $\frac{3}{2}$, $\frac{3}{2}$ and $\frac{13}{8}$, respectively.

*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. Research supported by the German Research Foundation (DFG).

In contrast to these results, we show that for uniformly related machines even the POS is unbounded already for two machines with a speed ratio *larger than* 2, and the POA is unbounded for a speed ratio of *at least* 2. The same property holds for m machines (where the speed ratio, denoted by s_{max}, or simply s) is defined to be the maximum speed ratio between any pair of machines). Surprisingly, we prove that the POS is equal to $\frac{3}{2}$ for two machines with the threshold speed ratio 2. We show that the POS is constant for m machines of speed ratio at most 2, and the POA is $\Theta(\frac{1}{2-s})$ for m machines of speed ratio $s < 2$. Additionally, we show cases where the POA and POS are unbounded. These results are very different from the situation for the makespan minimization social goal. For that problem, the POS is 1 for any speed combination. Chumaj and Vöcking [3] showed that the overall POA is $\Theta(\frac{\log m}{\log \log m})$ (see also [11, 15]).

Note that for identical machines, the results given in [7] are more similar to the situation for makespan minimization, where the POS is 1 and the POA is constant [12, 18].

The model. We now define the model of scheduling on related machines. A set of n jobs $J = \{1, 2, \ldots, n\}$ is to be assigned to a set of m machines $M = \{M_1, \ldots, M_m\}$, where machine M_i has a speed s_i. If $s_i = 1$ for $i = 1, \ldots, m$, the machines are called identical. The size of job $1 \leq k \leq n$ is denoted by p_k. An assignment or schedule is a function $\mathcal{A} : J \to M$. The load of machine M_i, which is also called the delay of this machine, is $L_i = \sum_{k:\mathcal{A}(k)=M_i} \frac{p_k}{s_i}$. The value, or the *social value* of a schedule is the minimum delay of any machine, also known as the *cover*. We denote it by $\text{COVER}(\mathcal{A})$. This problem is a dual to the makespan scheduling problem.

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 only. Preemption is not allowed. The outcome of the game is an assignment $\mathcal{A} = (\mathcal{A}_k)_{k\in N} \in \times_{k\in N} M_k$ 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^i charged from job k for running on machine M_i in a given assignment $\mathcal{A}$ is defined to be the load observed by machine i in this assignment, that is $c_k(i, \mathcal{A}_{-k}) = L_i(\mathcal{A})$, when $\mathcal{A}_{-k} \in \mathcal{S}_{-k}$; here $\mathcal{S}_{-k} = \times_{j\in N\setminus\{k\}} \mathcal{S}_j$ denotes the actions of all players except for player k.

The goal of the selfish jobs is to run on a machine with a load which is as small as possible. At an assignment that is a (pure) Nash equilibrium or NE assignment for short, there exists no machine $M_{i'}$ for which $L_{i'}(\mathcal{A}) + \frac{p_k}{s_{i'}} < L_i(\mathcal{A})$ for some job k which is assigned to machine M_i (see Figure 1(a) for an example). For this selfish goal of players, a pure Nash equilibrium (with deterministic agent choices) always exists [13, 10]. We can also consider mixed strategies, where players use probability distributions. Let t_k^i denote the probability that job $k \in N$ chooses to run on machine M_i. A strategy profile is a vector $p = (t_k^i)_{k\in N, i\in M}$ that specifies the probabilities for all jobs and all machines. Every strategy profile p induces a random schedule. The *expected load* $\mathbb{E}(L_i)$ of machine M_i in setting of mixed strategies is $\mathbb{E}(L_i) = \frac{1}{s_i} \sum_{k\in N} p_k t_k^i$. The *expected cost* of job k if assigned on machine M_i (or its *expected delay* when it is allocated to machine M_i) is $\mathbb{E}(c_k^i) = \frac{p_k}{s_i} + \sum_{j\neq k} p_j t_j^i / s_i = \mathbb{E}(L_i) + (1 - t_k^i)\frac{p_k}{s_i}$. The probabilities $(t_k^i)_{k\in N, i\in M}$ give rise to a (*mixed*) Nash equilibrium if and only if any job k will assign non-zero probabilities only to machines M_i that minimize c_k^i, that is, $t_k^i > 0$ implies $c_k^i \leq c_k^j$ for any $j \in M$. The social value of a strategy profile p is the *expected minimum load* over all machines, i.e. $\mathbb{E}(\min_{i\in M} L_i)$.

2 m Related machines

In the setting of related machines, we show that for large enough speed ratios the POA and the POS are unbounded already for two machines.

Denote the speeds of the m machines by $s_1, s_2, \ldots, s_m$, where $s_i \leq s_{i+1}$ and let $s = \frac{s_m}{s_1}$. Without loss of generality, we assume that the fastest machine has speed $s \geq 1$, and the slowest machine has speed 1.

We now characterize the situation in all cases.

Theorem 1. *On related machines with speed ratio s, $\text{POA}(s) = \infty$ for $s \geq 2$ and $\text{POS}(s) = \infty$ for $s > 2$.*

Proof. We only consider the case $s > 2$ here. Consider an instance that contains m identical sized jobs of size s. Clearly, $\text{COVER}(\text{OPT}) = 1$ for this input.

For $s > 2$, we show that any assignment where each job is assigned to a different machine is not a Nash equilibrium. In fact, in such an assignment, any job assigned to the first machine sees a load of s, while if it moves to the m-th machine, its load becomes $\frac{2s}{s} = 2 < s$. Thus, any NE assignment has a cost of 0 and the claim follows.

For $s = 2$, consider the assignment $\mathcal{A}$ where each machine is assigned a single job, 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 load. 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 another machine 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 that the job sees 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 $s = 2$. □

We next characterize the situation in all other cases.

We can in fact show that if we define $\varepsilon = 2 - s$, the POA grows with $1/\varepsilon$.

Theorem 2. *Let $\varepsilon = 2 - s$. The POA as a function of ε has the order of growth $\Theta(\frac{1}{\varepsilon})$.*

Proof. For the upper bound, consider an instance with $\text{COVER}(\text{OPT}) = 1$, and an arbitrary NE assignment $\mathcal{A}$ for this instance. We define a weight function $g(x)$ to be applied on sizes of jobs.

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

We show that the total size of jobs in any set I is at least 1 if and only if their total weight (by $g(x)$) is at least 1. In addition, if their total weight is strictly larger than 1, then $|I| \geq 2$.

Claim 3. *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 $|I| \geq 2$.*

Proof. If I has a total size of at least 1, we consider two cases. If it contains a job of size at least 1, then the total weight is at least its weight, which is 1. Otherwise, it contains only jobs of sizes less than 1, thus their total weight is equal to their total size. In both cases we get a total weight of at least 1.

If I has a total weight of at least 1, we consider two cases. If it contains a job of weight 1, then the total size is at least its size, which is 1. Otherwise, it contains only jobs of weight less than 1, thus their total size is equal to their total weight. In both cases we get a total weight of at least 1.

If $|I| = 1$, then the weight of jobs in I is at most 1. Thus if the total weight exceeds 1, then $|I| > 1$. □

Let P be the least loaded machine in $\mathcal{A}$, and let G denote the total weight of jobs assigned to P (if this set is empty then $G = 0$). Let s_p denote the speed of P.

If $G \geq 1$, 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 $G < 1$, then since every machine of 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. Therefore, the total weight of all jobs is at least m. Therefore, there exists a machine of $\mathcal{A}$ with a total weight of jobs of more than 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 at least 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, we have $\frac{p+a}{s_p} \geq \frac{q}{s_q}$ or $\frac{p}{s_p} > \frac{q}{s_q} - \frac{a}{s_p} \geq \frac{q}{2-\varepsilon} - a$, using $1 \leq s_p$ and $s_q \leq 2 - \varepsilon$.

If $a \geq \frac{1}{3}$, using $q \geq 2a$ we have $\frac{q}{2-\varepsilon} - a \geq \frac{2a-2a+a\varepsilon}{2-\varepsilon} \geq \frac{\varepsilon}{6}$. If $a < \frac{1}{3}$, we have $\frac{q}{2-\varepsilon} - a \geq \frac{1}{6}$ since $q \geq 1$. Thus the load of P is $\Omega(\frac{1}{\varepsilon})$, 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 jobs of size $s = 2 - \varepsilon$, and one job of size ε. We show that 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.

Similarly 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 $\text{COVER}(\mathcal{A}) = \varepsilon$. □

We next show that the POS has a finite value for any $s \leq 2$. For this purpose we use a well-known algorithm for scheduling, called LPT (see [14]). 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. In a case of a tie (a job can go to either machine), it assigns the job to the slowest machine which has the largest index among the slowest candidate machines.

It is known that for scheduling games on uniformly related machines with the same selfish goal of the players, LPT produces a pure NE schedule [13]. As this algorithm is deterministic, the value of POS is upper-bounded by its approximation ratio. We use this to derive an upper bound on the POS.

We first show that a ratio between the optimal cover and the cover of the schedule it creates (and accordingly the POS) is finite for $s \leq 2$.

Theorem 4. *On related machines with speed ratio s,* $\text{POS} \leq 2$ *for any $s \leq 2$.*

Proof. Suppose a counterexample exists, which shows that $\text{POS} > 2$ on m related machines that all have speeds in $[1, 2]$. Let $z > 2$ be a ratio which can be achieved by an example, i.e., the ratio between the cover of an optimal solution, and the maximum cover achieved by every Nash equilibrium, for a specific set of jobs. By normalizing, we may assume the optimal cover of the counterexample has a value of 1. Then, the total size of the jobs in it is at least m. There may be an unbounded number of (normalized) counterexamples for which the cover of the best Nash equilibrium assignment is $1/z$. Let T_z be the infimum total size of jobs in all normalized counterexamples with a best NE cover of $1/z$. Let I be such a counterexample for which the total size of all the jobs is at most $T_z + 1/(16m)$. We will derive a contradiction.

Consider the Nash equilibrium assignment $\mathcal{A}$ which is determined by the LPT assignment of the jobs in I. LPT begins by sorting the machines in order of nonincreasing speed, and gives each machine a fixed

index according to this sorting. Let P be the least loaded machine in $\mathcal{A}$. By assumption, the load of P is **exactly** $1/z < 1/2$, implying that the total size of the jobs assigned to P is less than 1 (since $s_P \leq 2$).

We herein prove a set of claims regarding qualities of assignment $\mathcal{A}$, which will lead us to a contradiction to the existence of such I.

Claim 5. *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 list of m jobs, where the size of the i-th job is denoted by p_i. We have $p_1 \geq p_2 \geq \ldots \geq p_m$. We show the claim by induction. By definition, the first job is assigned to the m-th machine. Assume that jobs $1, 2, \ldots, k$ ($k \leq m - 1$) have been assigned as claimed. Consider the 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 by definition it should be assigned to the machine of index $m - k + 1$. If the job is assigned to machine $j > m - k + 1$, the resulting load would become $\frac{p_{m-j+1}+p_k}{s_j} \geq p_k$ (since $p_{m-j+1} \geq p_k$ and $s_j \leq 2$). The load resulting from assigning the job to machine $m - k + 1$ is $\frac{p_k}{s_{m-k+1}} \leq p_k$. Thus, by definition, the job is assigned to machine $m - k + 1$, as claimed. □

We show in the next claim that the set of machines with load at least 1 is fixed until the end of LPT run after the first m jobs have been assigned.

Claim 6. *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 (Claim 5). After this and until LPT is finished, P had load of less than $1/2$, so any additional single job that P received after its first job could not by itself increase the load of P to 1 or more. This means that no assignment of any later job j increases a load of any machine above 1 once each machine has a job, because LPT would assign j to P instead. In addition, a machine which already has a load of 1 or more cannot receive this job. □

A job that is among the m largest jobs and that on assignment by LPT makes the load of its machine 1 or more is called huge. By Claim 6, the precise size of any huge job does not affect the assignment of LPT. As long as the size satisfies the following conditions, the LPT assignment is fixed:

- the size is at least the speed of the machine that it is assigned to (because then that machine receives no further jobs),
- the size is at least the size of the next job in the LPT ordering, and
- the size is at most the size of the previous job in the LPT ordering.

However, the size may of course affect the value of the optimal cover. Regarding the last two conditions, if we have two jobs of different sizes, then making those sizes equal might lead LPT to assign them in a different order. However, we ignore this and still say that the assignment is the same (i.e., we do not distinguish between jobs that have the same size). In the remainder of our proof, we will sometimes change the size of a huge job, but always in such a way that the LPT assignment (and its cover) remain unchanged, and the value of an optimal cover would not be affected. In particular, we will not increase the size of any huge job.

We furthermore show that there is a machine with load at least $1 + 1/(4m)$.

Let Q' be a machine with maximum load.

Claim 7. *Q' has load at least $1 + 1/(4m)$.*

Proof. Suppose not. Then all machines have load less than $1 + 1/(4m)$, whereas P has load less than 1. The total load on all the machines in $\mathcal{A}$ is then at most

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

On the other hand, since OPT $= 1$, the total load of all the jobs must be at least $\sum_{i=1}^{m} s_i$. This is a contradiction. □

We will return now to the proof of the main theorem.

Let Q be the machine with load at least $1 + 1/(4m)$, which has the smallest index out of such machines. Machine Q exists by Claim 7. Let A be the set of machines which received their first job before Q did, and let $A' = A \cup \{Q\}$. Let $m' = |A'|$. By Claim 6, all machines in A' that have load more than 1 have only one job. In particular, Q has a single, huge job j_Q of size

$$x \geq \left(1 + \frac{1}{4m}\right) s_Q \geq 1 + \frac{1}{4m} > 1.$$

All machines in A' have at least one job of size at least $x > 1$. All machines that are not in A' have speed at most s_Q.

We say that the jobs on machines with load at least $1 + \frac{1}{4m}$ are *threshold jobs* (they have size at least x, and are huge) while the first jobs on the other machines in A' are called *big* (their size is also at least x). Together, these are m' jobs that all have size at least x due to the definition of LPT. All big jobs were assigned before j_Q by LPT.

Consider an optimal schedule OPT. We may assume that no threshold job j is assigned to a machine of smaller index than a big job j' of the same size (j and j' can be switched if necessary).

If OPT has any threshold job j on Q or on a machine that is not in A' (and hence is not faster than Q), we claim that j can be made smaller. Let $M(j)$ be the machine that j is assigned to by LPT. First, note that j must be one of the first $m - 1$ jobs in the LPT ordering, since all machines that get a job before $M(j)$ have a job of size at least x and hence a load at least $x/2 > 1/2$. Hence, none of these machines can be P, and $M(j) \neq P$ as well. This shows that $M(j)$ must be one of the first $m - 1$ machines in the ordering used by LPT.

We now decrease p_j by $1/(8m)$. As a result, LPT may assign p_j to some later machine, but p_j is still more than 1 and j remains one of the $m - 1$ largest jobs (otherwise, there are m jobs of size more than 1, so both before and after the modification, P receives a load more than $1/2$ which is a contradiction).

If LPT still assigns j to $M(j)$, then the load of $M(j)$ remains more than 1, so $\mathcal{A}$ and $cover(\mathcal{A})$ do not change. If LPT assigns j to a later machine $M'(j)$, then the machines $M'(j), \ldots, M(j)$ all have a load at least 1 after assignment of their first job, both before and after the change. This holds because they all have a job of size at least $p_j - 1/(8m)$ (by definition of LPT) and all these machines have speed at most $s_{M(j)} \leq p_j/(1 + \frac{1}{4m}) < p_j - 1/(8m)$. Hence they receive no more jobs and the assignment of all later jobs and $cover(\mathcal{A})$ remain unchanged.

Hence, the value p_j can be decreased by at least $1/(8m)$ without affecting $cover(\mathcal{A})$. This also does not affect the value of OPT, since OPT assigns j to a machine which is not faster than Q. We have now found a counterexample with total size of jobs below T_z, a contradiction. Hence, all threshold jobs are on machines in A in the optimal assignment.

If any threshold job j is together with a big job on machine M' in the optimal assignment, it can again be made $1/(8m)$ smaller without affecting the value of OPT, since this means the load on M' is at least $2x/2 \geq 1 + \frac{1}{4m}$ on that machine (since $s_{M'} \leq 2$). (Again, the LPT assignment remains unchanged apart from possibly the order of some huge jobs, as above.)

If any big job j is on a machine with higher index than Q (i.e., not in A') in the optimal assignment, **or if j is together with another big job on machine M'**, j can be switched with j_Q (which is assigned to a machine in A by the above) without decreasing the value of OPT (since the size of j_Q is more than s_Q, but not more than that of j, **and M' has a load of at least $1 + \frac{1}{4m}$ as above**). After this, the size of j_Q can again be decreased for a contradiction.

We thus find that we may assume that all big and threshold jobs are on A' in the optimal assignment, one per machine. But then, Q must have a big job j' since it does not have a threshold job, and then j_Q and j' can again be switched in the optimal assignment (and, after this, j_Q can be made smaller as described above). We find a contradiction to the definition of I in all cases.

□

We note that unlike the POA which continuously grows from constant values to infinite values, the POS jumps from a constant value for $s = 2$ to ∞ for $s > 2$.

3 Two related machines

In this section we study the problem on two uniformly related machines. Recall that we denote the speed of the faster machine by $s \geq 1$, and the other machine has speed 1. We analyze the POA and the POS as a function of s. Thus $\text{POA}(s)$ and $\text{POS}(s)$ denote the POA and POS (respectively) of instances where the speed ratio between the machines is s.

The case where $s = 1$ was already analyzed in Section 1, and it was shown that $\text{POA}(1) = \frac{3}{2}$ and $\text{POS}(1) = 1$. Hence, from now on, we assume that $s > 1$. Moreover, Theorem 1 restricts us to the case $s < 2$ in the analysis of the POA as a function of s.

We give a complete analysis of the exact POA as a function of the speed of the faster machine, s, for $s < 2$.

Theorem 8. *For two related machines and $1 < s < 2$,*

$$\text{POA}(s) = \min\left\{\frac{s+2}{(s+1)(2-s)}, \frac{2}{s(2-s)}\right\} = \begin{cases} \frac{s+2}{(s+1)(2-s)} & \text{for } 1 < s < \sqrt{2} \\ \frac{2}{s(2-s)} & \text{for } \sqrt{2} \leq s < 2 \end{cases}$$

Proof. In order to generate the bounds on the value of the POA we use Linear Programming. First, we show that we can restrict our analysis for the upper bound to instances involving no more than 4 jobs. Thus, we need to consider a small number of cases. We then formulate for each one of these cases a linear program (LP) whose optimal objective function is exactly the POA for any s (or for some subinterval of s, depends on the case). It is possible to find the tight values using the solution for the corresponding dual linear program (DLP). Since we have several cases, we choose our bound on POA for each $1 < s < 2$ to be the highest bound accepted among all the cases. The idea for this analysis is motivated by the classic paper by Graham [14], where he used a similar approach to analyze the performance of an algorithm for a variant of the makespan minimization problem for m identical machines, and is often used since in the study of various scheduling problems. To the best of our knowledge, it was not applied to analyze the POA for any non-cooperative version of a scheduling problem before.

Consider an input instance I and an arbitrary Nash equilibrium assignment $\mathcal{A}$ for it.

Claim 9. *We can assume that I contains at most four jobs.*

Proof. If there are at least 5 jobs, then one of the machines in the optimal schedule has at least three jobs assigned to it. If a pair of jobs share a machine in both $\mathcal{A}$ and OPT (which must happen in this case), we can merge them. This is applied to any pair of jobs that behave the same in both solutions. The equilibrium properties are kept, so an alternative NE is created, which has at most four jobs, and the same ratio between the values of the optimal solution and the value of the NE solution. □

Thus we have to consider only a small number of cases. Yet, since crucial constraints in the linear programs state the fact that a given job does not have an incentive to move, it is not always possible to assume that a job, which does not exist, simply has a size of zero. Thus we need to consider the cases of one, two, three and four jobs, but some of the cases of three jobs would be seen as special cases of four jobs.

The case of one job is trivial as any schedule has a value 0, and we defined that in this case POA $= 1$. Consider a schedule with two jobs. Let A and B denote both the jobs and their sizes. A schedule where both are assigned to the same machine cannot be optimal. Moreover, such a schedule cannot be an NE either. Specifically, if both jobs are assigned to M_1, then any job has an incentive to move. If they are both assigned to M_2, then the job of size $\min\{A, B\}$ would have a load of at most $\min\{A, B\} \leq \frac{A+B}{2}$ being assigned to M_1, while the load of M_2 when both jobs are assigned to it is $\frac{A+B}{s} > \frac{A+B}{2}$.

Otherwise, COVER(OPT) > 0 and COVER($\mathcal{A}$) > 0. Thus each schedule has one job on each machine. We prove that for $s < 2$, POA $\leq s$. Suppose that the optimum schedule has job A assigned to M_1 and job B assigned to M_2. If $\mathcal{A}$ gives the same assignment we have have a ratio of 1. Else, we consider the four possibilities: If $A \leq \frac{B}{s}$ and $B \leq \frac{A}{s}$, COVER(OPT) $= A$ and NE $= B$, we have the ratio $\frac{A}{B} \leq \frac{1}{s} < 1$ for $s < 2$. If $A \leq \frac{B}{s}$ and $B > \frac{A}{s}$, COVER(OPT) $= A$ and NE $= \frac{A}{s}$, and $\frac{A}{\frac{A}{s}} \leq s$. If $A > \frac{B}{s}$ and $B \leq \frac{A}{s}$, COVER(OPT) $= \frac{B}{s}$ and NE $= B$, and the ratio is at most $\frac{\frac{B}{s}}{B} \leq \frac{1}{s} < 1$. If $A > \frac{B}{s}$ and $B > \frac{A}{s}$, COVER(OPT) $= \frac{B}{s}$ and NE $= \frac{A}{s}$, and the ratio is at most $\frac{\frac{B}{s}}{\frac{A}{s}} < s$ for $s < 2$. We conclude that the asserted upper-bound of s holds. For $s \geq \sqrt{2}$, since $\frac{2}{s(2-s)} \geq 2$, and $s < 2$, the obtained ratio s does not exceed the claimed bound. For $s < \sqrt{2}$, since $\frac{s+2}{(s+1)(2-s)} \geq \frac{3}{2}$, and $s < \frac{3}{2}$, the obtained ratio s again does not exceed the claimed bound.

We next consider the cases of three and four jobs. Whenever it is possible, we study the case of three jobs as a special case of the case of four jobs, with one of the jobs having size 0. There are four cases of three jobs, when each time a different job of the four has size 0. Also, we need two linear programs for every configuration, that is, one for the case where in $\mathcal{A}$ the slow machine is less loaded (which we call the *short* machine), and one for the opposite case. The linear programs do not state the fact that no job would benefit from leaving the short machine, which always holds.

We assume COVER(OPT) $= 1$ which can be achieved by scaling all instances.

So, suppose that the four jobs (and their sizes) are denoted by A, B, C and D. Both $\mathcal{A}$ and OPT assign exactly two jobs to each machine. Suppose also that the optimum schedule has jobs A and B assigned to M_1, and jobs C and D assigned to M_2. Suppose further that $\mathcal{A}$ assigns jobs A and C to M_1 and jobs B and D to M_2. Given these NE and optimal assignments and knowing that COVER(OPT) $= 1$, we illustrate the following linear programs, where all variables are non-negative.

The constraints first state the properties of OPT (that both machines have a load of at least 1), next the fact that the jobs assigned to the more loaded machine do not have an incentive to move, and finally, the relation between the loads of the two machines in $\mathcal{A}$. We wish to minimize the minimum load, which is the inverse of the ratio between COVER(OPT) $= 1$ and COVER($\mathcal{A}$).

Since we would like to provide an example which achieves the POA for each value of s, it is sufficient that we use only the primal linear programs and prove an upper bound on the POA, which is later matched by our examples.

The programs for four jobs are as follows.

(i) There are four jobs and the slow machine is short:

$\min A+C$ subject to: $A+B \geq 1$, $C+D \geq s$, $A+C+B \geq \frac{B+D}{s}$, $A+C+D \geq \frac{B+D}{s}$, $\frac{B+D}{s} \geq A+C$.

(ii) There are four jobs and the fast machine is short:

$\min \frac{B+D}{s}$ subject to: $A+B \geq 1$, $C+D \geq s$, $\frac{A+D+B}{s} \geq A+C$, $\frac{C+D+B}{s} \geq A+C$, $A+C \geq \frac{B+D}{s}$.

In the first program, it is assumed that $B > 0$ and $D > 0$, thus each one of the cases $B = 0$ and $D = 0$ need to be considered separately. In the second program, it is assumed that $A > 0$ and $C > 0$, thus each one of the cases $A = 0$ and $C = 0$ need to be considered separately. Thus we have additional four programs.

(iii) There are 3 jobs with non-zero sizes, $A = 0$, and the fast machine is short:

$\min \frac{B+D}{s}$ subject to: $B \geq 1$, $C+D \geq s$, $\frac{C+B+D}{s} \geq C$, $C \geq \frac{B+D}{s}$.

(iv) There are 3 jobs with non-zero sizes, $B = 0$, and the slow machine is short:

$\min A+C$ subject to: $A \geq 1$, $C+D \geq s$, $D+A+C \geq \frac{D}{s}$, $\frac{D}{s} \geq A+C$.

(v) There are 3 jobs with non-zero sizes, $C = 0$, and the fast machine is short:

$\min \frac{B+D}{s}$ subject to: $A+B \geq 1$, $D \geq s$, $\frac{A+B+D}{s} \geq A$, $A \geq \frac{B+D}{s}$.

(vi) There are 3 jobs with non-zero sizes, $D = 0$, and the slow machine is short:

$\min A+C$, subject to: $A+B \geq 1$, $C \geq s$, $B+A+C \geq \frac{B}{s}$,$\frac{B}{s} \geq A+C$.

Before we consider the first two programs, we show that the last four programs do not give a ratio larger than s.

Program (iii) gives a worst case ratio of at most s; as $B \geq 1$ and $D \geq 0$, we get that $\frac{B+D}{s} \geq \frac{1}{s} = \frac{\text{OPT}}{s}$.

Program (iv) gives a worst case ratio of at most 1; as $A \geq 1$ and $C \geq 0$, we get that $A+C \geq A \geq 1 = \text{OPT}$.

Program (v) gives a worst case ratio of at most 1; as $D \geq s$ and $B \geq 0$, we get that $\frac{B+D}{s} \geq \frac{D}{s} \geq 1 = \text{OPT}$.

Program (vi) gives a worst case ratio of at most 1; as $C \geq s$ and $A \geq 0$, we get that $A+C \geq C \geq s \geq 1 = \text{OPT}$.

We next consider the first two programs. Though it is possible to solve the programs parametrically and together with solving the corresponding parametric dual programs for all values of $1 < s < 2$. The minimum of the solutions is the POA (provided that it is not below s, the upper bound which we got for the case of two jobs).

For Program (i) with $s \leq \sqrt{2}$, we multiply the first two constraints by $2-s$, and the next two constraints by s, and sum all of the four resulting constraints. This gives $(2+s)(A+C)+2(B+D) \geq 2(B+D)+(s+1)(2-s)$, and thus $A+C \geq \frac{(s+1)(2-s)}{s+2}$. This gives an upper bound of $\frac{s+2}{(s+1)(2-s)}$.

For Program (i) with $s \geq \sqrt{2}$, we multiply the second two constraint by $2-s$, the third constraint by 1, the fourth constraint by $s-1$, the constraint $A \geq 0$ by $2-s$, and sum all of the four resulting constraints. This gives $2(A+C)+B+D \geq B+D+s(2-s)$, and thus $A+C \geq \frac{s(2-s)}{2}$. This gives an upper bound of $\frac{2}{s(2-s)}$.

For Program (ii) we multiply the first two constraints by $2s-1$, and the next two constraints by s, and sum all of the four resulting constraints. This gives $2s(A+C)+(2s+1)(B+D) \geq 2s(A+C)+(s+1)(2s-1)$, and thus $\frac{B+D}{s} \geq \frac{(s+1)(2s-1)}{s(2s+1)}$. This gives an upper bound of $\frac{s(2s+1)}{(s+1)(2s-1)}$. It is easy to verify that for $1 \leq s < 2$, $\frac{s(2s+1)}{(s+1)(2s-1)} \leq \frac{s+2}{(s+1)(2-s)}$ and $\frac{s(2s+1)}{(s+1)(2s-1)} \leq \frac{2}{s(2-s)}$ hold.

We can see from this discussion that Program (i) gives an upper bound of $\frac{s+2}{(s+1)(2-s)}$ on the POA for $1 < s < \sqrt{2}$ and gives an upper bound of $\frac{2}{s(2-s)}$ on the POA for $\sqrt{2} \leq s < 2$.

We conclude that

$$\text{POA}(s) \leq \begin{cases} \frac{s+2}{(s+1)(2-s)} & \text{, for } 1 < s < \sqrt{2} \\ \frac{2}{s(2-s)} & \text{, for } \sqrt{2} \leq s < 2 \end{cases}.$$

This bound is tight, as can be seen from the following two feasible solutions to the corresponding LPs.

For $1 < s < \sqrt{2}$, consider the following 4 jobs of sizes: $A = \frac{2-\mathcal{A}^2}{s+2}$, $B = \frac{s(s+1)}{s+2}$, $C = \frac{s}{s+2}$ and $D = \frac{s(s+1)}{s+2}$. The optimal solution has a balanced schedule of value 1. In the NE, the load of M_1 is $\frac{2+s-\mathcal{A}^2}{s+2}$, and the load of M_2 is $\frac{2(s+1)}{s+2}$. Moving a job of size $\frac{s(s+1)}{s+2}$ from M_2 to M_1 would result in a load of $\frac{2(s+1)}{s+2}$, thus this is indeed a NE. Its value is $\frac{(s+1)(2-s)}{s+2}$.

For $\sqrt{2} \leq s < 2$, consider the following 3 jobs of sizes (where the job of size A is absent): $B = \frac{\mathcal{A}^2}{2}$, $C = \frac{(2-s)s}{2}$ and $D = \frac{\mathcal{A}^2}{2}$. The optimal solution has a load of 1 on M_2 and a load of $\frac{\mathcal{A}^2}{2} \geq 1$ on M_1. In the NE, the load of M_1 is $\frac{2s-\mathcal{A}^2}{2}$, and the load of M_2 is s. Moving a job of size $\frac{\mathcal{A}^2}{2}$ from M_2 to M_1 would result in a load of s, thus this is indeed a NE. Its value is $\frac{(2-s)s}{2}$. □

We can see that as s approaches 2, POA(s) tends to infinity, as implied by Theorem 2.

Theorem 10. *For two related machines and $s \leq 2$,* POS $\leq 2 - 1/s$. *For $s \leq 3/2$,* POS $\leq 1/(2s - s^2)$. *These bounds are tight for $s \in [1, 4/3]$ and $s \in [\frac{1}{4}(3 + \sqrt{17}), 2]$. (We have $\frac{1}{4}(3 + \sqrt{17}) \approx 1.78078$).*

Proof. Consider an instance I that shows a POS of at least $2 - 1/s$ for some $s \in [1, 2]$. For this instance, the optimal solution OPT is not a Nash equilibrium. Hence, there must be at least one job in OPT that could improve its delay by moving.

Suppose there is such a job j that is alone on its machine in the optimal assignment. Then it must be on the slow machine, else it could not improve. Normalize all the job sizes so that j has size 1. The total size q of all other jobs must be less than 1, else $(1 + q)/s \geq 2/s \geq 1$, contradicting that job j can improve. Consider what happens if we switch the contents of the two machines. The cover is $\min(1/s, q)$, whereas in the optimal assignment, it was $\min(q/s, 1)$. Since $q < 1$, we have $\min(q/s, 1) = q/s$. However, the cover of the new schedule is either $1/s > q/s$ or $q > q/s$, contradicting that the original schedule is optimal. This shows that this situation cannot happen.

Therefore, any job that can improve its delay in the optimal assignment is sharing its machine with at least one other job. We consider the smallest job that can improve and denote it by j. We now normalize such that this job has size 1. Denote the delay that j experiences by D, and the delay on the other machine by D'. If j is on the fast machine, then D' must be less than $D - 1 < D - 1/s$. But then the current assignment does not give an optimal cover, since the cover is at most $D' < D - 1/s$ and $\min(D - 1/s, D' + 1)$ can be achieved by moving j.

We conclude that j must be on the slow machine. Then $D \geq 2$. Since j can improve by moving, we have

$$D' + 1/s < D.$$

We also have

$$D' \geq D - 1,$$

else the cover of the optimal assignment could be improved by moving j. Hence, after j moves, no other job can improve its delay by moving to the fast machine, since the delay on the slow machine is now only $D - 1 < D' + 1/s$. The cover of the new assignment is hence $\min(D - 1, D' + 1/s) = D - 1$, whereas

the cover of the optimal assignment was $\min(D, D') = D' < D - 1/s$, where the equality holds because j can improve. This gives us a ratio of

$$\frac{D'}{D-1} < \frac{D-1/s}{D-1} \leq \frac{2-\frac{1}{s}}{1} = 2 - \frac{1}{s}. \tag{1}$$

We have now reached a Nash equilibrium unless there are jobs on the fast machine which can improve. Any such job must have size strictly smaller than 1, since j preferred the fast machine.

We let such jobs move one by one, starting with the largest. Note that any job q which moves towards the slow machine (and thereby improves its delay) *strictly improves* the cover. The only case where this is not immediately clear is where the minimum load is achieved on the fast machine after q moves. Denote the loads before q moves by D_1 on the fast machine and D_2 on the slow machine. Then $D_2 + q < D_1$, the old cover was D_2 and the new cover is $D_1 - q/s > D_2$.

We claim that we end up in a Nash equilibrium. Suppose not. Then some job r can improve by moving back to the fast machine. Since all the jobs that moved to the slow machine have size strictly less than 1, this must be one of the jobs that moved in our process (after j moved). It cannot be the last job that moved. Consider the first time that r can improve. This happens just after another job r' moves to the slow machine which is not larger than r. But then, if r' prefers the slow machine at this point, r must do so as well, a contradiction. Since we showed that the cover only improves after job j moves, this process ends in a Nash equilibrium with a cover of at least $2 - 1/s$ due to (1).

We now improve this upper bound for $s \leq 3/2$. Consider the schedule OPT^- that we get by switching the jobs on the fast and the slow machines. The new loads are $D's$ and D/s.

The optimal cover is $\min(D', D) = D'$. The cover of OPT^- is $\min(D/s, D's)$. For $s \in [1, 3/2]$, we have $D's \geq (D-1)s > D/s$ since $D \geq 1/(s-1)$. For $s \in [3/2, 2]$, this holds because $D \geq 2$. Therefore, $\text{COVER}(\text{OPT}^-) = D/s$. This cannot be more than $\text{COVER}(\text{OPT}) = D' < D - 1/s$. We conclude that $D/s < D - 1/s$ and hence $D > 1/(s-1)$. This is a stronger condition than $D \geq 2$ for $s \leq 3/2$. Instead of (1) we now get

$$\text{POS} \leq \frac{D-1/s}{D-1} < \frac{\frac{1}{s-1} - \frac{1}{s}}{\frac{1}{s-1} - 1} = \frac{s-(s-1)}{s-s(s-1)} = \frac{1}{2s-s^2} \text{ for } s \leq 3/2.$$

Since $D/s < D's$, the only job that might be able to improve in OPT^- is a job which is now on the slow machine. Such a job currently experiences a delay of $D's$, and after moving, has a delay of at most $D/s + D'$ (the bound is reached if there is a single job on the slow machine).

Lower bounds Let $s_0 = \frac{1}{4}(3 + \sqrt{17})$. Consider some value $s_0 \leq s \leq 2$. Let M_1 be the slow machine and M_2 be the fast machine. Consider a list of 3 jobs, having sizes $2s - 1$, $1 + \varepsilon$ (for some small $\varepsilon > 0$) and 1. The jobs are listed in a non-increasing order of the sizes, as $2s - 1 > 1 + \varepsilon$ for any $s > 1 + \frac{\varepsilon}{2}$, which is the case here. The LPT algorithm applied to this list of jobs first assigns the job of size $2s - 1$ to M_2, then assigns the job of size $1 + \varepsilon$ to M_1, and as $2 + \varepsilon > \frac{2s-1+1}{s} = 2$, assigns the job of size 1 to M_2. This schedule is an NE (as any schedule produced by the LPT rule), and has a value of $1 + \varepsilon$. Obviously, LPT produces an NE schedule, but there may exist other schedules that are NE for this list of jobs. We claim that for $s \geq s_0$ this is the best NE schedule, that is, this is a schedule with the largest cover among all possible NE schedules for this instance. A different schedule, that assigns a job of size 1 to M_1 and the jobs of sizes $2s - 1$ and $1 + \varepsilon$ to M_2, is also an NE, but has a smaller cover (of 1). Clearly, no schedule that has all three jobs assigned to one of the machines and has a cover of value 0 is stable. Now consider a schedule that has

the job of size $2s-1$ assigned to M_1 and the two jobs of sizes $1+\varepsilon$ and 1 assigned to M_2. As $\frac{2+\varepsilon}{s} > 2s-1$ for any $s > 1.28$, it has a cover of $2s-1$, which is larger than the cover of the schedule produced by the LPT for s in the considered interval. This is not a NE schedule for $s \geq s_0$ though, as for these values of s we find $2s-1 > \frac{2s-1+2+\varepsilon}{s}$ holds, and the job of size $2s-1$ will benefit from moving to M_2. The optimal schedule assigns the two jobs of sizes $1+\varepsilon$ and 1 to M_1, and the job of size $2s-1$ to M_2, and has a cover of $\frac{2s-1}{s} = 2-\frac{1}{s}$ and $(2-\frac{1}{s} < 2$, for positive s). It is not stable, though, as the job of size 1, for example, will benefit from moving to M_2. We conclude that $\text{POS}(s) \geq 2-\frac{1}{s}$ for $s \in [s_0, 2]$ (letting ε tend to 0).

Now consider the case $1 \leq s \leq 4/3$. We now use an instance with four jobs, two of size $2-s+\varepsilon$ and two of size $s-1$ (note that $2(s-1) < 2-s+\varepsilon$). The optimal cover is $(1+\varepsilon)/s$ (two jobs on each machine). Consider the possible allocations of the jobs. If there is a machine with one job, then the cover is at most $2-s+\varepsilon$ for a bound that tends to $1/(2s-s^2)$ as desired. If the two small jobs are on one machine, the cover is at most $2(s-1)$, which is even less. If each machine has two different jobs, we do not have a Nash equilibrium, because the small job on the slow machine can improve by moving. Its current delay is $1+\varepsilon$, and the delay on the fast machine would be only $(2(s-1)+2-s+\varepsilon)/s = 1+\varepsilon/s$. Hence in all cases, either the cover is at most $2-s+\varepsilon$ or the schedule is not a Nash equilibrium. □

It is interesting to note that while the POS is equal to $\frac{3}{2}$ for $s=2$, it becomes unbounded already for $s = 2+\varepsilon$ for arbitrary small positive ε.

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 Journal on Computing*, 38(4):1602–1623, 2008.

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

[3] A. Czumaj and B. Vöcking. Tight bounds for worst-case equilibria. *ACM Transactions on Algorithms*, 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 Journal on Discrete Mathematics*, 3(2):190–196, 1982.

[5] P. Dhangwatnotai, S. Dobzinski, S. Dughmi, and T. Roughgarden. Truthful approximation schemes for single-parameter agents. In *49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 15–24, 2008.

[6] T. Ebenlendr, J. Noga, J. Sgall, and G. J. Woeginger. A note on semi-online machine covering. In *Approximation and Online Algorithms, Third International 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. of the 5th International Workshop on Internet and Network Economics (WINE'09)*, pages 232–243, 2009.

[8] L. Epstein, A. Levin, and R. van Stee. Max-min online allocations with a reordering buffer. In *Proc. of the 37th International Colloquium on Automata, Languages and Programming (ICALP'10), Part 1*, pages 336–347, 2010.

[9] L. Epstein and R. van Stee. Maximizing the minimum load for selfish agents. *Theoretical Computer Sciecne*, 411(1):44–57, 2010.

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

[11] R. Feldmann, M. Gairing, T. Lücking, B. Monien, and M. Rode. Nashification and the coordination ratio for a selfish routing game. In *Proc. of the 30th International Colloquium on Automata, Languages and Programming (ICALP'03)*, pages 514–526, 2003.

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

[13] 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. *Theoretical Computer Science*, 410(36):3305–3326, 2009.

[14] R. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal of Applied Mathematics*, 17(2):416–429, 1969.

[15] E. Koutsoupias and C. H. Papadimitriou. Worst-case equilibria. In *Proc. of the 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS'99)*, pages 404–413, 1999.

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

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

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