Convergence Time to Nash Equilibria

Eyal Even-Dar, Alex Kesselman, and Yishay Mansour

School of Computer Science, Tel-Aviv University,
{evend, alx, mansour}@cs.tau.ac.il.

Abstract. We study the number of steps required to reach a pure Nash Equilibrium in a load balancing scenario where each job behaves selfishly and attempts to migrate to a machine which will minimize its cost. We consider a variety of load balancing models, including identical, restricted, related and unrelated machines. Our results have a crucial dependence on the weights assigned to jobs. We consider arbitrary weights, integer weights, K distinct weights and identical (unit) weights. We look both at an arbitrary schedule (where the only restriction is that a job migrates to a machine which lowers its cost) and specific efficient schedulers (such as allowing the largest weight job to move first).

1 Introduction

As the users population accessing Internet services grows in size and dispersion, it is necessary to improve performance and scalability by deploying multiple, distributed server sites. Distributing services has the benefit reducing access latency, and improving service scalability by distributing the load among several sites. One important issue in such a scenario is how the user chooses the appropriate server. Similar problem occurs in the context of routing where the user has to select one of a few parallel links. For instance, many enterprise networks are connected to multiple Internet service providers (ISPs) for redundant connectivity, and backbones often have multiple parallel trunks.

Users are likely to behave "selfishly" in such cases, that is each user makes decisions so as to optimize its own performance, without coordination with the other users. Basically, each user would like to either maximize the resources allocated to it or, alternatively, minimize its cost. Load balancing and other resource allocation problems are prime candidates for such a "selfish" behavior.

A natural framework to analyze this class of problems is that of non-cooperative games, and an appropriate solution concept is that of Nash Equilibrium [23]. A strategy for the users is at a Nash Equilibrium if no user can gain by unilaterally deviating from its own policy. In this paper we focus on the load balancing problem. An interesting class of non-cooperative games, which is related to load balancing, is congestion games [25] and its equivalent model exact potential games [22].

Traditionally in Computer Science research has been focused on finding a global optimum. With the emerging interest in computational issues in game theory, the *coordination ratio* [18] has received considerable attention [2, 8, 9, 12, 18, 26]. The coordination ratio is the ratio between the worst possible Nash equilibrium (the one with maximum social cost) and the social optimum (an optimal solution with the minimal social cost). One motivation is to show that the gap between a Nash Equilibrium and the optimal solution is in some cases not significant, thus good performance can be achieved even without a centralized control.

In this work we are concerned with the time it takes for the system to converge to a Nash equilibrium, rather than the quality of the resulting allocation. The question of convergence to a Nash equilibrium has received significant attention in the Game Theory literature (see [11]). Our approach is different from most of that line of research in a few crucial aspects. First, we are interested in quantitative bounds, rather than showing a convergence in the limit. Second, we consider games with many players (jobs) and actions (machines) and study their asymptotic behavior. Third, We limit ourselves in this work to a subclass of games that arise from load balancing, for which there always exists a pure Nash equilibrium, and thus we can allow ourselves to study only deterministic policies.

Our Model. This paper deals with load balancing (see, [3]). Jobs (players) are allowed to select a machine to minimize their own cost. The cost that a job observes from the use of a machine is determined by the load on that machine. We consider weighted load functions, where each job has a corresponding weight and the load on a machine is sum of the weights of the jobs running on it. Until a Nash Equilibrium is reached, at least one job wishes to change its machine. In our model, similarly to the Elementary Stepwise System (see [24]), at every time step only one job is allowed to move, and a centralized controller decides which job would move in the current time step. By strategy we mean the algorithm used by the centralized controller for selecting which of the competing jobs would move. Due to the selfish nature of jobs, we assume that when a job migrates its observed load is strictly reduced, which we refer to as an *improvement* policy. We also consider the well known case of *best reply* policy, where each job moves to a machine in which its observed load is minimal.

Our Results. We assume that there are n jobs and m machines. We assume that K is the number of different weights, W is the total weight of all the jobs and w_{max} is the maximum weight assigned to a job.

For the general case of unrelated machines we show that the system always converges to a Nash equilibrium. This is done by introducing an order between the different configurations and showing that when a job migrates we move to a "lower" configuration in the order. Bounding the number of configurations by $\min\{[O(\frac{n}{Km}+1)]^{Km}, m^n\}$ derives a general bound. Using a potential base argument we derive a bound of $O(4^W)$ for integer weights, where W is the worse case sum of the weights of the jobs. For the specific strategy that first selects jobs from the most loaded machine we can show an improved bound of $O(mW + 4^{W/m+w_{max}})$.

In the simple case of identical machines and unrestricted assignments we show that if one moves the minimum weight job, the convergence may take an exponential number of steps. Specifically, the number of steps is at least,

$$\frac{(\frac{n}{K})^K}{2(K!)} = \Omega(\left(\frac{n}{K^2}\right)^K)$$

for $K = m-1$. In contrast, we show that if one moves the maximum weight job, and the jobs follow the best reply policy, a Nash Equilibrium is reached in at most n steps. This shows the importance of selecting of the "right" scheduling strategy. We also show that selecting the minimal weight job is "almost" the worst case for identical machines, by demonstrating that any strategy converges in $(\frac{n}{K}+1)^K$ time steps. We also show that any strategy converges in $O(W+n)$ steps for integer weights. For the Random and FIFO strategies we show that they converge in $O(n^2)$ steps.

For restricted assignment and related machines we bound by $O(W^2 \frac{S_{max}^2}{S_{min}})/\epsilon$ the convergence time to ϵ-Nash. Using the strategy that schedules first jobs from the most loaded machine we can derive an improved convergence bound. Note that in our setting there always exists an ϵ_{min} such that for any $\epsilon < \epsilon_{min}$ we have that any ϵ-Nash equilibrium is a Nash equilibrium. For example, in the case of identical machine with integer weights $\epsilon_{min} = 1$.

For K integer weights, we are able to derive an interesting connection between W and K, for the case of identical and related machines. We show that for any set V of K integer weights there is an equivalent set V' of K integer weights such that the maximum weight in V' is at most $O(K(cS_{max}n)^{4K})$ for some positive constant c. The equivalence guarantees that the relative cost of different machines is maintained in all configurations. (In addition, we never need to compute V', but rather it is only used in the convergence proofs.) The equivalence implies that $W = O(Kn(cS_{max}n)^{4K})$. Thus, all bounds that depend on W can depend on $O(Kn(cS_{max}n)^{4K})$.

Related Work. Milchtaich [21] describes a class of non-cooperative games, which is related to load balancing. (In order to make the relations between the models clearer we will use the load balancing terminology to describe his work.) The jobs (players) share a common set of machines (strategies). The cost of a job when selecting a particular machine depends only on the total number of jobs mapped to the machine (implicitly, all the weights are identical). However, each job has a different cost function for each machine, this is in contrast to the load balancing model where the cost of all the jobs that map to the same machine is identical. It is shown that these games always possess at least one pure (deterministic) Nash Equilibrium and there exists a best reply improvement strategy that converges in polynomial time. However, for the weighted version of this games there are cases where a pure Nash Equilibrium does not exist. In contrast, in the load balancing setting we show that any improvement policy converges to a pure Nash Equilibrium.

Our model is related to the makespan minimization problem since job moves can be viewed as a sequence of local improvements. The analysis of the approximation ratio of the local optima obtained by iterative improvement appears in [6, 7, 27]. It has also been shown in [7] that a jump (one job moves at a time) iterative improvement for two identical machines requires at most n^2 iterations, which immediately translates to an n^2 upper bound for two identical machines with general weight setting in our model. In [27] they observe that the improvement strategy that moves the maximum weight job converges in n steps.

Some interesting related learning models are stochastic fictitious play [11], graphical games [14, 20], and large population games [15]. Uniqueness of Nash Equilibria in communication networks with selfish users has been investigated in [24]. An analysis of the convergence to a Nash Equilibrium in the limit appears in [1, 5].

Paper organization: The rest of the paper is organized as follows. In Section 2 we present our model. The analysis of unrelated, related and identical machines appears in Section 3, Section 4 and Section 5, respectively. We conclude with Section 6. The Appendices contain some of the proofs.

2 Model Description

In our load balancing scenario there are m parallel machines and n independent jobs. Each job selects exactly one machine.

Machines Model. We consider identical, related and unrelated machines. We denote by S_i the speed of M_i. Let S_{min} and S_{max} denote the minimal and maximal speed, respectively. We assume that $S_{min} \geq 1$. For identical and unrelated machines we have $S_i = 1$ for $1 \leq i \leq m$.

Jobs Model. We consider both restricted and unrestricted assignments of jobs to machines. In the unrestricted assignment case each job can select any machine while in the restricted assignment case each job J can only select a machine from a pre-defined subset of machines denoted by $R(J)$.

For a job J, we denote by $w_i(J)$ the weight of J on machine M_i (where $i \in R(J)$) and by $M(J,t)$ the index of the machine on which J runs at time t. When considering identical machines, each job J has a weight $w(J) = w_i(J)$. We denote by W the maximal total weight of the jobs, that is $W = \sum_{i=1}^{n} \max_{j \in R(J_i)} \{w_j(J_i)\}$, and by $w_{max} = \max_i \max_{j \in R(J_i)} \{w_j(J_i)\}$ the maximum weight of a job.

We consider the following weight settings: *General weight setting* – the weights may be arbitrary real numbers. *Discrete weight setting* – there are K different integer weights $w_1 \leq \ldots \leq w_K = w_{max}$. *Integer weight setting* – the weights are integers.

Load Model. We denote by $B_i(t)$ the set of jobs on machine M_i at time t. The load of a machine M_i at time t is the sum of the weights of the jobs that chose M_i, that is $L_i(t) = \sum_{J \in B_i(t)} w(J)$, and its normalized load is $T_i(t) = L_i(t)/S_i$. We also define $L_{max}(t) = \max_i \{L_i(t)\}$ and $T_{max}(t) = \max_i \{T_i(t)\}$. The cost of job J at time t is the normalized load on the machine $M(J,t)$, i.e., $T_{M(J,t)}(t)$. We define the *marginal* load with respect to a job to be the load in the system when this job is removed.

System Model. The *system state* consists of the current assignment of the jobs to the machines. The system starts in an arbitrary state and each job has a full knowledge of the system state. A job wishes to migrate to another machine, if and only if, after the migration its cost is strictly reduced. Before migrating between machines, a job needs to receive a grant from the centralized controller. The controller has no influence on the selection of the target machine by a migrating job, it just gives the job a permission to migrate. The above is known in the literature as an Elementary Stepwise System (ESS) (see [5, 24]). Essentially, the controller serves as a critical section control. The execution is modeled as a sequence of steps and in each step one job changes its machine. Notice that if all jobs are allowed to move simultaneously, the system might oscillate and never ever reach a Nash Equilibrium.

Let $A(t)$ be the set of jobs that may decrease the experienced load at time t by migrating to another machine. When a migrating job selects a machine which minimizes its cost (after the migration), we call to this *best-reply* policy. Otherwise, we call to this *improvement* policy.

The system is said to reach a *pure* (or deterministic) Nash Equilibrium if no job can benefit from unilaterally migrating to another machine. The system is said to reach an ϵ-Nash Equilibrium if no job can benefit more than ϵ from unilaterally migrating to another machine. We study the number of time steps it takes to reach a Nash Equilibrium (or ϵ-Nash equilibrium) for different strategies of ESS job scheduling.

Scheduling Strategies: We define a few natural strategies for the centralized controller. The input at time t is always a set of jobs $A(t)$ and the output is a job $J \in A(t)$ which would migrate at time t. (For simplicity we assume each job has a unique weight, extension for unrelated machines is possible.) The specific strategies that we consider are:

Random: Selects $J \in A(t)$ with probability $1/|A(t)|$.
Max Weight Job: Selects $J \in A(t)$ such that $w(J) = \max_{J' \in A(T)}\{w(J')\}$.
Min Weight Job: Selects $J \in A(t)$ such that $w(J) = \min_{J' \in A(T)}\{w(J')\}$.
FIFO: Let $E(J)$ be the smallest time t' such that $J \in A(t'')$ for every $t'' \in [t', t]$.
FIFO selects $J \in A(t)$ such that $E(J) = \min_{J' \in A(T)}\{E(J')\}$.
Max Load Machine: Selects $J \in A(t)$ such that $T_{M(J,t)}$ is maximal.

3 Unrelated Machines

In this section we consider the unrelated machines case with the restricted assignment. To show the convergence we define a sorted lexicographic order of the vectors describing the machine loads as follows. Consider the sorted vector of the machine loads. One vector is called "larger" than another if its first (after the common beginning of the two vectors) load component is larger than the corresponding load component of the second vector. Formally, given two load vectors ℓ_1 and ℓ_2, let $s_1 = sort(\ell_1)$ and $s_2 = sort(\ell_2)$ where $sort()$ returns a vector in the sorted order. We define $\ell_1 \succ \ell_2$ if $s_1 \succ s_2$ using a lexicographic ordering, i.e., $s_1[i] = s_2[i]$ for $i < k$ and $s_1[k] > s_2[k]$.

We demonstrate that the sorted lexicographic order of the load vector always decreases when a job migrates. To observe this one should note that only two machine are influenced by the migration of the job J at time t, $M_i = M(J, t)$, where job J was before the migration and $M_j = M(J, t+1)$, the machine J migrated to. Furthermore $L_i(t) > L_j(t+1)$, otherwise job J would not have migrated. Also note that $L_i(t) > L_i(t+1)$ since job J has left M_i. Let $L = \max\{L_i(t+1), L_j(t+1)\}$. Since $L < L_i(t)$ one can show that the new machine loads vector is smaller in the sorted lexicographic order than the old machine loads vector. This is summarized in the following claim.

Claim 1. *The sorted lexicographic order of the machine loads vector decreases when a job migrates.*

The above argument shows that any improvement policy converges to a Nash equilibrium, and gives us an upper bound on the convergence time equal to the number different sorted machine loads vectors (which is trivially bounded by the number of different system configurations).

General Weights. In the general case, the number of different system configurations is at most m^n, which derives the following corollary.

Corollary 1. *For any ESS strategy with an improvement policy, the system of multiple unrelated machines with restricted assignment reaches a Nash Equilibrium in at most m^n steps.*

Discrete Weights. For the discrete weight setting, the number of different weights is K. Let n_i be the number of jobs with weight w_i. The number of different configurations of jobs with weight w_i is bounded by $\binom{m+n_i}{m}$. Multiplying the number of configurations for the different weights bounds the number of different system configurations. Since, by definition, $\sum_{i=1}^{k} n_i = n$, we can derive the following.

Corollary 2. *For any ESS strategy with an improvement policy, the system of multiple unrelated machines with restricted assignment under the discrete weight setting reaches a Nash Equilibrium in at most*

$$\prod_{i=1}^{k} \binom{m+n_i}{m} \leq \left(c\frac{n}{Km} + c\right)^{Km},$$

steps for some constant $c > 0$.

Integer Weights. To bound the convergence time for the integer weight setting, we introduce a potential function and demonstrate that it decreases when a job migrates. We define the potential of the system at time t, as $P(t) = \sum_{i=1}^{m} 4^{L_i(t)}$. After job J migrates from M_i to M_j then we have that $L_i(t) - 1 \geq L_j(t+1)$, since J migrated. Also, since we have integer weights, $L_i(t+1) \leq L_i(t) - 1$. Therefore, the reduction in the potential is at least,

$$P(t) - P(t+1) = 4^{L_i(t)} + 4^{L_j(t)} - [4^{L_i(t+1)} + 4^{L_j(t+1)}] \geq 4^{L_i(t)}/2 \geq 2. \quad (1)$$

Since in the initial configuration we have that $P(0) \leq 4^W$ we derive the following theorem.

Theorem 1. *For any ESS strategy with an improvement policy, the system of multiple machines under the integer weight setting reaches a Nash Equilibrium in* $4^W/2$ *steps.*

Next we show that this bound can be reduced to $O(mW + m4^{W/m+w_{max}})$ when using the Max Load Machine strategy.

Theorem 2. *For Max Load Machine strategy with an improvement policy, the system of multiple machines under the integer weight setting reaches a Nash Equilibrium in at most* $4mW + m4^{W/m+w_{max}}/2$ *steps.*

Proof. We divide the schedule into two phases with respect to the maximum load among the machines. The first phase continues until $L_{max}(t) \leq W/m + w_{max}$, and then the second phase starts. At the start of the second phase, at time T, the potential is at most $m4^{L_{max}(T)} \leq m4^{W/m+w_{max}}$. By (1), at every step the potential drops by at least two, therefore the length of the second phase is bounded by $m4^{W/m+w_{max}}/2$. Thus, it remains to bound the length of the first phase, namely T. At any time $t < T$ we have $L_{max}(t) > W/m + w_{max}$, which implies that $L_{min}(t) \leq W/m$. Therefore every job in the maximum loaded machine can benefit by migrating to the least loaded machine. The Max Load Machine strategy will choose one of those jobs. By (1), the decrease in the potential is at least $4^{L_{max}(t)}/2 \geq P(t)/2m$. Therefore, after T steps we have $P(T) \leq P(0)(1 - 1/2m)^T$. Since $P(0) \leq 4^W$ and $P(T) \geq 1$, it follows that $T \leq 4mW$, which establishes the theorem. □

Two Weights. It is worth to note that for the special case of two different weights there exists an efficient ESS strategy the converges in linear time (proof omitted).

4 Related Machines

In this section we consider the related machines. (The omitted proofs can be found in Appendix A.) We first consider restricted assignments and assume that all jobs follow an *improvement* policy. We define the potential of the system as follows:

$$P(t) = \sum_{i=1}^{m} \frac{(L_i(t))^2}{S_i} + \sum_{j=1}^{n} \frac{w_j^2}{S_{M(j,t)}} = \sum_{i=1}^{m} S_i (T_i(t))^2 + \sum_{j=1}^{n} \frac{w_j^2}{S_{M(j,t)}}$$

The following Lemma shows that the potential drops after each improvement step.

Lemma 1. *When a job of size* w *migrates from machine* i *to machine* j *at time* t *then* $P(t+1) - P(t) = 2w(T_j(t+1) - T_i(t)) < 0$.

We now like to bound the drop in the potential in each step. Clearly, if we are interested in ϵ-Nash equilibrium, then the drop is at least $2w\epsilon > \epsilon$. Considering a Nash equilibrium, for integer weights and speeds the the drop is at least $(S_{max})^{-2}$. Since the initial potential is bounded by W^2/S_{min}, we can derive the following Theorem.

Theorem 3. *For any ESS strategy with an improvement policy, the system of multiple related machines with restricted assignment reaches an ϵ-Nash Equilibrium in at most $O(\frac{W^2}{\epsilon S_{min}})$ steps, and reaches a Nash Equilibrium, assuming both integer weights and speeds, in at most $O(W^2 \frac{S_{max}^2}{S_{min}})$ steps.*

For unrestricted assignment, by forcing to move the job from the most loaded machine we can improve the bound as follows.

Theorem 4. *Max Load Machine strategy with best reply policy reaches an ϵ-Nash Equilibrium in at most*

$$O(W\sqrt{\frac{mS_{max}}{S_{min}}} + \frac{nw_{max}^2}{\epsilon S_{min}})$$

steps.

Discrete Weights. We show that for any K integer weight there is an equivalent model in which w_{max} is bounded by $O(K(S_{max}n)^{4K})$, and therefore $W = O(Kn(S_{max}n)^{4K})$. This allows us to translate the results using W to the discrete weight model by replacing W by $O(Kn(S_{max}n)^{4K})$. (We do not need to calculate the equivalent weights, since they are only used for the convergence time analysis.) We first define what we mean by an equivalent set of weights.

Definition 1. *Two discrete set of weights $w_1, \ldots, w_K$ and $\alpha_1, \ldots, \alpha_K$ are equivalent if for any two assignments, $n_1, \ldots, n_K$ and $\ell_1, \ldots, \ell_K$ we have $\sum_{i=1}^K n_i w_i > \sum_{i=1}^K \ell_i w_i$ if and only if $\sum_{i=1}^K n_i \alpha_i > \sum_{i=1}^K \ell_i \alpha_i$, and $\sum_{i=1}^K n_i w_i = \sum_{i=1}^K \ell_i w_i$ if and only if $\sum_{i=1}^K n_i \alpha_i = \sum_{i=1}^K \ell_i \alpha_i$. (We require that both $\sum_{i=1}^K n_i \leq n$ and $\sum_{i=1}^K \ell_i \leq n$.)*

Intuitively, the above definition says that as long as we use only comparisons, we can replace $w_1, \ldots, w_K$ by $\alpha_1, \ldots, \alpha_K$. Most important for us is that we can use in the potential the α's rather than the w's. From the definition of an equivalent set of weights we can derive the following. Any strategy based on comparisons of job weights and machine loads and an improvement policy based on comparisons of machine loads (e.g. best reply) would produce the same sequence of job migrations starting from any initial configuration.

The following theorem, which is proven using standard linear integer programming techniques, bounds the size of the equivalent weights. (The linear program and and a sketch of the proof is in Appendix C.)

Theorem 5. *For any discrete set of weights $w_1, \ldots, w_K$ there exist an equivalent set of weights $\alpha_1, \ldots, \alpha_K$ such that $\alpha_K \leq K(cS_{max}n)^{4K}$ for some constant $c > 0$.*

Unit Weight Jobs. We show that for unit weight jobs, there exists a strategy that converges in mn steps. The unit weight jobs is a special case of [21] with a symmetric cost function, where was derived an upper bound of $O(mn^2)$ on the convergence time of a specific strategy. We follow the proof of [21] and obtain a better bound in our model.

Theorem 6. *There exists an ESS strategy with an improvement policy such that the system of multiple related machines with restricted assignment reaches a Nash Equilibrium in at most mn steps in the case of unit weight jobs.*

The next theorem presents a lower bound of $\Omega(mn)$ on the convergence time of some ESS strategy (different from that of Theorem 6).

Theorem 7. *There exists an ESS strategy with an improvement policy such that for the system of multiple related machines with unrestricted assignment, there exists a system configuration that requires at least $\Omega(mn)$ steps to reach a Nash Equilibrium in the case of unit weight jobs.*

5 Identical Machines

In this section we will show improved upper bounds that apply to identical machines with unrestricted assignment. We also show a lower bound for K weights which is exponential in K. The lower bound is presented for the Min Weight Job policy. Clearly, this lower bound also implies a lower bound in all the other models. First we derive some general properties. The next observation states the minimal load cannot decrease.

Observation 1. *At every time step the minimal load among the machines either remains the same or increases.*

Now we show that when a job moves to a new machine, this machine still remains a minimal marginal load machine for all jobs at that machine which have greater weight.

Observation 2. *If job J has migrated to its best response machine M_i at time t then M_i is a minimal marginal load machine with regard to any job $J' \in B_i(t)$ such that $w(J') \geq w(J)$.*

Next we show that once a job has migrated to a new machine, it will not leave it unless a larger job arrives.

Claim 2. *Suppose that job J has migrated to machine M at time t. If $J \in A(t')$ for $t' > t$ then another job J' such that $w(J') > w(J)$ switched to M at time t'', and $t < t'' \leq t'$.*

Proof. Since M was the best response machine for J it implies that is the minimal marginal load machine at time t with respect to J. By observation 1, the minimal load never decreases. Thus, the only reason that J wishes to switch machine is that another job(s) arrived at M. By Observation 2 arrival of a smaller weight job will maintain M as the minimal marginal load machine with respect to J. Therefore, it must be the case that at least one job of weight greater than that of J joined M between $t + 1$ and t'. $\square$

Next we present an upper bound on the convergence time of Max Weight Job strategy. (A similar claim (without proof) appears in [27].)

Theorem 8. *The Max Weight Job strategy with best response policy, for the system of multiple identical machines with unrestricted assignment reaches a Nash Equilibrium in at most n steps.*

Proof. By Claim 2, once the job has migrated to a new machine, it will not leave it unless a larger job arrives. But under Max Weight Job strategy only smaller jobs can arrive in the subsequent time steps, so each job stabilizes after the first migration, and the theorem follows. □

Now we present a lower bound for the Min Weight Job strategy.

Theorem 9. *For the Min Weight Job strategy with best response policy, for the system of multiple identical machines with unrestricted assignment, there exists a system configuration that requires at least $(\frac{n}{K})^K/(2(K!))$ steps to reach a Nash Equilibrium, where $K = m - 1$.*

We also present a lower bound of $n^2/4$ on the convergence time of Min Weight Job and FIFO strategies for the case of two machines.

Theorem 10. *For the Min Weight Job and FIFO strategies with best response policy, for the system of two identical machines with unrestricted assignment, there exists a system configuration that requires at least $n^2/4$ steps to reach a Nash Equilibrium.*

Proof. Consider the following scenario. There are $n/2$ classes of jobs $C_1, \ldots, C_{n/2}$ and each class contains exactly 2 jobs and has weight $w_i = 3^{i-1}$. Notice that a job in C_i with weight $w_i = 3^{i-1}$ has weight equal to the total weight of all the jobs in the first $i - 1$ classes plus 1.

Initially, all jobs are located at the same machine. We divide the schedule into *phases*. Let C_j^i we denote all jobs from classes $C_j, \ldots, C_i$. A *k-phase* is defined as follows. Initially, all jobs from classes C_1^k are located at one machine. During the phase these jobs, except one job from C_k, migrate to the other machine. Thus, the duration of a k-phase is $2k - 1$. It is easy to see that the schedule consists of the phases $n/2, \ldots, 1$ for Min Weight Job strategy. One can observe that FIFO can generate the same schedule, if ties are broken using minimal weight. □

The following theorem shows a tight upped bound of $\Theta(n^2)$ on the convergence time of FIFO strategy.

Theorem 11. *For FIFO strategy with best response policy, the system of multiple identical machines with unrestricted assignment reaches a Nash Equilibrium in at most $n(n+1)/2$ steps.*

Similarly to FIFO, we bound the expected convergence tome of Random strategy by $O(n^2)$.

Theorem 12. *For Random strategy with best response policy, the system of multiple identical machines with unrestricted assignment reaches a Nash Equilibrium in expected time of at most $n(n+1)/2$ steps.*

Discrete Weights. For the discrete weight case, we demonstrate an upper bound of $O((n/K+1)^K)$ on the convergence time of any ESS strategy, showing that the bound of Theorem 9 for the Min Weight Job is not far from the worst convergence time.

Theorem 13. *For any ESS strategy with best response policy, the system of multiple identical machines with unrestricted assignment under the discrete weight setting reaches a Nash Equilibrium in $O((n/K+1)^K)$ steps.*

Integer Weights. For the integer weight case, we show that the convergence time of any ESS strategy is proportional to the sum of weights.

Theorem 14. *For any ESS strategy with best response policy, the system of multiple identical machines with unrestricted assignment under the integer weight setting reaches a Nash Equilibrium in $W+n$ steps.*

Unit Weight Jobs. For the unit weight jobs, we present a lower bound on the convergence time of a specific strategy.

Theorem 15. *There exists an ESS strategy with the improvement policy for which the worst case number of steps for the system of multiple identical machines with unrestricted assignment and unit weight jobs to reach a Nash Equilibrium is at least $\Omega(\min\{mn, n\log n \frac{\log m}{\log\log n}\})$ steps.*

6 Concluding Remarks

In this paper we have studied the online load balancing problem that involves selfish jobs (users). We have focused on the number of steps required to reach a Nash Equilibrium and established the convergence time for different strategies. While some strategies provably converge in polynomial time, for the others the convergence time might require an exponential number steps.

In the real world, the convergence time is of high importance, since even if the system starts operation at a Nash Equilibrium, the users may join or leave dynamically. Thus, when designing distributed control algorithms for systems like the Internet, the convergence time should be taken into account.

References

1. E. Altman, T. Basar, T. Jimenez and N. Shimkin, "Routing into two parallel links: Game-Theoretic Distributed Algorithms," *Special Issue of Journal of Parallel and Distributed Computing on "Routing in Computer and Communication Networks"*, pp. 1367-1381, Vol. 61, No. 9, September 1, 2001.

2. B. Awerbuh, Y. Azar, and Y. Richter, "Analysis of worse case Nash equilibria for restricted assignment," unpublished manuscript.
3. Y. Azar, "On-line Load Balancing Online Algorithms - The State of the Art," chapter 8, 178-195, Springer, 1998.
4. Borosh and L. Treybis. "Bounds on positive integral solutions of linear Diophantine equations," *Proc. Amer. Math Soc.*, 55:299–304, 1976.
5. T. Boulogne, E. Altman and O. Pourtallier, "On the convergence to Nash equilibrium in problems of distributed computing," *Annals of Operation research*, 2002.
6. P. Brucker, J. Hurink, and F. Werner, "Improving Local Search Heuristics for Some Scheduling Problems, Part I," *Discrete Applied Mathematics*, 65, pp. 97 - 122, 1996.
7. P. Brucker, P.; J. Hurink, and F. Werner, "Improving Local Search Heuristics for Some Scheduling Problems, Part II," *Discrete Applied Mathematics*, 72, pp. 47 - 69, 1997.
8. A. Czumaj, P. Krysta and B. Vocking, "Selfish traffic allocation for server farms," STOC 2002.
9. A. Czumaj and B. Vocking, "Tight bounds on worse case equilibria," SODA 2002.
10. Florian, M. and D. Hearn, "Network Equilibrium Models and Algorithms", *Network Routing. Handbooks in RO and MS*, M.O. Ball et al. Editors, Elsevier, pp. 485-550. 1995.
11. D. Fudenberg and D. Levine, "The theory of learning in games," *MIT Press*, 1998.
12. D. Fotakis, S. Kontogiannis, E. Koutsoupias, M. Mavronicolas, and P. Spirakis, "The Structure and Complexity of Nash Equilibria for a Selfish Routing Game," *In Proceedings of the 29th ICALP*, Malaga, Spain, July 2002.
13. J.E. Hopcroft and J.D. Ullman, Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, 1979.
14. M. Kearns, M. Littman and S. Singh, "Graphical models for game theory," *In Proceedings of UAI*, 2001.
15. M. Kearns and Y. Mansour, " Efficient Nash Computation in Large Population Games with Bounded Influence," To appear *In Proceedings of UAI*, 2002.
16. Y. A. Korilis and A. A. Lazar, "On the Existence of Equilibria in Noncooperative Optimal Flow Control," *Journal of the ACM*, Vol. 42, pp. 584-613, 1995.
17. Y.A. Korilis, A.A. Lazar, and A. Orda. Architecting Noncooperative Networks. *IEEE J. on Selected Areas in Communications*, Vol. 13, pp. 1241-1251, 1995.
18. E. Koutsoupias, C. H. Papadimitriou, "Worst-case equilibria," *STACS 99*.
19. R. J. La and V. Anantharam, "Optimal Routing Control: Game Theoretic Approach," *Proceedings of the 36rd IEEE Conference on Decision and Control*, San Diego, CA, pp. 2910-2915, Dec. 1997.
20. M. Littman, M. Kearns, and S. Singh, "An efficient exact algorithm for singly connected graphical games," To appear *In Neural Information Processing Systems*, 2002.
21. I. Milchtaich, "Congestion Games with Player-Specific Payoff Functions," *Games and Economic Behavior*, vol. 13, pp. 111-124, 1996.
22. D. Monderer and L. S. Shapley, "Potential games," *Games and Economic Behavior*, 14, pp. 124-143, 1996.
23. J. F. Nash, "Non-cooperative games," *Annals of Mathematics*, Vol. 54, pp. 286-295, 1951.
24. A. Orda, N. Rom and N. Shimkin, "Competitive routing in multi-user communication networks," *IEEE/ACM Transaction on Networking*, Vol 1, pp. 614-627, 1993.
25. R. W. Rosenthal, "A class of games possessing pure-strategy Nash equilibria," *International Journal of Game Theory*, 2, pp. 65-67, 1973.

26. T. Roughgarden and E, Tardos, "How Bad is Selfish Routing?," *In the Proceedings of the 41st Annual IEEE Symposium on the Foundations of Computer Science*, 2000.
27. P. Schuurman and T. Vredeveld, "Performance guarantees of local search for multiprocessor scheduling," *Proceedings IPCO*, pp. 370-382, 2001.
28. S. Shenker, "Making greed work in networks a game-theoretic analysis of switch service disciplines," *IEEE/ACM Transactions on Networking*, Vol. 3, pp. 819-831, 1995.

A Proofs from Section 4

Proof (Lemma 1). Let $\Delta(P) = P(t+1) - P(t)$.

$$\begin{aligned}
\Delta(P) &= \frac{(L_j(t+1))^2}{S_j} + \frac{(L_i(t+1))^2}{S_i} + \frac{w^2}{S_j} - \frac{(L_i(t))^2}{S_i} - \frac{(L_j(t))^2}{S_j} - \frac{w^2}{S_i} \\
&= \frac{(L_j(t)+w)^2}{S_j} + \frac{(L_i(t)-w)^2}{S_i} + \frac{w^2}{S_j} - \frac{(L_i(t))^2}{S_i} - \frac{(L_j(t))^2}{S_j} - \frac{w^2}{S_i} \\
&= \frac{(L_j(t))^2 + 2wL_j(t) + w^2}{S_j} + \frac{(L_i(t))^2 - 2wL_i(t) + w^2}{S_i} + \frac{w^2}{S_j} \\
&\quad - \frac{(L_i(t))^2}{S_i} - \frac{(L_j(t))^2}{S_j} - \frac{w^2}{S_i} \\
&= 2w(\frac{L_j(t)+w}{S_j} - \frac{L_i(t)}{S_i}) = 2w\left(\frac{L_j(t+1)}{S_j} - \frac{L_i(t)}{S_i}\right) \\
&= 2w\left(T_j(t+1) - T_i(t)\right)
\end{aligned}$$

□

Proof (Theorem 4). Let $P(t) = P_1(t) + P_2(t)$, where , $P_1(t) = \sum_{i=1}^{n} \frac{(L_i(t))^2}{S_i}$ and $P_2(t) = \sum_{j=1}^{m} \frac{w_i^2}{S_{M(j,t)}}$. Let $T = \frac{W}{\sum_{i=1}^{m} S_i}$. We can rewrite the potential function as follows,

$$\begin{aligned}
P_1(t) &= \sum_{i=1}^{n} \frac{(L_i(t))^2}{S_i} = \sum_{i=1}^{n} (T_i(t))^2 S_i \\
&= \sum_{i=1}^{n} ((T - T_i(t)) + T)^2 S_i \\
&= \sum_{i=1}^{n} S_i T^2 + \sum_{i=1}^{n} S_i (T_i - T)^2 + 2\sum_{i=1}^{n} S_i (T - T_i) \\
&= \sum_{i=1}^{n} S_i T^2 + \sum_{i=1}^{n} S_i (T_i - T)^2,
\end{aligned}$$

where we used the fact that $\sum S_i T_i = W = \sum S_i T$. The first term, $\sum_{i=1}^{n} S_i T^2$ is constant and therefore can be ignored. We can rewrite the potential as:

$$P(t) = \sum_{i=1}^{n} S_i (T_i - T)^2 + \sum_{j=1}^{m} \frac{w_i^2}{S_{M(j,t)}}$$

and let $P_1(t) = \sum_{i=1}^{n} S_i (T_i - T)^2$. By Lemma 1 when a job of size w migrates from machine i to machine j at time t then $\Delta(P) = P(t) - P(t+1) = 2w(T_i(t) - T_j(t+1))$. We also define $\delta(t) = \max\left(T_{max}(t) - T, T - T_{min}(t)\right)$.

We divide the run of the algorithm to two phases. The first phase ends when either $\delta(t) < 2\frac{w_{max}}{S_{min}}$ or $P_1(t) < 4mw_{max}^2/S_{min}^2$ and then the second phase starts. During a time t in the first phase we have $P_1(t) = \sum_i (T_i - T)^2 S_{max} \le m\delta(t)^2 S_{max}$, which implies that $\sqrt{\frac{P_1(t)}{mS_{max}}} \le \delta(t)$. Since we have that $\Delta(P) = 2w(T_i(t) - T_j(t+1)) \ge 2w(\delta(t) - \frac{w_{max}}{S_{min}}) \ge \delta(t)$ we can obtain the following recursion

$$P_1(t+1) \le P_1(t) - \sqrt{\frac{P_1(t)}{mS_{max}}}.$$

Let $t_0 = 1$ and let t_i be the first time t at which $P_1(t) \le P_1(t_0)/2^i$. Since $P_1(t) \ge P_1(t_{i-1})/2$ for $t \in [t_{i-1}, t_i]$, we have that, $t_i \le t_{i-1} + \sqrt{mS_{max} P_1(t_{i-1})/2}$. When $P_1(t) < 4mw_{max}^2/S_{min}^2$ we clearly finished the first phase. This implies that the duration of the first phase is bounded by,

$$t_\ell \le \sum_{i=1}^{\ell} \sqrt{mS_{max} P_1(t_0)/2^i} = O(\sqrt{mS_{max} P_1(t_0)})$$

where $\ell = \log P_1(t_0) \le \log W^2 S_{max}$.

At the start of the second phase, at time T, we bound the potential as follows. Clearly, $P_2(T) \le n\frac{w_{max}^2}{S_{min}}$. Since we ended the first phase either $P_1(T)$ is bounded by $\frac{4mw_{max}^2}{S_{min}^2}$, or $\delta(T) < 2\frac{w_{max}}{S_{min}}$, which implies that $P_1(T)$ is bounded by $\frac{4mw_{max}^2}{S_{min}^2}$. This implies that

$$P(T) = O(m\frac{w_{max}^2}{S_{min}^2} + n\frac{w_{max}^2}{S_{min}}).$$

Since we can assume that $n \ge m$, and recall that $S_{min} \ge 1$, we obtain that

$$P(T) = O(n\frac{w_{max}^2}{S_{min}}).$$

Since we are interested in ϵ-Nash, the minimal improvement is at least ϵ, and we derive a $P(T)/\epsilon$ bound on the second phase. □

Proof (Theorem 7). Let $S_i = 1 + i/n$. The initial configuration has n unit weight jobs on M_1. The optimal solution assigns n/m jobs on each machine. Now consider the strategy that each time takes a job from M_1 and moves it through all the machines to the that with the least number of jobs. The number of steps $\sum_{i=1}^{n/m} \sum_{j=1}^{m} (m - j) = \Omega(nm)$. □

B Proofs from Section 5

Proof (Theorem 9). Consider the following scenario. There are $m-1$ classes of jobs $C_1, \ldots, C_{m-1}$ and each class contains $l = n/(m-1)$ jobs. The weights of the jobs are defined recursively: all jobs in class C_1 have weight 1 and all jobs in a higher class have weight larger than the total weight of all the jobs in the first $i-1$ classes, i.e. $w_k = l\sum_{i=1}^{k-1} w_i + 1$. Initially, all jobs of class $C_i (i = 1, \ldots, m-1)$ are located at machine M_{i+1}. We show that for Min Weight Job strategy it takes at least $l^{m-1}/(2((m-1)!))$ steps to converge. Notice that the strategy, the best response policy and the initial configuration determine uniquely the whole schedule.

We start with a few useful notations. We denote by C_j^i all the classes $C_j, \ldots, C_i$. Similarly, M_J^i denotes the machines $M_j, \ldots, M_i$.

We divide the schedule into *phases*. A *k-phase* is defined as follows. Initially, all the jobs from classes C_1^k are located at a machine M' and there is a set S of k machines (not including M') participating in the phase that contain equal number of jobs from any class higher than C_k and this number is less than or equal to the number of jobs of the corresponding class located on any machine not in S. During the phase all but $l/(k+1)$ jobs from C_k are balanced between the machines in S. We will demonstrate that the duration of a phase grows exponentially with k. First we need the following observation.

Definition 2. *For a set of machines S we define by $MIN_i(S)$ the subset of machines in S that contain the minimal number of jobs from C_i.*

Observation 3. *Suppose that a job from C_k is selected by Min Weight Job strategy. Let C_r be a class in $\{C_1^{k-1}\}$ and let J be a job from C_r. It must be the case that $M(J) \in MIN_{r+1}(MIN_{r+2}(\ldots(MIN_{m-1}(\{M_1^m\}))))$.*

The observation follows from the fact that no job from C_r wishes to change its machine and the weight of any job in a high class is greater than the total weight of all the jobs in the lower classes. Now we give a lower bound on the duration of a k-phase.

Lemma 2. *The duration of a k-phase is at least $l^k/(2(k!))$.*

Proof. The proof is by induction on the phase index.
Hypothesis. The duration of a k-phase is at least $l^k/(2(k!))$.
Basis $(k = 1)$. Let us consider 1-phase. Half of the jobs from C_1 migrate to another machine. Thus, the induction hypothesis trivially holds.
Step. Suppose that the induction hypothesis holds for all phases with index k' such that $k' < k$ and let us prove that it is also satisfied for a k-phase. By Observation 3, after at most $k-1$ migrations of jobs from C_k, all jobs form the lower classes are eventually concentrated at the same machine. Thus, every k'th moving job from C_k would initiate a $(k-1)$-phase. Hence, the number of $(k-1)$-phases initiated by the jobs from C_k is l/k. By the induction hypothesis

the duration of each $(k-1)$-phase is at least $l^{k-1}/(2((k-1)!))$. Therefore, the duration of a k-phase is at least

$$\frac{l}{k} \cdot \frac{l^{k-1}}{(2((k-1)!)} = \frac{l^k}{2(k!)}.$$

□

It is easy to see that every $(m-1)$'th moving job from C_{m-1} generates a new $(m-2)$-phase. The duration of the convergence period follows by Lemma 2 after substituting $m-1$ instead of k. □

Proof (Theorem 11). We define a round to be a maximal sequence of jobs that move in which no job is repeated twice. Consider a round R and let J be the maximum weight job that wishes to move at the beginning of R. It must be the case that J is selected by FIFO strategy during R. According to Claim 2, J will not move in any of the consequent rounds. Therefore, the duration of k'th round is at most $n-k+1$. Thus, the total convergence time is bounded by $n(n+1)/2$. □

Proof (Theorem 13). Suppose that we have K classes of jobs $C_1, \ldots, C_K$ with weights $w_1 < \ldots < w_K$. Notice that each job from K'th class moves at most once while the number of migrations of a job in a lower class is bounded by the number of job migrations in all the higher classes plus one.

Hence, the number of moves of a job is defined by the following recursion: $f(i) = \sum_{j=i+1}^{K}(f(j) \cdot |C_j|) + 1$, where $f(i)$ is the maximal number of moves of a job from class C_i. Notice that $f(K) = 1$. We argue that $f(i) = \prod_{j=i+1}^{K}(|C_j|+1)$ for $i < K$:

$$\begin{aligned} f(i) &= \sum_{j=i+1}^{K}(f(j) \cdot |C_j|) + 1 \\ &= f(i+1) \cdot |C_{i+1}| + \sum_{j=i+2}^{K}(f(j) \cdot |C_j|) + 1 \\ &= f(i+1) \cdot |C_{i+1}| + f(i+1) = f(i+1)(|C_{i+1}| + 1), \end{aligned}$$

then we can continue recursively with $f(i+1)$ and so forth. Thus, the total number of job moves is bounded by $\sum_{i=1}^{K-1}(|C_i| \prod_{j=i+1}^{K}(|C_j|+1)) + |C_K|$. One can show that this expression is maximized when $|C_i| = n/K$ for all $1 \le i \le K$ since $\sum_{i=1}^{K}|C_i| = n$. This derives an upper bound of $\sum_{i=1}^{K-1}(\frac{n}{K}\prod_{j=i+1}^{K}(n/K+1)) + n/K = O((n/K+1)^{K-1}n/K) = O((n/K+1)^K)$. □

Proof (Theorem 14). By Claim 2, once the job has switched to a new machine, it will not leave it unless a larger job arrives. Thus, all but one move of a job results from migrations of jobs with greater weight to its machine. Observe that when a job J moves to a machine M it can force other jobs with the total weight of at most $w(J)-1$ to migrate from M. Otherwise, some job would leave a minimal marginal load machine. Then these jobs, in their turn, may cause migrations of other jobs on their destination machines. We claim that the total number of recursive migrations due to J is bounded by $w(J)-1$. let us define the push-out potential of the set of migrating jobs S, $PO(S)=\sum_{J\in S}(w(J)-1)$. Initially, S consists of one job J and $PO(S)=w(J)-1$. Then when a job $J'\in S$ migrates we remove it from S and add jobs on its target machine that would move due to J'. Obviously, each migration decreases PO by at least one. Thus, the total number of migrations resulting from moves of all jobs is bounded by W and the theorem follows. □

Proof (Theorem 15). The initial configuration is as follows: n identical jobs on machine 1 none on the other machines. First we show for $m\leq\log(n)$ a strategy which requires at least $\frac{(m-1)n}{2}$ steps to reach Nash equilibrium. The strategy works as follows. First it moves $n/2$ jobs to machine M_2. Now recursively, considering machines M_2 to m we have $m-1$ machines and $n/2$ jobs (on machine 2). After balancing the $n/2$ jobs on machines M_2^m we reconsider machine 1. Now every machine has at least $n/(2(m-1))$ jobs. Therefore we can continue recursively with m machines and $n/2-n/(2(m-1))$ jobs.

We let $f(n,m)$ be the number of steps in which the algorithm reaches Nash Equilibria. Then we have $f(n,m)=\frac{n}{2}+f(n/2,m-1)+f(\frac{n}{2}-\frac{n}{2(m-1)},m)$. Next we prove by induction that that $f(n,m)\geq\frac{(m-1)n}{2}$. For the basis we have that $f(k,1)=0$ and since we have that $m\leq\ln(n)$ this is suffice for the basis. We assume that the induction holds for $k'\leq k$, $l'\leq l$ and prove for (l,k).

$$\begin{aligned}
f(l,k) &= \frac{l}{2}+f(\frac{l}{2},k-1)+f(\frac{l}{2}-\frac{l}{2(k-1)},k) \\
&= \frac{l}{2}+\frac{\frac{l}{2}(k-2)}{2}+\frac{(\frac{l}{2}-\frac{l}{2(k-1)})(k-1)}{2} \\
&= \frac{l}{2}+\frac{lk}{4}-\frac{l}{2}+\frac{lk}{4}-\frac{l}{4}-\frac{l}{4} \\
&= \frac{lk}{2}-\frac{l}{2}=\frac{l(k-1)}{2}
\end{aligned}$$

For the case where $m>\log(n)$. We consider the case where the strategy first balances the jobs on machines $M_1^{\log(n)}$, requiring $\Omega(n\log n)$ steps. Latter we consider $\log n$ independent problems, each has $m\log n$ machines and $n/\log n$ jobs, all starting on one machine. Each such level would require requiring $\Omega(n\log n)$ steps. There are $\log m/\log\log n$ such levels, therefore we have a lower bound of $\Omega(n\log(n)\frac{\log m}{\log\log n})$. □

C Equivalent Weights

Proof (Theorem 5).

Definition 1 defines equivalent weights. Let $\{w_1, \ldots, w_K\}$ be a set of K integer weights. An equivalent set of weights can also be viewed as follows: consider all possible assignments (of up to n) jobs from K different integer weights to a single machine. We can encode an assignment by $(n_1, \ldots, n_K)$, where $\sum_{i=1}^{K} n_i \leq n$, and the load on a machine is $L = \sum_{i=1}^{K} w_i n_i$.

The idea of equivalent weights is that rather of considering the exact load of each assignment, we are only interested in their relative load. Namely, we are only interested in comparing the load on two different machines. From definition 1 an equivalent set of weights is a set of weights, which keeps the relative order between every pair of assignments. Our aim is to write an linear integer program that will describe the constraints that for any two possible assignments the comparison between the loads is maintained.

The integer program for identical machines is defined as follows. Let $x_1, \ldots, x_K$ be the unknown (new) weights. For every two possible assignments $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K)$ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_K)$, such that $\sum_{i=1}^{K} \alpha_i \leq n$ and $\sum_{i=1}^{K} \beta_i \leq n$, we generate an inequality in the integer linear program. The inequality compares $\sum_{i=1}^{K} \alpha_i x_i$ and $\sum_{i=1}^{K} \beta_i x_i$. To decide the result of the comparison we compare $\sum_{i=1}^{K} \alpha_i w_i$ to $\sum_{i=1}^{K} \beta_i w_i$. If they are equal we add the equation $\sum_{i=1}^{K} (\alpha_i - \beta_i) x_i = 0$. If $\sum_{i=1}^{K} \alpha_i x_i > \sum_{i=1}^{K} \beta_i x_i$ we add $\sum_{i=1}^{K} (\alpha_i - \beta_i) x_i > 0$. Otherwise, we add $\sum_{i=1}^{K} (\alpha_i - \beta_i) x_i < 0$. In addition, we require that the weights are positive, namely, for every i we have an inequality $x_i > 0$.

For the related machines we need to take into account their speeds as well. (We assume that the speeds are integers.) Rather than generating an inequality for each pair of assignments, we generate an inequality for each pair of assignments and machines. For instance, if the assignments are $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ and the machines are M_1 and M_2 we compare $\sum_{i=1}^{K} S_2 \alpha_i w_i$ to $\sum_{i=1}^{K} S_1 \beta_i w_i$. The generated inequality would depend, as before, on the output of the comparison.

Let A be the matrix that represent the inequalities. For identical machines note that the sum of the absolute entries in each row is bounded by $O(n)$, and for related machines by $O(nS_{max})$. The main observation that we use to bound the magnitude of the solution is based on the following lemma.

Lemma 3. *If B is a square submatrix of A, then $|det(B)| \leq (cS_{max}n)^k$, for some constant $c \geq 1$.*

The above lemma, with standard integer programming arguments (see [13]) allows us to derive a bound of $K(cnS_{max})^{4K}$, for some constant $c \geq 1$, on the maximum weight in the equivalent set of weights. □