Maximizing the Minimum Load: The Cost of Selfishness*

Xujin Chen† Leah Epstein‡ Elena Kleiman§ Rob van Stee¶

Abstract

We consider a scheduling problem on m machines, where each job is controlled by a selfish agent. Each agent is only interested in minimizing its own cost, defined as the total load of the machine that its job is assigned to. We consider the objective of maximizing the minimum load (the value of the cover) over the machines. Unlike the regular makespan minimization problem, which was extensively studied in a game theoretic context, this problem has not been considered in this setting before.

We study the price of anarchy (POA) and the price of stability (POS). These measures are unbounded already for two uniformly related machines [11], and therefore we focus on identical machines. We show that the POS is 1, and derive tight bounds on the pure POA for $m \leq 7$ and on the overall pure POA, showing that its value is exactly 1.7. To achieve the upper bound of 1.7, we make an unusual use of weighting functions. Finally, we show that the mixed POA grows exponentially with m for this problem.

In addition, we consider a similar setting of selfish jobs with a different objective of minimizing the maximum ratio between the loads of any pair of machines in the schedule. We show that under this objective the POS is 1 and the pure POA is 2, for any $m \geq 2$.

keywords: scheduling; price of anarchy; machine covering; envy-ratio.

1 Introduction

Classical optimization problems, and network optimization problems in particular, are often modelled as non-cooperative strategic games. Many solution concepts are used to study the behavior of selfish agents in non-cooperative games. Probably the best known concept is that of the Nash equilibrium (NE) [24]. This is a state which is stable in the sense that no agent can gain from switching strategies unilaterally. Following recent interest of computer scientists in game theoretical concepts with respect to scheduling and routing problems [25, 21, 27], we study Nash equilibria for a scheduling problem on identical machines where the goal is maximizing the minimum load. The novelty of our study compared to other work in the area is that the social goal is very different from the private goals of the players.

A set of n jobs $J = \{1, 2, \ldots, n\}$ is to be assigned to a set of m identical machines $M = \{1, \ldots, m\}$, such that each job is assigned to one of the machines. The size of job k (for $1 \leq k \leq n$) is denoted by p_k. An assignment or a 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 $L_i(\mathcal{A}) = \sum_{k:\mathcal{A}_k=i} p_k$. We let $L_{min}(\mathcal{A}) = \min_{1\leq i\leq m} L_i(\mathcal{A})$ and $L_{max}(\mathcal{A}) = \max_{1\leq i\leq m} L_i(\mathcal{A})$. We sometimes use L_i, L_{min}, and L_{max}, if the schedule $\mathcal{A}$ is clear from the context. The value (or the *social*

*A preliminary version of this paper appeared as [10].

†Institute of Applied Mathematics, Chinese Academy of Sciences, Beijing 100190, PR China. xchen@amss.cas.cn. Research supported in part by the National Science Foundation of China under grant 11222109.

‡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).

value) of a schedule $\mathcal{A}$ is the minimum delay of any machine, $L_{min}(\mathcal{A})$, also known as the value of the *cover*. We sometimes denote it by $\text{COVER}(\mathcal{A})$. We study the problem of maximizing the value of the cover [6]. This problem is dual to the makespan scheduling problem [18], where the goal is to minimize $L_{max}(\mathcal{A})$.

The problem of maximizing the minimum load, considering jobs as selfish agents, can be modelled as a routing problem. In this setting, machines are associated with parallel links between a source and a destination. The links have bounded capacities, and a set of users request to send certain amounts of unsplittable flow between the two nodes. Requests are to be assigned to links and consume bandwidth that depends on their sizes. The cost charged to a user for using a link equals to the total amount of the utilized bandwidth of that link. Thus, the selfish users prefer to route their traffic on a link with small load. This scenario is similar to the model proposed by Koutsoupias and Papadimitriou [21, 17, 14], but our model has a different social goal function. Other motivations come from issues of Quality of Service, fair resource allocation, and fair queuing.

A machine covering game is characterized by a set of atomic players N such that each player controls a single job 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 for each job is the set M of all machines. Each job must be assigned to one machine. The outcome of the game is a schedule or an assignment $\mathcal{A} = (\mathcal{A}_k)_{k\in N} \in \times_{k\in N} M$ 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 $c_k(\mathcal{A}) : \mathcal{S} \to \mathbb{R}^+$ (abbreviated by c_k). 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 that 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 does not exist a machine $i' \neq i$ for which $L_{i'}(\mathcal{A}) + p_k < L_i(\mathcal{A})$. 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 [17, 14]. In these articles the private goals of players are defined as here, while a different social goal is studied. The social goal is irrelevant to the property of a schedule being an NE since only the private goals have a role in the definition, so the set of NE schedules is the same, no matter whether the social goal is to minimize the makespan or to maximize the minimum load.

We can also consider mixed strategies, where players use probability distributions over possible strategies. Let $t_k^i \geq 0$ denote the probability that job $k \in N$ chooses to run on machine i (such that $\sum_{i=1}^m t_k^i = 1$). A strategy profile is a matrix $\mathcal{P} = (t_k^i)_{k\in N, i\in M}$ that specifies the probabilities for all jobs and all machines. Every strategy profile $\mathcal{P}$ induces a random schedule. The *expected load* $\mathbb{E}(L_i)$ of machine i in the setting of mixed strategies is $\mathbb{E}(L_i) = \sum_{k\in N} p_k t_k^i$, that is, job k adds to the load of machine i its size times the probability that it chooses this machine. The *expected cost* of job k if assigned to machine i (or its *expected delay* when it is allocated to machine i) is $\mathbb{E}(c_k^i) = p_k + \sum_{j\neq k} p_j t_j^i = \mathbb{E}(L_i) + (1 - t_k^i)p_k$. Here the expected load is computed under the assumption that job k chooses machine i (with probability 1), while the other jobs choose their machines according to the probability distribution. 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 cannot strictly decrease its cost by changing its probability distribution unilaterally, and it assigns non-zero probabilities only to machines i that minimize $\mathbb{E}(c_k^i)$, that is, $t_k^i > 0$ implies $\mathbb{E}(c_k^i) \leq \mathbb{E}(c_k^\ell)$ for any $\ell \in M$. This is equivalent to saying that a job assigns positive probabilities only to machines where the expected load resulting from other jobs is minimum. The social value of a strategy profile $\mathcal{P}$ is the *expected minimum load* over all machines, i.e. $\mathbb{E}(\min_{i\in M} L_i)$.

The sets of optimal solutions for the two goal functions, maximization of the minimum load, and makespan minimization, may be different. An example given in [1], with $m = 3$, $p_1 = 13$, $p_2 = p_3 = 9$, and $p_4 = p_5 = p_6 = 6$, has disjoint sets of optimal solutions for the two problems (see Figure 1 for an illustration). However, the set of NE schedules is independent of the social goal as explained above, and therefore this set is common to the two goal functions. To demonstrate the non-triviality of the problem of

maximizing the minimum load with respect to Nash equilibria, consider the following example (see Figure 2 for an illustration). Let $m = 3$, $p_1 = p_2 = p_3 = 8$, $p_4 = p_5 = p_6 = 4$, and $p_7 = p_8 = 1$. One optimal solution assigns one job of size 8 and one job of size 4 to each machine, and machine 1 has also the last two unit sized jobs. The social value of this assignment is 12, but this solution is not an NE (since moving job 7 or job 8 to another machine would result in a smaller cost for the moved job). On the other hand, assigning the subsets of jobs $\{1, 2\}$, $\{4, 5, 6\}$, $\{3, 7, 8\}$ to the three machines results in an NE, but its social value is only 10. We will show, however, that the two sets, consisting of optimal solutions and of NE schedules (for the same instance), are never disjoint, similarly to the situation for the makespan minimization problem [17, 14]. In particular, for cases where the sets of optimal solutions for the two social goals are disjoint, there must exist at least two distinct NE schedules.

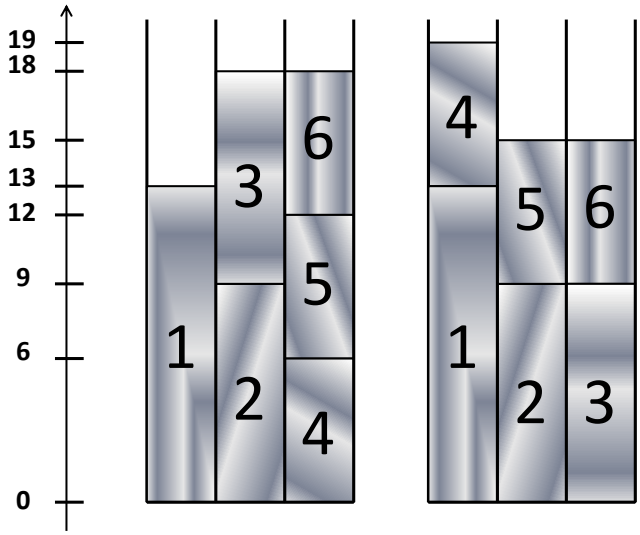

Figure 1: The example of [1]. The unique optimal solution for makespan minimization is given on the left hand side, and the unique optimal solution for maximizing the minimum load (up to swapping jobs of equal size) is given on the right hand side.

For a game in this class, the *coordination ratio*, or *price of anarchy (*POA*)* [26] 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*)* [2] is the worst case ratio between the social value of an optimal solution, and the social value of the *best* NE. Again, 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 $\text{POA}(m)$ and $\text{POS}(m)$ denote the POA and POS, respectively, for the class of games with m identical machines. We let POA and POS denote the overall values for all possible games, that is, $\text{POA} = \sup_m \text{POA}(m)$ and $\text{POS} = \sup_m \text{POS}(m)$. Note that $\text{POS}(1) = \text{POA}(1) = 1$, as in the case $m = 1$ there is one schedule for every set of jobs (where this schedule is socially optimal and an NE).

In addition, we study the *mixed* POA (MPOA), where we consider mixed Nash equilibria that result from mixed strategies, where the players' choices are not deterministic and are regulated by probability distributions on a set M of pure strategies. A mixed Nash equilibrium (defined above) is characterized by the property that there is no incentive for any job to deviate from its probability distribution (a deviation is

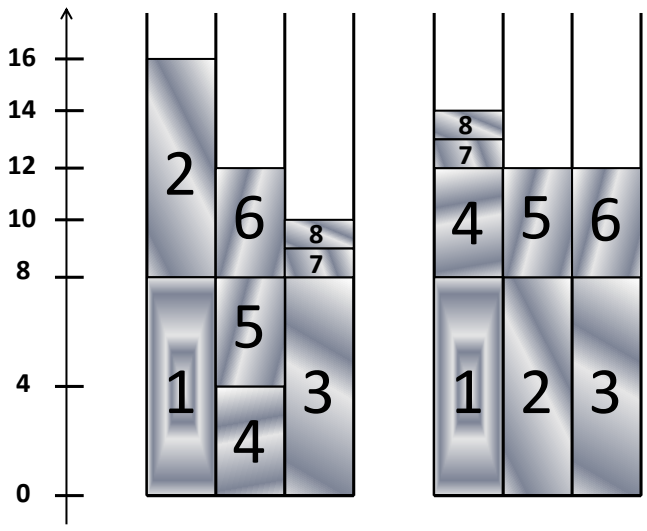

Figure 2: An exa,ple. An NE that is not a socially optimal solution is given on the left hand side, and an optimal solution for the same input that is not an NE is given on the right hand side.

any modification of its probability vector over machines), while probability distributions of other players remain unchanged. The existence of such an equilibrium over mixed strategies for non-cooperative games was shown by Nash in his famous work [24]. The values MPOA and MPOS are defined similarly to the pure ones, but mixed Nash equilibria are being considered instead of pure ones. Clearly, any pure NE is a mixed NE. In this case we use MPOA to denote overall value for all machine covering games, and MPOA(m) for games with m machines (and analogously, we use MPOS and MPOS(m)).

Our results and related work. 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. [6, 3, 9]). Various game-theoretic aspects of max-min fairness in resource allocation games were considered before this paper (e.g. in [28]), but unlike the makespan minimization problem for which the POA and POS were extensively studied (see [21, 5, 23]), these measures were not previously considered for the uncoordinated machine covering problem in the setting of selfish jobs. A different model, where machines are selfish rather than jobs (with the same social goal function) was studied recently in [13, 7, 12].

For identical machines, we show that POS$(m) = 1$ for every $m \geq 2$ in Section 3. Hence POS $=$ MPOS $= 1$. As our main result, we study the POA and show the tight bound of 1.7 on its overall value. This is in contrast with the makespan minimization social goal, where it is known that the POA for m identical machines is $\frac{2m}{m+1}$, giving an overall bound of 2 [16, 29].

For the analysis of our upper bound we use the weighting function technique, which is uncommon in scheduling problems. Moreover, we use not only the weight function but also its inverse function in our analysis. Surprisingly, this bound is the approximation ratio of the well-known algorithm for bin packing, First-Fit [19] (see also [8]). We furthermore prove that the POA is monotonically non-decreasing as a function of m. For small numbers of machines we provide the exact values of POA: we find that POA$(2) =$ POA$(3) = 3/2$, POA$(4) =$ POA$(5) = 13/8 = 1.625$, and POA$(6) =$ POA$(7) = \frac{5}{3}$. As for the MPOA, we show that its value is very large as a function of m, and MPOA$(2) = 2$. Note that for the makespan minimization social goal MPOA$(m) = \Theta(\frac{\log m}{\log \log m})$ and MPOA$(2) = \frac{3}{2}$ [21, 5]. The POA is

analyzed in Section 2, and the MPOA in Section 4.

In this paper, we focus on identical machines, which is an important special case of uniformly related machines. It is shown in [11][1] that for this more general variant, even the POS is unbounded already for any number of machines $m \geq 2$ with a maximum speed ratio *larger than* 2, and the POA is unbounded for a speed ratio of *at least* 2. This is very different from the results in the situation of the makespan minimization social goal, where the POA is finite [5, 15]. A study of the cases where the speed ratio is no larger than 2 can be found in [11, 20, 30].

In Section 5, we present a different model where the social goal is to minimize the ratio between the load of the most loaded machine and the least loaded machine, and we show tight bounds on the POS and POA for any number of identical machines. Similarly to the results of [11], one can argue that for any number of uniformly related machines these measures are unbounded.

2 The POA of the machine covering game

An example given in the introduction demonstrates that not every NE schedule is optimal. In what follows we measure the extent of deterioration in the quality of NE schedules due to the effect of selfish and uncoordinated behavior of the players (jobs), in the worst case. As described in the introduction, the measures we use are the POA and the POS. In this section, we analyze the POA.

Throughout this section, we consider assignments of jobs to machines whose value of cover is exactly 1. Every schedule whose value of cover is positive can be scaled to satisfy this requirement. We neglect other schedules (where the value of cover is zero) since in such a schedule there is either a machine which has at least two jobs, in which case the schedule is not an NE (since another machine has no jobs), or $n < m$. In the latter case, such a schedule is socially optimal. Thus, neither of these cases is of interest for the study of the POA. We let $\mathcal{A}$ denote an assignment for a given instance of the machine covering game (and we assume $\text{COVER}(\mathcal{A}) = 1$).

Before we proceed, we prove two useful observations.

Observation 1. *$\mathcal{A}$ is an* NE *if and only if for every job the total size of all other jobs assigned to the same machine is at most* 1.

Proof. Assume that for every job k, the total size of jobs assigned with it is at most 1. All other machines have loads of at least 1, and thus moving to another machine is not beneficial for k. Therefore, a schedule satisfying this property for every job is an NE.

Assume now that the schedule is an NE, and assume by contradiction that a job k does not satisfy this property. The machine of k is not the least loaded machine, as there is a machine of load exactly 1. Thus, k would benefit from moving to the least loaded machine, contradicting the property that $\mathcal{A}$ is an NE schedule. □

Observation 2. *The function* $\text{POA}(m)$ *is monotonically non-decreasing as a function of the number of machines* m.

Proof. Assume that $\mathcal{A}$ is an NE schedule for a set of jobs $J = \{1, 2, \ldots, n\}$ and $m-1$ machines (such that $\text{COVER}(\mathcal{A}) = 1$). Let $C' \geq 1$ be the optimal cover value for J and $m-1$ machines. Let $J' = J \cup \{n+1\}$, where $p_{n+1} = C'$. Let $\mathcal{A}'$ be the schedule that is identical to $\mathcal{A}$ for $m-1$ machines, and job $n+1$ is assigned to machine m. Similarly, modify an optimal schedule by adding a machine to which this job is assigned. The value of cover of $\mathcal{A}'$ is exactly 1, and the value of the cover of an optimal schedule for J' is at least C'. By Observation 1 $\mathcal{A}'$ is an NE assignment. □

[1] See also the conference version [10], where this result was claimed.

In what follows, in every upper bound proof we consider a specific NE schedule $\mathcal{A}$ for m machines. We denote a machine which is loaded by 1 in $\mathcal{A}$ by P. All other machines are called *tall* machines. We partition the tall machines into three classes. Tall machines running a single job in $\mathcal{A}$ are called singleton machines. Tall machines running two jobs in $\mathcal{A}$ are called paired machines, and other tall machines are called regular machines. The jobs assigned to singleton, paired, and regular machines are called singleton, paired, and regular jobs, respectively.

Let C be the value of the optimal cover for this instance. Out of all instances with m machines, and an optimal cover of value at least C, we assume without loss of generality that the instance of $\mathcal{A}$ has a maximum number of jobs of size 1; by Observation 1, the number of such jobs does not exceed $2m$ (in fact, it does not exceed $2m-1$, since a machine of load 1 in $\mathcal{A}$, which must exist, can only contain one such job), and thus $\mathcal{A}$ is well-defined. Let W denote the total size of jobs in this instance. Let x denote the number of paired machines in $\mathcal{A}$, let y denote the number of singleton machines in $\mathcal{A}$, and let z denote the number of regular machines in $\mathcal{A}$. Including P, we have in total $x+y+z+1=m$ machines.

Observation 3. *$C \geq 1$ and $W \geq mC$. All paired jobs have size 1.*

Proof. The claims regarding C are obvious. If there exists at least one paired job whose size is not 1, then by Observation 1, its size is below 1. Replace all such jobs with jobs of size 1 (without changing their machines in $\mathcal{A}$), and let $\mathcal{A}'$ be the resulting schedule. The value of the cover of an optimal solution for the new instance is at least C. By Observation 1, the schedule $\mathcal{A}'$ is an NE schedule. This contradicts the choice of $\mathcal{A}$ as an NE schedule for m machines with a maximum number of jobs of size 1. □

Lemma 4. *Let k be a job assigned to a regular machine in $\mathcal{A}$, and let T be the load of its machine in $\mathcal{A}$. We have $T \leq 1+p_k$, $T \leq 2-p_k$, and $T \leq 1.5$.*

Proof. By Observation 1, the total size of all jobs except for k is at most 1, i.e., $T \leq 1+p_k$. Let k_1 and k_2 be two other jobs assigned to the same machine in $\mathcal{A}$. Using $T \leq 1+p_{k_1}$ and $T \leq 1+p_{k_2}$, together with $T \geq p_k+p_{k_1}+p_{k_2}$, we find $2T \leq 2+p_{k_1}+p_{k_2} \leq 2+T-p_k$, which implies $T \leq 2-p_k$. The sum of the two inequalities $T \leq 1+p_k$ and $T \leq 2-p_k$ implies $T \leq 1.5$. □

Observation 5. *If $y \geq 1$, then there exists a schedule $\mathcal{A}'$ for $m-1$ machines that is an* NE *schedule, such that the value of the optimal cover for it is at least C. Thus $C \leq \text{POA}(m-1)$.*

Proof. We construct an instance for $m-1$ machines, for which the value of cover in an optimal schedule is at least C, and an NE schedule $\mathcal{A}'$ for which the value of cover is 1.

Remove a singleton job j_k and the singleton machine running j_k in $\mathcal{A}$. By Observation 1, the resulting schedule $\mathcal{A}'$ is an NE. The value of the cover remains 1 as only a tall machine was removed. To create a schedule whose value of cover is at least C, consider an optimal schedule that uses m machines for the original number of jobs. Remove the machine running j_k. If this machine has additional jobs, move them to other machines. Let $C' \geq C$ be the resulting value of cover. We find $C' \leq \text{POA}(m-1)$ and thus $C \leq \text{POA}(m-1)$. □

Our analysis of the upper bound on the POA for small values of m strongly relies on the following two observations, which hold for any m.

Observation 6. *The total size of jobs assigned in $\mathcal{A}$ to a regular machine that has $t \geq 3$ jobs assigned to is at most $\frac{t}{t-1}$, which is a decreasing function of t. The total size of jobs assigned to a paired machine is 2.*

Proof. By Observation 1, every $t-1$ of these jobs have a size of at most 1. Summing this up over all t subsets of $t-1$ jobs produces the first inequality claimed, as every job participates in $t-1$ such inequalities. The second part follows from Observation 3. □

Observation 7. *Assume that there are no singleton machines. Then* $C \leq \frac{4x+3z+2}{2m} = \frac{3m-1+x}{2m} \leq \frac{2m-1}{m} = 2 - \frac{1}{m}$.

Proof. From Observations 6 and 3 we get $mC \leq W \leq 2x + \frac{3}{2}z + 1$, which gives $C \leq \frac{4x+3z+2}{2m} = \frac{3m-1+x}{2m} \leq \frac{2m-1}{m}$, where the equality holds since $3(x+z) = 3m-3$, and the second inequality holds since $x \leq m-1$. □

The next theorem gives a simple upper bound of 1.75 on the overall POA. It is used to prove upper bounds on $\text{POA}(m)$ for small values of m.

Theorem 8. *For even values of* $m \geq 2$, $\text{POA}(m) \leq \frac{7}{4} - \frac{1}{2m}$. *For odd values of* $m \geq 3$, $\text{POA}(m) \leq \frac{7}{4} - \frac{1}{2(m-1)}$. *Thus,* $\text{POA} \leq 1.75$.

Proof. We prove the claim by induction on m. For $m = 2$ the claim is $\text{POA}(2) \leq \frac{3}{2}$. If there are no singleton machines, then the claim follows from Observation 7. Otherwise, it follows from Observation 5, as $\text{POA}(1) = 1$.

For the inductive step, we can only consider cases where there are no singleton machines (otherwise the inductive hypothesis can be used).

Consider first the case that there are at most $\lfloor \frac{m}{2} \rfloor$ paired machines. Thus, the number of regular machines is at least $\lceil \frac{m}{2} \rceil - 1$ (i.e., $x \leq \frac{m}{2}$ and $z \geq \frac{m}{2} - 1$ if m is even, and $x \leq \frac{m-1}{2}$ and $z \geq \frac{m-1}{2}$ if m is odd). Using Observation 7 we find $C \leq \frac{3m-1+\lfloor \frac{m}{2} \rfloor}{2m}$, which gives $C \leq \frac{7}{4} - \frac{1}{2m}$ for even m, and $C \leq \frac{3m-1+\frac{m-1}{2}}{2m} = \frac{7}{4} - \frac{3}{4m} \leq \frac{7}{4} - \frac{1}{2(m-1)}$ for odd $m \geq 3$.

Next, suppose that $x > \lfloor \frac{m}{2} \rfloor$, i.e., $x > \frac{m}{2}$. Consider a socially optimal schedule $\mathcal{O}$ where the number of machines having at least three paired jobs is minimal. We show that no machine has more than two paired jobs in $\mathcal{O}$. Assume that there exists a machine having $q \geq 3$ such jobs in $\mathcal{O}$. There are at most $m-1$ paired machines in $\mathcal{A}$, and thus there are at most $2(m-1)$ paired jobs. We show that we can move $q-2$ jobs from this machine of $\mathcal{O}$ to other machines such that no machine will have more than two paired jobs. The value of the cover of the resulting schedule cannot be larger than the cover value of the original one, as this would contradict optimality. For every machine whose load has decreased, its new load is at least 2 (by Observation 3), and since $C < 2$ (by Observation 7), its new load is at least C. Therefore, the value of the cover of the resulting schedule is still exactly C. Thus, if moving these jobs can be done, then we can create an alternative optimal schedule where the number of machines having more than two paired jobs is smaller, contradicting the choice of $\mathcal{O}$.

To satisfy the condition that no machine will have more than two paired jobs, it is possible to move at most two jobs to each machine of $\mathcal{O}$ that has no paired jobs, and at most one job to a machine which already has one paired job. Let γ_0 and γ_1 be the numbers of such machines in $\mathcal{O}$ (having no paired jobs, and one paired job, respectively). The number of paired jobs is thus at least $\gamma_1 + 2(m - \gamma_1 - \gamma_0 - 1) + q = 2m - 2\gamma_0 - \gamma_1 - 2 + q$, and since it is at most $2(m-1)$, we find $2\gamma_0 + \gamma_1 \geq q > q - 2$, and all $q-2$ jobs can be moved appropriately.

Let $\rho = 2x$ be the number of paired jobs (so $m < \rho \leq 2(m-1)$). In the optimal schedule $\mathcal{O}$ there are at least $\rho - m$ machines with exactly two paired jobs assigned to each of them (as otherwise some machine would have more than two such jobs). Therefore, in $\mathcal{O}$ there are at least $\rho - m$ machines of load at least 2, and we find $W \geq 2(\rho - m) + C(2m - \rho)$. By Lemma 4 we also have $W \leq \frac{3}{2}(m-1) + \frac{x}{2} + 1 = \frac{3}{2}m + \frac{\rho}{4} - \frac{1}{2}$. Combining the two inequalities we find $\frac{3}{2}m + \frac{\rho}{4} - \frac{1}{2} - 2(\rho - m) \geq C(2m - \rho)$, or alternatively, $C(2m-\rho) \leq \frac{7}{4}(2m-\rho) - \frac{1}{2}$. Finally, using $\rho \geq m+1$ we get $C \leq \frac{7}{4} - \frac{1}{2(m-1)}$. □

Before we present the proof of our bounds on the POA for general m, to get the notion of the problem, we prove tight bounds on the POA for simple cases where $2 \leq m \leq 7$.

Theorem 9. $\text{POA}(2) = \text{POA}(3) = \frac{3}{2}$, $\text{POA}(4) = \text{POA}(5) = \frac{13}{8}$, $\text{POA}(6) = \text{POA}(7) = \frac{5}{3}$.

Proof. The upper bounds follow from Theorem 8. By Observation 2, we only need to prove lower bounds for $m = 2, 4, 6$. The lower bounds follow from the following instances. For $m = 2$, there are two jobs of size 1 and two jobs of size $\frac{1}{2}$. By assigning one job of each size to each machine, we can create a schedule whose value of cover of is $\frac{3}{2}$. An assignment where one machine has the two larger jobs and the other machine has the two smaller jobs is an NE by Observation 1, and its value of cover is 1. This shows $\text{POA}(2) \geq \frac{3}{2}$. For $m = 4$, there are four jobs of each of the sizes 1, $\frac{1}{2}$, $\frac{1}{8}$. By assigning one job of each size to each machine, we can create a schedule whose value of cover of is $\frac{13}{8}$. An assignment where two machines have two jobs of size 1 each, one machine has three jobs of size $\frac{1}{2}$, and another machine has the remaining jobs, is an NE by Observation 1, and its value of cover is 1. For $m = 6$, there are six jobs of each of the sizes 1, $\frac{1}{2}$, $\frac{1}{6}$. By assigning one job of each size to each machine, we can create a schedule whose value of cover of is $\frac{5}{3}$. An assignment where three machines have two jobs of size 1 each, two machines have three jobs of size $\frac{1}{2}$, and another machine has the six remaining jobs (each of which has size $\frac{1}{6}$) is an NE by Observation 1, and its value of cover is 1. □

2.1 The overall POA

In this section, we improve the upper bound above, and prove that $\text{POA} = 1.7$. Recall that we consider an NE assignment $\mathcal{A}$ for which the value of the optimal cover is C. Additionally to the prior assumptions, we also assume that the analyzed assignment is minimal with respect to the number of machines (among assignments for which $\text{COVER}(\text{OPT}) \geq C$). Thus, no machine in $\mathcal{A}$ is a singleton machine. This can be assumed due to Observation 5.

Claim 10. *No job has a size larger than* 1.

Proof. This follows from the fact that there is no singleton machine in $\mathcal{A}$, and thus the condition of Observation 1 holds for any proper subset of jobs assigned to a tall machine. Jobs assigned to P obviously have sizes no larger than 1. □

We define a weight function $w(x)$ on sizes of jobs.

$$w(x) = \begin{cases} \frac{1}{2} & , \quad \text{for} \quad x \in [\frac{2}{3}, 1] \\ \frac{x}{2-x} & , \quad \text{for} \quad x \in (\frac{1}{2}, \frac{2}{3}) \\ \frac{x}{x+1} & , \quad \text{for} \quad x \in (0, \frac{1}{2}] \end{cases}$$

The motivation for the weight function is to define the weight of a job j to be at least the fraction of its size out of the total size of jobs assigned to the same machine in $\mathcal{A}$.

Claim 11. *The function w is continuously non-decreasing. For every $x \in (0, 1]$, $w(x) < x$. For every $x \in (0, \frac{2}{3})$, $w(x) < \frac{1}{2}$. For every $x \in (0, \frac{2}{3}]$, $w(x) \geq \frac{2x}{3}$. For a job of size x assigned to a tall machine of load T in $\mathcal{A}$, $w(x) \leq \frac{x}{T}$.*

Proof. It is not difficult to see that the function is piecewise non-decreasing, and continuous at breakpoints. Thus $w(x) \leq \frac{1}{2}$, and if $x < \frac{2}{3}$, then $w(x) < \frac{1}{2}$. For $x < 1$, $2 - x > 1$, and for $x > 0$, $x + 1 > 1$. Therefore, $w(x) < x$. If $x \in (0, \frac{1}{2}]$, then $\frac{w(x)}{x} = \frac{1}{x+1} \geq \frac{2}{3}$, as $x \leq \frac{1}{2}$. If $x \in (\frac{1}{2}, \frac{2}{3}]$, then $\frac{w(x)}{x} = \frac{1}{2-x} \geq \frac{2}{3}$, as $x \geq \frac{1}{2}$.

Consider the last claim, and a job j of size x assigned to a tall machine. If j is assigned to a paired machine in $\mathcal{A}$, then by Observation 3, $x = 1$, $w(x) = \frac{1}{2}$, $T = 2$, and we have $w(x) = \frac{x}{T}$. Otherwise, j is assigned to a regular machine i. By Lemma 4, the total size of jobs assigned to i in $\mathcal{A}$ is no larger than $\min\{1 + x, 2 - x\}$. Thus the claim holds if $x < \frac{2}{3}$. Otherwise, using $x \geq \frac{2}{3}$ we have $T \leq 2 - x \leq 2x$, and $w(x) = \frac{1}{2} = \frac{x}{2x} \leq \frac{x}{T}$. □

Claim 12. *The total weight of jobs scheduled on P in $\mathcal{A}$ is less than* 1*. The total weight of jobs scheduled on a tall machine in $\mathcal{A}$ is at most* 1*.*

Proof. The property for P follows from the fact that P has load 1 and $w(x) < x$. For a tall machine, where the total size of jobs is T, the total weight is at most 1, as $w(x) \leq \frac{x}{T}$ for all x. □

Claim 13. *There is a machine in the optimal assignment whose total weight is strictly smaller than* 1*.*

Proof. The total weight of all jobs is less than m by Claim 12. □

The inverse function, $f(y)$, of the weight function, $w(x)$, is defined for $y \in (0, \frac{1}{2})$ by

$$f(y) = \begin{cases} \frac{2y}{y+1} & , \quad \text{for} \quad y \in (\frac{1}{3}, \frac{1}{2}) \\ \frac{y}{1-y} & , \quad \text{for} \quad y \in (0, \frac{1}{3}] \end{cases}$$

Note that $f(y)$ is continuous at $\frac{1}{3}$, and that it is monotonically increasing in $(0, \frac{1}{2})$. We do not define f for $y = \frac{1}{2}$, since $w(x) = \frac{1}{2}$ for all $x \in [\frac{2}{3}, 1]$.

Claim 14. *The total size of any set of jobs with total weight below* 1 *is at most* 1.7*.*

Proof. Consider a set of jobs I of a total weight strictly below 1. If there is no job of size at least $\frac{2}{3}$ in I, then as $\frac{x}{w(x)} \leq \frac{3}{2}$, the total size of the jobs in I does not exceed $\frac{3}{2}$. Otherwise, recall that for any x, $w(x) \leq \frac{1}{2}$, thus there is exactly one job of weight $\frac{1}{2}$, and its size is no larger than 1. It is therefore sufficient to show that the total size of any set of jobs I' which has total weight below $\frac{1}{2}$ is at most 0.7, that is, if there is a set of numbers $\mu_1 \geq \cdots \geq \mu_\ell > 0$, such that $\sum_{i=1}^{\ell} \mu_i < \frac{1}{2}$, then $\sum_{i=1}^{\ell} f(\mu_i) \leq \frac{7}{10}$. Clearly, $\mu_2 < \frac{1}{3}$ (as otherwise $\mu_1 + \mu_2 \geq 2\mu_2 > \frac{2}{3}$).

There is at most one job of weight in $(\frac{1}{3}, \frac{1}{2})$ in I'. If there is one such job we show that without loss of generality, there is at most one other job in I', and its weight is in $(0, \frac{1}{3}]$. Else, we show that there are at most two jobs, and their weights are in $(0, \frac{1}{3}]$. Let $f_1(y) = \frac{y}{1-y}$ and let $f_2(y) = \frac{2y}{y+1}$.

The function $f_1(y)$ is convex, thus, for any pair of jobs of weights α, β such that $\alpha + \beta \in (0, \frac{1}{3}]$, $f_1(0) + f_1(\alpha + \beta) \geq f_1(\alpha) + f_1(\beta)$ holds. As $f_1(0) = 0$, it turns into $f_1(\alpha + \beta) \geq f_1(\alpha) + f_1(\beta)$. Therefore, if there are two jobs of total weight at most $\frac{1}{3}$, then they can be combined into a single job while as a result, their total size cannot decrease. If there exists a job of weight larger than $\frac{1}{3}$, then the total weight of jobs of weight at most $\frac{1}{3}$ is at most $\frac{1}{6}$, so they can all be combined into a single job. Moreover, among any three jobs of a total weight of at most $\frac{1}{2}$, there exists a pair of jobs of total weight no larger than $\frac{1}{3}$, which can be combined as described above, so if there is no job of weight larger than $\frac{1}{3}$, still jobs can be combined until at most two jobs remain. Thus, there are only two cases to consider.

Case 1 There is one job of weight in $[0, \frac{1}{3}]$ and at most one job of weight in $(\frac{1}{3}, \frac{1}{2})$. Let their weights be $0 \leq y_1 \leq \frac{1}{6}$ and $\frac{1}{3} < y_2 < \frac{1}{2} - y_1$ (the case $y_1 = \frac{1}{6}$ has the meaning that the job of weight in $(\frac{1}{3}, \frac{1}{2})$ does not exist). We have $f_1(y_1) + f_2(y_2) < f_1(y_1) + f_2(\frac{1}{2} - y_1)$, due to the monotonicity of f_2. We bound the function $f_1(y_1) + f_2(\frac{1}{2} - y_1) = \frac{y_1}{1-y_1} + \frac{1-2y_1}{\frac{3}{2}-y_1}$. This function of y_1 is monotonically increasing for $y_1 \in [0, \frac{1}{6}]$, so its greatest value is for $y_1 = \frac{1}{6}$, and it is $\frac{1}{5} + \frac{1}{2} = \frac{7}{10}$.

Case 2 There are at most two jobs, where each job has a weight in $(0, \frac{1}{3}]$. If there is at most one job, then its size is at most $\frac{1}{2}$. We therefore focus on the case of exactly two such jobs. Recall that the total weight of the two jobs is larger than $\frac{1}{3}$ (since they cannot be combined). The total weight of these jobs is less than $\frac{1}{2}$, and we denote their weights by $y > \frac{1}{6}$ and $\frac{1}{3} - y < y' < \frac{1}{2} - y$. By the monotonicity of f,

$f_1(y) + f_1(y') \leq f_1(y) + f_1(\frac{1}{2} - y) = \frac{y}{1-y} + \frac{\frac{1}{2}-y}{y+\frac{1}{2}}$. This function is monotonically decreasing in $(\frac{1}{6}, \frac{1}{4}]$ and increasing in $(\frac{1}{4}, \frac{1}{3}]$, and its values at the endpoints $\frac{1}{6}$ and $\frac{1}{3}$ are both $\frac{7}{10}$. □

Theorem 15. POA ≤ 1.7.

Proof. This follows from Claims 13 and 14. □

Next, we provide a tight lower bound.

Theorem 16. POA ≥ 1.7.

Proof. For $i \geq 0$, let $n_i = \frac{2(4^i-1)}{3}$. Note that $n_0 = 0$, and $n_i = 4n_{i-1} + 2$ for all $i \geq 1$. We define an instance of machine covering $\mathscr{I}(\ell)$ for every integer $\ell \geq 1$. Let

$$\delta = \frac{1}{30n_\ell} = \frac{1}{20(4^\ell - 1)} .$$

There are $m = 2(10^\ell - 1) = 18\sum_{j=0}^{\ell-1} 10^j$ identical machines, and five types of jobs.

The first type of jobs are m jobs of size 1. The other job types $2, 3, 4, 5$ are summarized in the following table. These jobs are introduced for $1 \leq i \leq \ell$.

Type	Size	Number of jobs
2	$a_i = \frac{1}{2} + (n_i - 1)\delta$	$6 \cdot 10^{\ell-i}$
3	$b_i = \frac{1}{2} - (n_i - 1)\delta$	$12 \cdot 10^{\ell-i}$
4	$c_i = \frac{1}{5} + 4n_{i-1}\delta$	$12 \cdot 10^{\ell-i}$
5	$d_i = \frac{1}{5} - n_i\delta$	$6 \cdot 10^{\ell-i}$

Note that $c_1 = \frac{1}{5} + 4n_0\delta = \frac{1}{5}$, and $d_\ell = \frac{1}{5} - n_\ell\delta = \frac{1}{5} - \frac{1}{30} = \frac{1}{6}$.

Lemma 17. *For any $\mathscr{I}(\ell)$, there exists an assignment that has the cover value* 1*, and it is an* NE.

Proof. We define an assignment where every machine has load of at least 1, and for every job, the total size of other jobs assigned to the same machine is at most 1. This last property implies that the assignment is an NE, by Observation 1.

Let assignment $\mathcal{A}$ be defined as follows.

(i) Jobs of size 1 are assigned in pairs to $m_1 = \frac{m}{2}$ machines.

(ii) For every $i = 1, 2, \ldots, \ell$, there are $6 \cdot 10^{\ell-i}$ machines with two jobs of size b_i and one job of size a_i assigned to each such machine. This results in the load $a_i + 2b_i = \frac{1}{2} + (n_i - 1)\delta + 2(\frac{1}{2} - (n_i - 1)\delta) = \frac{3}{2} - (n_i - 1)\delta$, and the total size of any two jobs assigned to such a machine is at most $a_i + b_i = 1$. These jobs occupy $m_2 = 6\sum_{j=0}^{\ell-1} 10^j = 6\frac{10^\ell - 1}{9}$ machines in total.

(iii) The jobs of size c_1 are assigned to $m_3 = 2 \cdot 10^{\ell-1}$ machines, such that every such machine has six jobs. The load of each machine is $\frac{6}{5}$, and the size of every five such jobs is exactly 1.

(iv) For every $i = 1, 2, \ldots, \ell - 1$, one job of size c_{i+1} is assigned with five jobs of size d_i to a machine. The number of jobs of the first size is $12 \cdot 10^{\ell-i-1}$, and the number of jobs of the second size is $6 \cdot 10^{\ell-i} = 5 \cdot 12 \cdot 10^{\ell-i-1}$ (note that $\ell - i - 1 \geq 0$), thus the number of required machines is $12 \cdot 10^{\ell-i-1}$. The load of each machine is $c_{i+1} + 5d_i = \frac{1}{5} + 4n_i\delta + 5(\frac{1}{5} - n_i\delta) = \frac{6}{5} - n_i\delta$. The total size of five jobs assigned to one machine is at most $c_{i+1} + 4d_i = 1$.
This subset of jobs occupies $m_4 = 12\sum_{j=0}^{\ell-2} 10^j = 12\frac{10^{\ell-1} - 1}{9}$ machines in total.

(v) All the jobs of size d_ℓ (six such jobs) are assigned together to one machine. The load of this machine is 1. Let $m_5 = 1$.

Since $\sum_{j=1}^{5} m_j = \frac{m}{2} + 6\frac{10^\ell - 1}{9} + 2 \cdot 10^{\ell-1} + 12\frac{10^{\ell-1}-1}{9} + 1 = \frac{m}{2} + 10^\ell - 1 = m$, we see that $\mathcal{A}$ is well-defined and has the cover value 1 (since there is one machine with load 1, and other machines have larger loads). □

Lemma 18. *For any $\mathscr{I}(\ell)$, there exists an assignment that has a cover value of* $1.7 - \delta = 1.7 - \frac{1}{30n_\ell}$.

Proof. Every machine will be assigned one job of size 1, and a pair of jobs, either of sizes a_i and d_i, or of sizes b_i and c_i (for some $1 \le i \le \ell$). We have $1+a_i+d_i = \frac{17}{10} - \delta$, and $1+b_i+c_i = \frac{17}{10} + (4n_{i-1} - n_i + 1)\delta = \frac{17}{10} - \delta$.

The number of machines is $18\sum_{j=0}^{\ell-1} 10^j = m$, as required. □

With ℓ tending to ∞, $\delta = \frac{1}{30n_\ell}$ approaches zero, which implies that the POA of the covering game on identical machines is at least 1.7. □

3 The POS of the machine covering game

We show that for every instance of the machine covering game, there exists an optimal assignment that is also an NE. Our proof technique is based upon the technique which was used in [17, 14] to prove that in job scheduling games where the selfish goal of the players is to run on the least loaded machine (like in our machine covering game), any sequence of improvement steps converges to an NE. To compare assignments, we use a lexicographical non-decreasing ordering $\prec_L$ of the vectors of their loads, where these vectors are first sorted internally by non-increasing load.

Lemma 19. *For any instance of the machine covering game, an optimal schedule that is minimal with respect to the lexicographical order is an* NE.

Proof. Let $\mathcal{A}^*$ be an optimal assignment such that no other optimal assignment $\mathcal{A}$ satisfies $\mathcal{A} \prec_L \mathcal{A}^*$. We show that $\mathcal{A}^*$ is an NE assignment. Assume by contradiction that $\mathcal{A}^*$ is not NE, that is, there is at least one job that would benefit from moving to another (lower) machine. Consider the maximum index machine with such a job, call it i and let the job be denoted by k, and its size by p. Job k is selfish, and will move to the least-loaded machine (or one of the least-loaded machines if there are several such machines) in $\mathcal{A}^*$. In particular it will want to move to machine m. Denote the resulting assignment by $\mathcal{A}'$.

We have $L_{max}(\mathcal{A}^*) - p \ge L_i(\mathcal{A}^*) - p > L_m(\mathcal{A}^*) = L_{min}(\mathcal{A}^*)$ by the assumption and the sorting of the machines. This implies that the maximum load does not increase and the minimum load does not decrease as a result of the move of k. Hence, the cover can only increase, and since $\mathcal{A}^*$ is optimal, it must remain unchanged. So the minimum load remains the same as well.

As a result of k moving from i to m, i may move down in the machines since they are ordered by load, while m may move up. If they do not pass each other, it is clear that the new vector is lexicographically smaller since the loads on all machines until i (wherever it is now) are the same or smaller, and at least one is really smaller. If they do pass each other, consider the machines in the order in which they are in assignment $\mathcal{A}^*$, but with machines i and m interchanged. Then move m down to its correct place (it does not pass i). The resulting load vector, restricted to the first j entries where j is the final position of machine m, is lexicographically smaller than $\mathcal{A}^*$ analogously to before (the new load of m is less than the old load of i). Then this also holds for $\mathcal{A}'$ which only permutes the remainder of the vector, which we ignore for the lexicographic comparison. But this contradicts the minimality of $\mathcal{A}^*$. □

Theorem 20. $\text{POS}(m) = 1$ *for any* $m \ge 2$.

Proof. Since for any set of n jobs there are finitely many possible assignments, among the assignments that are optimal with respect to our social goal there exists at least one which is minimal with respect to the total order $\prec_L$, and according to Lemma 19 this assignment is an NE. As no NE assignment can have a strictly greater social value than the optimal one, we conclude that $\text{POS} = 1$. □

4 Mixed equilibria for the machine covering game

In the setting of mixed strategies, we consider the case of identical machines, similarly to [21]. In that work, it was shown that the mixed POA for two machines and the makespan minimization objective is $\frac{3}{2}$. In this section, we prove that the mixed POA for two machines is equal to 2 for the objective of maximizing the value of the cover.

We start by showing that for m identical machines, $\text{MPOA}(m)$ can be exponentially large as a function of m, unlike the makespan minimization problem, where the mixed POA is $\Theta(\frac{\log m}{\log \log m})$ [21, 5].

Theorem 21. $\text{MPOA}(m) \geq \frac{m^m}{m!}$.

Proof. Consider the following instance of the machine covering game with m machines. Let $n = m$, and $p_j = 1$ for $1 \leq j \leq n$ (the condition $n = m$ is used in what follows). Next, we describe the strategy of each job. Each job j, $j = 1, \ldots, n$ chooses each one of the m machines with probability $1/m$. The expected load of each machine is therefore 1 (every job contributes $\frac{1}{m}$ to the load of machine i, which is its size times the probability to be assigned to machine i). The expected cost of job j if it is assigned to machine i is $1 + \frac{m-1}{m}$. Thus, each job has the same cost on each one of the machines and thus has no incentive to change its probability distribution vector. As there are exactly $n = m$ jobs, if each job selects a different machine, then the value of cover is 1, while otherwise it is 0. The probability that the resulting schedule is such that each job selects a different machine is $\frac{m!}{m^m}$. We find that for the mixed Nash equilibrium the expected minimum load is $\frac{m!}{m^m}$. A socially optimal solution can be achieved by (deterministically) allocating each job to a different machine (and it has a social value 1), and so it follows that the mixed POA for this game is $\frac{m^m}{m!}$. We conclude that $\text{MPOA}(m) \geq \frac{m^m}{m!}$. □

Theorem 22. $\text{MPOA}(2) = 2$.

Proof. The lower bound follows from Theorem 21.

Consider a mixed NE for a given machine covering game. Consider a specific job i and let q_i be the probability that the job (which has a size of p_i) is assigned to the most loaded machine of the two. It was shown in [21] that the expected maximum load over the two machines, call it $\mathbb{E}(\text{MAKESPAN})$, satisfies $\mathbb{E}(\text{MAKESPAN}) = \sum_k q_k p_k \leq (\frac{3}{2} - q_i) \sum_k p_k + (2q_i - \frac{3}{2})p_i$, for any job i. We use $\mathbb{E}(\text{COVER})$ to denote the expected value of the cover.

As there are two machines, this implies that the expected minimum load over the two machines,

$$\mathbb{E}(\text{COVER}) = \sum_k (1 - q_k)p_k = \sum_k p_k - \sum_k q_k p_k = \sum_k p_k - \mathbb{E}(\text{MAKESPAN}) \,,$$

is at least

$$\sum_k p_k - (\frac{3}{2} - q_i) \sum_k p_k - (2q_i - \frac{3}{2})p_i = (q_i - \frac{1}{2}) \sum_k p_k - (2q_i - \frac{3}{2})p_i \,.$$

Note that $\text{COVER}(\text{OPT}) \leq \frac{\sum_k p_k}{2}$, which is the case if we can distribute the weight of each job in an equal manner over the two machines. Also, if there exists a job i such that $p_i > \sum_{k \neq i} p_k$, then $\text{COVER}(\text{OPT}) \leq \sum_{k \neq i} p_k$. In total, $\text{COVER}(\text{OPT}) \leq \min\{\sum_{k \neq i} p_k, \frac{\sum_k p_k}{2}\}$.

One of the following may occur:
1. There exists a job i such that $q_i > \frac{3}{4}$. Therefore, $\mathbb{E}(\text{COVER}) \geq (q_i - \frac{1}{2}) \sum_{k \neq i} p_k + (1 - q_i)p_i = (1 - q_i) \sum_k p_k + \sum_{k \neq i} p_k(2q_i - \frac{3}{2}) \geq (2(1 - q_i) + 2q_i - \frac{3}{2})\text{COVER}(\text{OPT}) \geq \frac{1}{2}\text{COVER}(\text{OPT})$.
2. For any job i, $q_i \leq \frac{3}{4}$. Therefore, $\mathbb{E}(\text{COVER}) = \sum_k (1 - q_k)p_k \geq \frac{1}{4} \sum_k p_k \geq \frac{1}{4} \cdot 2\text{COVER}(\text{OPT}) = \frac{1}{2}\text{COVER}(\text{OPT})$.

In both cases, we get that $\mathbb{E}(\text{COVER}) \geq \frac{1}{2}\text{COVER}(\text{OPT})$, which proves our claim. □

5 The envy-ratio game

In this paper, we investigated up until now the tradeoff between fairness and optimality of an NE schedule, when the fairness was considered with respect to maximizing the minimum load in the schedule. However, there are different accepted notions of fairness of a job allocation. Another suitable fairness criterion that we can consider in the same setting is the so-called maximum envy-ratio.

The envy-ratio of job i for job j is the cost of i over the cost of j. As before, the set of players is the set of n jobs to be scheduled on m identical machines. The definitions of loads and costs remain unchanged. The objective is to schedule the jobs so as to minimize the ratio of the maximum load over the minimum load of the machines in the assignment. Going back to the application of data routing on identical parallel links, this objective aims to reduce the "envy" of the job (or jobs) that suffers the greatest delay towards the job (or jobs) that suffers the smallest delay in the network (assuming that each machine has at least one job assigned to it).

For the study of this problem, we use the following notations. Let $\mathcal{B}$ be a schedule. The maximum envy-ratio of schedule $\mathcal{B}$ is defined by $e(\mathcal{B}) = L_{max}(\mathcal{B})/L_{min}(\mathcal{B})$. Accordingly, we denote the maximum envy-ratio of an optimal schedule OPT by $e(\text{OPT})$.

The maximum envy-ratio objective was previously considered in the context of scheduling by [4]. It is shown that Graham's greedy algorithm (also known as Longest Processing Time or LPT) [18] has an approximation ratio of exactly 1.4 for the maximum envy-ratio minimization problem. This fairness criterion was considered in a game theoretic setting by [22]. They considered the envy-ratio objective for the problem of envy-free allocation of indivisible goods. In the special case where all the players have the same utility function for each good, one can think of the players as identical machines and the set of goods as a set of jobs, and then this problem is equivalent to minimizing the ratio of the maximum completion time over the minimum completion time.

Viewing this as a game similar to the machine covering game, with the only change that the social goal is minimization of the envy-ratio, we call this the envy-ratio game. The measures POA and POS were not previously considered for this objective in the setting of selfish jobs. They are respectively defined as the worst-case ratio between $e(\mathcal{A})$, where $\mathcal{A}$ is an NE assignment with the highest/lowest maximum envy-ratio, and $e(\text{OPT})$. We show tight bounds of 2 on the POA and 1 on the POS for any $m \geq 2$. For the POS, we can in fact follow the approach of Section 3 in its entirety, noting that the proof of Lemma 19 actually shows that the envy-ratio does not increase as a result of the move of k. Thus, $\text{POS}(m) = 1$ for all $m \geq 2$. Next, we analyze the POA for this game.

Theorem 23. *For the envy-ratio game,* $\text{POA}(m) = 2$ *for any* $m \geq 2$.

Proof. We start with the upper bound. Consider an NE assignment $\mathcal{A}$ with m machines. If all machines in $\mathcal{A}$ are assigned at most one job then the schedule is optimal. Moreover, in the case that there is exactly one machine that has more than one job assigned to it, and this is a machine of load L_{min}, the schedule is optimal as well. This holds since any schedule has a machine of load L_{max} (since at least one machine with this load has a single job assigned to it that must be assigned to a machine by any schedule), and any schedule has a machine that does not contain any of the jobs that are assigned to a dedicated machine in

$\mathcal{A}$, where the load of such a machine cannot exceed L_{min}. Consider a machine i in $\mathcal{A}$ which has at least two jobs assigned to it, having a maximum load among such machines. By the discussion above regarding optimal schedules, the machine i cannot be the unique machine of $\mathcal{A}$ with a load of L_{min}. Consider the size of a smallest job running on i, denoted by p. As this job is assigned with at least one more job, $p \leq \frac{1}{2}L_i$. As this schedule is an NE, and there is a machine $i' \neq i$ that has a load of L_{min}, $L_{min} + p \geq L_i$ holds, since the job does not benefit from moving to i'. Hence $L_{min} \geq L_i - p \geq \frac{1}{2}L_i$.

If $L_i = L_{max}$, then it immediately follows that $e(\mathcal{A}) \leq 2$, and as the envy-ratio of the optimal schedule $e(\text{OPT}) \geq 1$, we get that POA ≤ 2. Next, we consider the case $L_i < L_{max}$. In this case, by the definition of i, all machines that run at least two jobs have loads no larger than L_i, and thus L_{max} is the load of at least one machine of $\mathcal{A}$ that runs a single job. Let $1 \leq k \leq m-1$ denote the number of machines in $\mathcal{A}$ having loads strictly above L_i (and thus running exactly one job each), and let J_{long} denote the k jobs assigned to them in $\mathcal{A}$. In particular, as among these machines there is a machine of load L_{max}, there is at least one job of size exactly L_{max}. Thus we find $L_{max}(\text{OPT}) \geq L_{max}$, since any schedule must assign this job. In OPT there are at least $m-k$ machines which have no jobs of J_{long}, and the total size of all other jobs is at most $(m-k)L_i$. Therefore, $L_{min}(\text{OPT}) \leq L_i$. We get $e(\text{OPT}) = \frac{L_{max}(\text{OPT})}{L_{min}(\text{OPT})} \geq \frac{L_{max}}{L_i}$ and POA $\leq \frac{e(\mathcal{A})}{e(\text{OPT})} \leq \frac{L_{max}/L_{min}}{L_{max}/L_i} = \frac{L_i}{L_{min}} \leq 2$.

The matching lower bound is derived using the construction of [29]. Consider the following game instance with m machines, where the set of the jobs contains $m(m-1)$ jobs of size $\frac{1}{m}$ and two jobs of size 1, and consider the schedule where each machine among machines $1, \ldots, m-1$ runs m of the jobs of size $\frac{1}{m}$ and has load 1, and machine m runs the two jobs of size 1. This schedule is an NE by Observation 1 (since its value of cover is 1, this observation can be used). The load of m is 2 while all other loads are 1. Thus, the maximum envy-ratio of this schedule is exactly 2 for any value of m. The optimal maximum envy-ratio is 1, and it is obtained by a schedule where $m-2$ machines run $m+1$ jobs of size $\frac{1}{m}$ each, and each additional machine runs one job of size 1 and one job of size $\frac{1}{m}$. This results in a schedule with an maximum envy-ratio of 1 since the machines are balanced (and have loads of $1 + \frac{1}{m}$). The POA of this instance of the game is exactly 2. □

References

[1] N. Alon, Y. Azar, G. J. Woeginger, and T. Yadid. Approximation schemes for scheduling. In *Proc. 8th Symp. on Discrete Algorithms (SODA'97)*, pages 493–500, 1997.

[2] 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.

[3] N. Bansal and M. Sviridenko. The Santa Claus problem. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC'06)*, pages 31–40, 2006.

[4] E. G. Coffman Jr. and M. A. Langston. A performance guarantee for the greedy set-partitioning algorithm. *Acta Informatica*, 21(4):409–415, 1984.

[5] A. Czumaj and B. Vöcking. Tight bounds for worst-case equilibria. *ACM Transactions on Algorithms*, 3(1), 2007.

[6] 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.

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

[8] G. Dósa and J. Sgall. First Fit bin packing: A tight analysis. Manuscript, 2012.

[9] 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.

[10] 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.

[11] L. Epstein, E. Kleiman, and R. van Stee. The cost of selfishness for maximizing the minimum load on uniformly related machines. Journal of Combinatorial Optimization, to appear, 2012.

[12] 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.

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

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

[15] 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.

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

[17] 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.

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

[19] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing*, 3(4):299–325, 1974.

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

[21] 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.

[22] R. J. Lipton, E. Markakis, E. Mossel, and A. Saberi. On approximately fair allocations of indivisible goods. In *ACM Conference on Electronic Commerce*, pages 125–131, 2004.

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

[24] J. Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951.

[25] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35(1):166–196, 2001.

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

[27] T. Roughgarden and É. Tardos. How bad is selfish routing? *Journal of the ACM*, 49(2):236–259, 2002.

[28] A. Sahasrabudhe and K. Kar. Bandwidth allocation games under budget and access constraints. In *Proc. of the 42nd Annual Conference on Information Sciences and Systems (CISS'08)*, pages 761–769, 2008.

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

[30] Z. Tan, L. Wan, Q. Zhang, and W. Ren. Inefficiency of equilibria for the machine covering game on uniform machines. *Acta Informatica*, 49(6):361–379, 2012.