

Maximizing the Minimum Load: The Cost of Selfishness

Leah Epstein¹, Elena Kleiman¹, and Rob van Stee^{2*}

¹ Department of Mathematics, University of Haifa, 31905 Haifa, Israel.
lea@math.haifa.ac.il, elena.kleiman@gmail.com

² Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany.
vanstee@mpi-inf.mpg.de

Abstract. We consider a scheduling problem where each job is controlled by a selfish agent, who is only interested in minimizing its own cost, which is defined as the total load on the machine that its job is assigned to. We consider the objective of maximizing the minimum load (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). We show that on related machines, both these values are unbounded. We then focus on identical machines. We show that the POS is 1, and we derive tight bounds on the POA for $m \leq 6$ and nearly tight bounds for general m . In particular, we show that the POA is at least 1.691 for larger m and at most 1.7. Hence, surprisingly, the POA is less than the POA for the makespan problem, which is 2. To achieve the upper bound of 1.7, we make an unusual use of weighting functions. Finally, in contrast we show that the mixed POA grows exponentially with m for this problem, although it is only $\Theta(\log m / \log \log m)$ for the makespan.

1 Introduction

Classical optimization problems, and network optimization problems in particular, are often modeled 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. This is a state which is stable in the sense that no agent can gain from unilaterally switching strategies. Following recent interest of computer scientists in game theory [17, 13, 19], we study Nash equilibria for a scheduling problem where the goal is maximizing the minimum load.

This goal function is motivated by issues of Quality of Service and fair resource allocation. It is useful for describing systems where the complete system relies on keeping all the machines productive for as long as possible, as the entire system fails in case even one of the machines ceases to be active. From the networking aspect, this problem has applications to basic problems in network optimization such as fair bandwidth allocation. Consider pairs of terminal nodes

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

that wish to communicate; we would like to allocate bandwidth to the connections in a way that no link unnecessarily suffers from starvation, and all links get a fair amount of resources. Another motivation is efficient routing of traffic. Consider parallel links between pairs of terminal nodes. Requests for shifting flow are assigned to the links. We are interested in having the loads of the links balanced, in the sense that each link should be assigned a reasonable amount of flow, compared to the other links. Yet another incentive to consider this goal function is congestion control by fair queuing. Consider a router that can serve m shifting requests at a time. The data pieces of various sizes, need to be shifted, are arranged in m queues (each queue may have a different data rate), each pays a price which equals the delay that it causes in the waiting line. Our goal function ensures that no piece gets a "preferred treatment" and that they all get at least some amount of delay.

The problem of maximizing the minimum load, seeing jobs as selfish agents, can be modeled 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 a certain amount of unsplitable flow between the two nodes. Requests are to be assigned to links and consume bandwidth which depends on their sizes. The cost charged from 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 [13], but our model has a different social goal function. To demonstrate the non-triviality of the problem, see Figure 1.

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.

In our scheduling model, the *coordination ratio*, or *price of anarchy* (POA) [18] 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.

In addition, we study the *mixed* POA (MPOA), where we consider mixed Nash equilibria that result from mixed strategies, where the player's choices are not deterministic and are regulated by probability distributions on a set M of pure strategies. A mixed Nash equilibrium is characterized by the property that there is no incentive for any job to deviate from its probability distribution (a deviation is 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 [16]. 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 also a mixed NE.

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. [4, 2, 6]). Various game-theoretic aspects of max-min fairness in resource allocation games were considered before this paper (e.g. in [20]), but unlike the makespan minimization problem POA and POS of which were extensively studied (see [13, 3, 15]), 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 was studied recently in [7, 5].

For identical machines, we show that the POS is equal to 1. As our main result, we study the pure POA and show close bounds on the overall value of the POA ($\text{POA} = \sup_m \text{POA}(m)$, where $\text{POA}(m)$ is the POA on m machines), i.e., that it is at least 1.691 and at most 1.7. This in contrast with the makespan minimization problem, where it is known that the POA for m identical machines is $\frac{2m}{m+1}$, giving an overall bound of 2 [10, 21]. This is rather unusual, as the cover maximization problem is typically harder than the makespan minimization problem, thus it could be expected that the POA for the covering problem would be higher.

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, these lower and upper bounds are approximation ratios of well known algorithms for Bin-Packing (Harmonic [14] and First-Fit [12], respectively). We furthermore prove that the POA is monotonically non-decreasing as a function of m . For small numbers of machines, in the full version we provide the exact values of POA: we find $\text{POA}(2) = \text{POA}(3) = 3/2$ and $\text{POA}(4) = 13/8 = 1.625$. We show $\text{POA}(m) \geq \frac{5}{3}$ for $m > 5$. As for the MPOA, we show that its value is very large as a function of m , and $\text{MPOA}(2) = 2$.

In contrast to these results, we can 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 is defined to be the maximum speed ratio between any pair of machines). 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 [9]).

2 The model

In this section, we define the more general model of scheduling on related machines, which we will consider first. A set of n jobs $J = \{1, 2, \dots, n\}$ is to be assigned to a set of m machines $M = \{M_1, \dots, M_m\}$, where machine M_i has a speed s_i . If $s_i = 1$ for $i = 1, \dots, m$, the machines are called identical. This is an important and widely studied special case of uniformly related machines. The size of job $1 \leq k \leq n$ is denoted by p_k . An assignment or schedule is a function $\mathcal{A} : J \rightarrow M$. The load of machine M_i , which is also called the delay of this

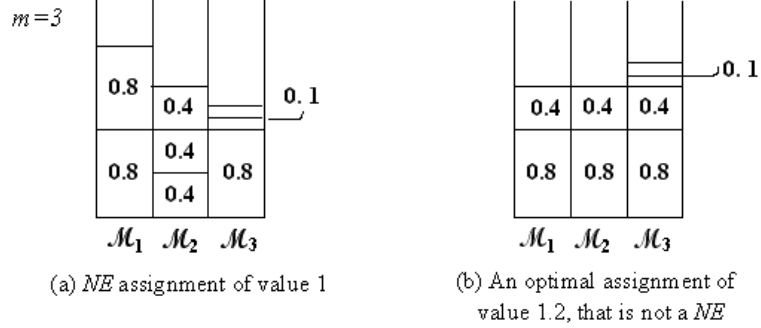


Fig. 1. An example of two packings with different social values. This example demonstrates the non-triviality of the problem. There are three jobs of size 0.8, three jobs of size 0.4 and two jobs of size 0.1. The three machines are identical. The assignment on the right hand side is not a Nash equilibrium, since a job of size 0.1 would reduce its delay from 1.4 to 1.3 by migrating to another machine. The social value of this assignment is 1.2. The assignment on the left hand side is a Nash equilibrium, but its social value is only 1.

machine, is $L_i = \sum_{k:A(k)=M_i} \frac{p_k}{s_i}$. 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 $c_k : \mathcal{S} \rightarrow \mathbb{R}$. 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 [11, 8]. 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. $\mathbb{E}(\min_{i \in M} L_i)$.

We omit some of the proofs due to space constraints.

3 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, \dots, 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.

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. \square

4 The POS for m identical machines

Theorem 2. *On identical machines, $\text{POS} = 1$ for any m .*

Proof. We show that for every instance of the machine covering game, among the optimal assignments there exists an optimal assignment which is also an NE. Our proof technique is based upon the technique which was used in [8, 11] to prove that in job scheduling games where the selfish goal of the players is run on the least loaded machine (like in our machine covering game), any sequence of improvement steps converges to an NE.

We first define a complete order relation on the assignments, and then show that an optimal assignment which is the “highest” among all optimal assignments with respect to this order is always an NE.

Definition 1. A vector (l_1, l_2, \dots, l_m) is larger than $(l'_1, l'_2, \dots, l'_m)$ with respect to the inverted lexicographic order, if for some i , $l_i > l'_i$ and $l_k = l'_k$ for all $k > i$. An assignment s is called larger than s' according to the inverted lexicographic order if the vector of machine loads $L(s) = (L_1(s), L_2(s), \dots, L_m(s))$, sorted in non-increasing order, is larger in the inverted lexicographic order than the vector $L(s') = (L_1(s'), L_2(s'), \dots, L_m(s'))$, sorted in non-increasing order. We denote this relation by $s \succ_{L^{-1}} s'$.

Lemma 1. For any instance of the machine covering game, a maximal optimal packing w.r.t. the inverted lexicographic order is an NE.

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 maximal w.r.t. the total order $\succ_{L^{-1}}$, and according to Lemma 1 this assignment is an NE. As no NE assignment can have a strictly greater social value than the optimal one, we conclude that $\text{POS} = 1$. \square

5 The price of anarchy

Figure 1 clearly demonstrates that not every NE schedule is optimal. We next 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 mentioned before, the measure metrics we use are the POA and the POS. In Figure 2, we give some lower bounds on the POA for small m (without proof).

Consider a pure NE assignment of jobs to machines, denoted by \mathcal{A} , for an instance of the machine covering game. We assume that the social value of \mathcal{A} , that is, the load of the least loaded machine in \mathcal{A} , is 1. Otherwise, we can simply scale all sizes of jobs in the instances which we consider so that $\text{COVER}(\mathcal{A}) = 1$.

We denote a machine which is loaded by 1 in \mathcal{A} by P . All other machines are called *tall* machines. We would like to estimate the load of P in the optimal assignment. Let $C = \text{COVER}(\text{OPT})$. Obviously, $C \geq 1$, and the total sum of jobs sizes, denoted by W , satisfies $W \geq mC$. First, we introduce some assumptions on \mathcal{A} . Note that the modifications needed to be applied such that this instance will satisfy these assumptions do not increase $\text{COVER}(\mathcal{A})$, do not violate the conditions for NE and do not decrease $\text{COVER}(\text{OPT})$.

1. Machine P contains only tiny jobs, that is, jobs of infinitesimal size.
Since no machine has a smaller load, replacing the jobs on this machine by tiny jobs keeps the schedule as an NE. The value $\text{COVER}(\text{OPT})$ may only increase.
2. For a tall machine in \mathcal{A} which has two jobs, both jobs have a size of 1.
If one of them is larger, then the second job would want to move to P , so this case cannot occur. If some such job is smaller, its size can be increased up to 1 without affecting the NE.
3. This assignment is minimal with respect to the number of machines (among assignments for which $\text{COVER}(\text{OPT}) \geq C$). In particular, no machine in \mathcal{A} has a single job.

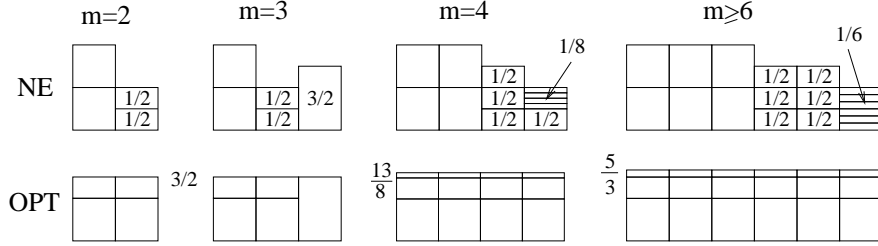


Fig. 2. Lower bounds for the price of anarchy for small m . Machines are on the horizontal axis, jobs are on the vertical axis. The squares in the first few figures represent jobs of size 1.

Else, if some machine has a single job, remove this machine and the job from \mathcal{A} , and the machine with it from the optimal assignment OPT . Assign any remaining jobs that ran on this machine in OPT arbitrarily among the remaining machines. This gives a new assignment with $\text{COVER}(\text{OPT}) \geq C$, and less machines.

4. Given jobs of sizes $p_1 \leq p_2 \leq \dots \leq p_t$ assigned to a machine Q in \mathcal{A} , then $p_2 + \dots + p_t = 1$. In fact, $p_2 + \dots + p_t > 1$ is impossible since this would mean that the job of size p_1 has an incentive to move. If the sum is less, enlarge the size p_t to $1 - p_{t-1} - \dots - p_2$. This does not affect the NE conditions, and keeps the property $\text{COVER}(\text{OPT}) \geq C$.
5. Consider a machine $Q \neq P$ in \mathcal{A} which has $t \geq 3$ jobs assigned to it. Let a, b denote the sizes of the smallest and largest jobs on it, respectively. Then, $b < 2a$.

Otherwise, if we have an assignment where $b \geq 2a$, replace b with two jobs of size $\frac{b}{2}$. This modification preserves the NE, as the new jobs do not have an incentive to move; Let T denote the total size of jobs on machine Q . As a does not want to move to P , $T \leq 1 + a$ holds. As in this case $a \leq \frac{b}{2}$, we have $T \leq 1 + \frac{b}{2}$, whereas $1 + \frac{b}{2}$ would have been the load of P if the job $\frac{b}{2}$ moved there.

Lemma 2. *No job has a size larger than 1.*

Proof. There is no machine in \mathcal{A} with a single job. □

Lemma 3. *There is no job of size in $[\frac{2}{3}, 1)$ assigned to a machine Q ($Q \neq P$) in \mathcal{A} .*

Proof. If there is such a job, then it has at least two jobs assigned together with it, each of size greater than $\frac{1}{3}$ (due to assumption 5), which contradicts assumption 4. □

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

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

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} . In fact, for a job j of size x , the total size of jobs assigned in an NE to the same machine as j is no larger than $1 + x$. Moreover if $x > \frac{2}{3}$, then by our assumptions, it is possible to prove that the total size of these jobs is at most $2 - x$.

Its inverse function $f(y)$ is

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

Note that $f(y)$ is continuous at $\frac{1}{3}$ but not at $\frac{1}{2}$. Both functions are monotonically increasing.

We now state several lemmas, which follow from the properties of this weight function defined above.

Lemma 4. *The total weight (by w) of jobs on P in \mathcal{A} is less than 1.*

Proof. Follows from the fact that P has a load 1 and for any x , $w(x) < x$. \square

Lemma 5. *The total weight (by w) of jobs assigned to a machine $Q \neq P$ in \mathcal{A} is at most 1.*

Proof. By Assumption 3', each machine Q in \mathcal{A} has at least two jobs. The claim clearly holds for a machine with two jobs, as by Assumption 2 both these jobs have size 1, and by the definition of w have a total weight of $\frac{1}{2} + \frac{1}{2} = 1$. For a machine which has at least three jobs assigned to it, there are two cases. If there are no jobs with size in $(\frac{1}{2}, \frac{2}{3})$ (i.e., all jobs are of size in $(0, \frac{1}{2}]$), then let T be the total size of jobs on Q . For each job of size x_i , as the job does not want to move to machine P , $x_i + 1 \geq T$ holds. Combining this with the definition of w for jobs of size in $(0, \frac{1}{2}]$, we get that $w(x_i) = \frac{x_i}{x_i+1} \leq \frac{x_i}{T}$. Summing this up over all the jobs on Q proves the claim.

Else, for a job $x_i \in (\frac{1}{2}, \frac{2}{3})$ (there can be only one such job assigned to Q , by Assumption 4), we have $T \leq 2 - x_i$. Otherwise, let a be the size of the smallest job on Q . Then, by Assumption 4, $T = 1 + a$. As $T > 2 - x_i$, we get $x_i + a > 1$. Therefore, as there is an additional job of size of at least a assigned to Q , we get that the total size of all jobs except for the smallest job is more than 1, contradicting Assumption 4.

Combining this with the definition of w for jobs of size in $(\frac{1}{2}, \frac{2}{3})$, we get that $w(x_i) = \frac{x_i}{2-x_i} \leq \frac{x_i}{T}$ for this job too. \square

Lemma 6. *There is a machine in the optimal assignment with a total weight strictly smaller than 1.*

Proof. The total weight of all jobs is less than m by Lemmas 4 and 5. \square

Lemma 7. *The total size of any set of jobs with total weight below 1 is at most 1.7.*

Proof. (Sketch) To prove the claim, we use the property that there is at most one item of weight 1. If it exists, then there is at most one item of weight larger than $\frac{1}{3}$. If such an item of weight 1 does not exist, then the ratio between size and weight is at most 1.5. We find the supremum possible weight in each one of three cases.

Consider a set of jobs I of a total weight strictly below 1. Note that for any x , $w(x) \leq \frac{1}{2}$. If there is no job of weight $\frac{1}{2}$ in I , then since $\frac{2}{y+1} \leq \frac{3}{2}$ for $y \geq \frac{1}{3}$ and $\frac{1}{1-y} \leq \frac{3}{2}$ for $y \leq \frac{1}{3}$, the size of any job is at most $\frac{3}{2}$ times its weight, and thus the total size of the jobs in I does not exceed $\frac{3}{2}$.

Otherwise, there is exactly one job of weight $\frac{1}{2}$, and its size is 1. We therefore need 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. 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 $(0, \frac{1}{3}]$.

Note that $f_1(y) = \frac{y}{1-y}$ is a convex function, thus, for any pair of jobs of weights $\alpha, \beta \in (0, \frac{1}{3}]$, $f_1(0) + f_1(\alpha + \beta) \geq f_1(\alpha) + f_1(\beta)$ holds. As we can define $f_1(0) = 0$, it turns into $f_1(\alpha + \beta) \geq f_1(\alpha) + f_1(\beta)$.

Thus, any two jobs of total weight of at most $\frac{1}{3}$ can be combined into a single job while as a result, their total size cannot decrease. This is due to the convexity of f_1 and the fact that it is monotonically increasing. The replacement may only increase the weight, and respectively, the size. 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 one job of weight in $(\frac{1}{3}, \frac{1}{2})$.

Since the inverse function f is monotonically increasing as a function of y and their weight does not exceed $\frac{1}{2}$, we can assume that their total weight is $\frac{1}{2} - \gamma$ for a negligible value of $\gamma > 0$ (by increasing the weight the job of the smaller weight, which may only increase the total size). Letting $d < \frac{1}{6}$ denote the weight of the smaller job (since if $d \geq \frac{1}{6}$ then $\frac{1}{2} - \gamma - d < \frac{1}{3}$), we get a size of at most

$$\frac{d}{1-d} + \frac{2(\frac{1}{2} - \gamma - d)}{\frac{1}{2} - \gamma - d + 1} < \frac{d}{1-d} + \frac{2-4d}{3-2d}$$

(by letting $\gamma \rightarrow 0$). This function is increasing (as a function of d) so its greatest value is for $d \rightarrow \frac{1}{6}$ and it is 0.7. As the inverse function f is monotonically increasing, this case also encompass the case where there is only one job of weight in $(\frac{1}{3}, \frac{1}{2})$, and no jobs of weight in $(0, \frac{1}{3})$.

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 size of the two jobs is larger than $\frac{1}{3}$ (since they cannot be combined). The total weight of these jobs is less than $\frac{1}{2}$, so we can assume that their total weight is $1/2 - \gamma$ for a negligible value of $\gamma > 0$ (increasing the respective size). Let the weight of the large of these jobs be $d > \frac{1}{6}$. We get (by letting $\gamma \rightarrow 0$) a total size of at most

$$\frac{d}{1-d} + \frac{\frac{1}{2} - \gamma - d}{\frac{1}{2} + d + \gamma} < \frac{d}{1-d} + \frac{1-2d}{2d+1},$$

where $\frac{1}{6} < d \leq \frac{1}{3}$. 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 0.7. \square

Theorem 3. *For covering identical machines, the POA is at most 1.7.*

Proof. This follows from Lemmas 6 and 7. \square

Theorem 4. *For covering identical machines, the POA is at least 1.691.*

Proof. We first define a sequence t_i of positive integers, which is often used in the literature for analysis and proving of lower bounds for online bin packing algorithms. Let $t_1 = 1$ and $t_{i+1} = t_i(t_i + 1)$ for $i \geq 1$.

Let $m = t_k$ for an integer k . Consider the following assignment \mathcal{A} , that has $\frac{m}{t_i+1}$ machines with $t_i + 1$ jobs of size $\frac{1}{t_i}$, (for $1 \leq i < k$) and one machine (i.e., $\frac{m}{t_k}$ machines) with t_k jobs of size $\frac{1}{t_k}$. We assume that the machines are sorted in a non-increasing order w.r.t. their load. We define the load class i , $1 \leq i \leq k$ as the subset of $\frac{m}{t_i+1}$ machines with the same load $L_i = \frac{t_i+1}{t_i} = 1 + \frac{1}{t_i} > 1$ in this assignment. As t_i is an increasing sequence of integers, it follows that L_i is monotonically non-increasing as a function of i . Since $L_k = 1$, the social value of this assignment is 1. We now verify that it is a Nash equilibrium. As $L_i > 1$ for any $1 \leq i < k$, no job will benefit from leaving the machine of class L_k . It is enough to show that any job assigned to a machine of a class L_i ($1 \leq i < k-1$) would not benefit from moving to the machine of class k . Since machine i has $t_i + 1$ jobs of size $\frac{1}{t_i}$, the migration of such a job to the machine of load 1 would again result in a load of $1 + \frac{1}{t_i}$, thus the job would not benefit from the migration.

In the socially optimal assignment, each machine has a set of jobs of distinct sizes, $1, \frac{1}{2}, \frac{1}{6}, \dots, \frac{1}{t_{k-1}}, \frac{1}{t_k}$. The social value of this assignment is $\sum_{i=1}^k \frac{1}{t_i}$. Thus, the POA equals $\sum_{i=1}^k \frac{1}{t_i}$. For $k \rightarrow \infty$, this value tends to $h_\infty = \sum_{i=1}^{\infty} \frac{1}{t_i} = 1.69103\dots$, the well-known worst-case ratio of the Harmonic algorithm for bin packing. As the POA is monotonically non-decreasing as a function of the number of the machines, we conclude that this is a lower bound for any number of machines larger than t_k . In particular, the overall POA for identical machines is at least 1.69103. \square

Conjecture 1. For covering identical machines, $\text{POA} = \sum_{i=1}^{\infty} \frac{1}{t_i} = 1.691\dots$

6 Mixed equilibria

In the setting of mixed strategies we consider the case of identical machines, similarly to [13]. In that work, it was shown that the mixed POA for two machines is $\frac{3}{2}$. In this section we prove that the mixed POA for two machines is equal to 2.

We start by showing that for m identical machines, the mixed POA can be exponentially large as a function of m , unlike the makespan minimization problem, where the mixed POA is $\Theta(\frac{\log m}{\log \log m})$ [13, 3].

Theorem 5. *The mixed POA for m identical machines is at least $\frac{m^m}{m!}$.*

Proof. Consider the following instance $G \in MC$ of the Machine Covering game. $N = \{1, 2, \dots\}$ such that $p_1 = \dots = p_n = 1$, and let $M_j = \{M_1, \dots, M_m\}$, for $j = 1, \dots, n$. Each of the jobs p_i , $i = 1, \dots, n$ chooses each machine with probability $t_i^j = 1/m$. Each job sees the same expected load for each machine, and thus has no incentive to change its probability distribution vector. We get a schedule having a non-zero cover, where each job chooses to run on a different machine, with a probability of $\frac{m!}{m^m}$. So, for the mixed Nash equilibrium the expected minimum load is $\frac{m!}{m^m}$. But the coordinated optimal solution achieved by deterministically allocating each job to its own machine has a social value $\text{COVER}(\text{OPT}(G)) = 1$, and so it follows that the mixed $\text{POA}(G) = \frac{m^m}{m!}$. We conclude that the mixed $\text{POA} \geq \frac{m^m}{m!}$.

Theorem 6. *The mixed POA for two identical machines is exactly 2.*

7 Summary and conclusion

In this paper we have studied a non-cooperative variant of the machine covering problem for identical and related machines, where the selfish agents are the jobs. We considered both pure and mixed strategies of the agents. We provided various results for the POA and the POS that are the prevalent measures of the quality of the equilibria reached with uncoordinated selfish agents.

For the pure POA for m identical machines, we provided nearly tight lower and upper bounds of 1.691 and 1.7, respectively. An obvious challenge would be bridging this gap. As stated in the paper, we believe that the actual bound is the lower bound we gave.

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. In *Proc. of the 45th Symposium on Foundations of Computer Science (FOCS'04)*, pages 295–304, 2004.
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. Deuermeier, 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 2005)*, pages 110–118, 2005.
7. L. Epstein and R. van Stee. Maximizing the minimum load for selfish agents. In *Theoretical Informatics, 8th Latin American Symposium (LATIN 2008)*, pages 264–275, 2008.
8. E. Even-Dar, A. Kesselman, and Y. Mansour. Convergence time to nash equilibrium in load balancing. *ACM Trans. Algorithms*, 3(3):32, 2007.
9. 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.
10. G. Finn and E. Horowitz. A linear time approximation algorithm for multiprocessor scheduling. *BIT Numerical Mathematics*, 19(3):312–320, 1979.
11. 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. In *Proc. of the 29th International Colloquium on Automata, Languages and Programming (ICALP'02)*, pages 123–134, 2002.
12. 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.
13. 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.
14. C. C. Lee and D. T. Lee. A simple online bin packing algorithm. *Journal of the ACM*, 32(3):562–572, 1985.
15. M. Mavronicolas and P. G. Spirakis. The price of selfish routing. In *Proc. of the 33rd Annual ACM Symposium on Theory of Computing (STOC'01)*, pages 510–519, 2001.
16. J. Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951.
17. N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.
18. T. Roughgarden. *Selfish routing and the price of anarchy*. MIT Press, 2005.
19. T. Roughgarden and É. Tardos. How bad is selfish routing? *Journal of the ACM*, 49(2):236–259, 2002.
20. A. Sahasrabudhe and K. Kar. Bandwidth allocation games under budget and access constraints. In *CISS*, pages 761–769, 2008.
21. P. Schuurman and T. Vredeveld. Performance guarantees of local search for multiprocessor scheduling. *INFORMS Journal on Computing*, 19(1):52–63, 2007.