Convergence Time to Nash Equilibrium in Load Balancing

EYAL EVEN-DAR
University of Pennsylvania, Philadelphia, PA
ALEX KESSELMAN
Max-Planck-Institut fur Informatik, Saarbrucken, Germany
and
YISHAY MANSOUR
Tel-Aviv University, Tel-Aviv, Israel

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). A by product of our results is establishing a connection between the various scheduling models and the game theoretic notion of potential games. We show that load balancing in unrelated machines is a generalized ordinal potential game, load balancing in related machines is a weighted potential game, and load balancing in related machines and unit weight jobs is an exact potential game.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problems Complexity**]: Nonnumerical Algorithms and Problems

General Terms: Algorithms, Theory, Game Theory

Additional Key Words and Phrases: Nash Equilibrium, Convergence Time

A preliminary version of this work appeared in the *30th International Conference on Automata, Languages and Programming (ICALP)*, 2003, pp. 502-513. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778, by a grants no. 525/00 and 1079/04 from the Israel Science Foundation and an IBM faculty award. This publication only reflects the authors' views.
Author's addresses: Eyal Even-Dar, Department of Computer and Information Science, University of Pennsylvania, e-mail: `evendar@seas.upenn.edu`. Alex Kesselman, Max-Planck-Institut fur Informatik, email: `akessel@mpi-sb.mpg.de`. Yishay Mansour, School of computer Science, Tel-Aviv University, e-mail: `mansour@cs.tau.ac.il`.
This work was done while E.E and A.K were graduate students at Tel-Aviv Univeristy.

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 [Nash 1951]. Users' strategy is at a Nash equilibrium if no user can gain by unilaterally deviating from its own policy. An interesting class of non-cooperative games, which is related to load balancing, is congestion games [Rosenthal 1973] and its equivalent model of potential games [Monderer and Shapley 1996]. In a potential game there is a potential function which maps the current state to a real number (in the load balancing scenario a state would include assignment of jobs to machines). We now consider deviations of a single player (job) and compare the change in the deviating player's utility (load) to the change in the potential function. In an exact potential game the changes are identical. In a weighted potential game the changes are related by a factor that depends only on the player. In an ordinal potential game the changes are in the same direction, while in a generalized potential game an increase in a player's utility implies an increase in the potential function (but not vice versa). Almost by definition, every potential game has a pure (deterministic) equilibrium, and by iteratively performing improvements of the player we can reach such an equilibrium. In this paper we focus on the load balancing problem and relate them to potential games. Although our model is a simplification of the internet model, it still captures several aspects of it and can represent simpler networks as pointed out in [Koutsoupias and Papadimitriou 1999; Mavronicolas and Spirakis 2001; Papadimitriou 2001; Czumaj and Vocking 2002].

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* [Koutsoupias and Papadimitriou 1999; Papadimitriou 2001] has received considerable attention in load balancing [Koutsoupias and Papadimitriou 1999; Mavronicolas and Spirakis 2001; Czumaj and Vocking 2002] and in other problems such as routing and facility location [Czumaj et al. 2002; Fotakis et al. 2002; Roughgarden and Tardos 2002; Fotakis et al. 2004; Awerbuch et al. 2005]. 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 a 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 [Fudenberg and Levine 1998]). 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 arises 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, [Azar 1998]). 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 can benefit from changing its machine. In our model, similarly to the Elementary Stepwise System (ESWS) (see [Orda et al. 1993]), 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. This model measures both the sequential complexity of reaching pure Nash equilibrium, and the complexity of distributed algorithms converging to Nash equilibrium in which only one movement is allowed at each time step.

In our notation a strategy is 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. Let K be the number of different weights, W be the sum of the weights of all the jobs, and w_{max} be the maximum weight assigned to any job (we normalize the minimal weight to 1).

For the general case of unrelated machines we show that the system always converges to a Nash equilibrium. We do it by introducing an order between the different configurations and showing that when a job migrates we move to a "lower" configuration in the order. This shows that this case is a generalized ordinal potential game. This implies that we never reach the same configuration twice, and therefore bounding the number of configurations bounds the convergence time. Bounding the number of configurations by $\min\{[O(\frac{n}{Km}+1)]^{Km}, m^n\}$ derives a general upper 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 the controller moves the minimum weight job, the convergence may take an exponential number of steps. Specifically, the number of steps is at least,

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

for $K = m - 1$. In contrast, we show that if the controller 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 derive an $O(n^2)$ bound on their convergence time.

For restricted assignment and related machines we show that it is a weighted potential game, and in the case that all the jobs have the same weight it is an exact potential game. For this setting we bound by $O\left(W^2 S_{max}^2/\epsilon\right)$ the convergence time to ϵ-Nash, where no job can benefit more than ϵ from unilaterally migrating to another machine and S_{max} is the maximal speed. 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 we have $\epsilon_{min} = 1$.

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', and it is only used in the convergence analysis.) 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 [1996] describes a class of non-cooperative games, which is related to load balancing. (In order to make the relations between the models clearer we 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 that 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. He shows 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 these games there are cases where a pure Nash equilibrium does not exist. In contrast, we show that any improvement policy converges to a pure Nash equilibrium in the load balancing setting.

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 [Brucker et al. 1996; 1997; Schuurman and Vredeveld 2001]. The approximation ratio of a jump (one job moves at a time) iterative improvement has been studied in [Finn and Horowitz 1979]. In [Brucker et al. 1997] it was shown that for two identical machines this heuristic 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 [Schuurman and Vredeveld 2001] they observe that the improvement strategy that moves the maximum weight job converges in n steps.

Goldberg [2004] studied a randomized model in which each user can select a random delay over continuous time. The continuous time implies that only one user tries to reroute at each specific time. In that model rerouting succeeds only if the user lowers its load. The work shows a simple randomized algorithm in which the expected number of rerouting attempts, until convergence to a Nash equilibrium, is polynomial in the number of links m and users n.

Mirrokni and Vetta [2004] studied a model in which only one user is allowed to move in each time step, but the main interest was not in the equilibrium point but on the social value after a short best response path and therefore they were interested in the convergence time to an approximate solution and to exact solution as we are.

A Nashification algorithm is an algorithm which changes the system state from an arbitrary state to Nash equilibrium without increasing the social cost. In [Feldmann et al. 2003] a Nashification algorithm in a Load balancing setting was considered. They studied the time it takes for a specific scheduler to Nashify a non Nash equilibrium state. They also provide an example where there can exists a sequence of an exponential number of selfish improvements with respect to the the number of machines to reach Nash equilibrium. (This last result is similar to Theorem 5.5 in this paper and was derived independently at the same time.)

Some interesting related learning models are stochastic fictitious play [Fudenberg and Levine 1998], graphical games [Littman et al. 2002], and large population games [Kearns and Mansour 2002]. Uniqueness of Nash equilibrium in communication networks with selfish users has been investigated in [Orda et al. 1993]. An analysis of the convergence to a Nash equilibrium in the limit appears in [Altman et al. 2001; Boulogne et al. 2002].

Following the proceeding version of this work, the work of [Even-Dar and Mansour 2005] studies also the convergence rate in load balancing scenario. The major difference there is that they consider a model without centralized unit and the model they consider is simpler, only related machines and mainly unweighed jobs. A logarithmic bound on the convergence time is derived there.

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 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 machine M_i. Let S_{min} and S_{max} denote the minimal and maximal speed, respectively. WLOG, we assume that $S_{min} = 1$. 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 $w_i(J) > 0$ 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.[1] 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 positive real numbers. *Discrete weight setting* – there are K different weights $1 = w_1 \leq \ldots \leq w_K = w_{max}$. *Integer weight setting* – the weights are positive 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 (ESWS) (see [Boulogne et al. 2002; Orda et al. 1993]). 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 reach a stable system state. This is evident in a simple example where there are two machines and in each time step all jobs behave selfishly and migrate to the least loaded machine.

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

[1] We do not consider mixed strategies, namely, every job J at time t is mapped to a unique machine $M(J,t)$.

minimizes its cost (after the migration), we call it a *best-reply* policy. Otherwise, we call it a *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. (Note that by definition, the only stable system states are Nash equilibria.) 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 ESWS 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. 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:. One can view this policy as using a queue, where new jobs that would like to migrate enter the end of the queue, and the job at the head of the queue is selected to migrate. (Note that jobs in the queue might change their status and would not like to migrate, in which case they leave the queue.) More formally, 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.

Potential Games: Monderer and Shapley [1996] defined few classes of potential games. Common to the various classes of potential games is a potential function P that maps a game state (system state in our notation) into the Reals. We give here the definition of the different classes of potential games using our notations, where the players are the jobs, their utility function is their load and their actions are selecting a machine. Let s_t be the system state at time t and P be the potential function.

Exact potential game. when a job migrates from M_i to M_j at time t then the following holds: $T_j(t+1) - T_i(t) = P(s_{t+1}) - P(s_t)$.

Weighted potential game. when job J migrates from M_i to M_j at time t then the following holds: $T_j(t+1) - T_i(t) = c(J)(P(s_{t+1}) - P(s_t))$, for some constant $c(J)$ which depends only on the job J.

Ordinal potential game. when a job migrates from M_i to M_j at time t then the following holds: $T_j(t+1) - T_i(t) < 0 \iff P(s_{t+1}) - P(s_t) < 0$.

Generalized ordinal potential game. when a job migrates from M_i to M_j at time t then the following holds: $T_j(t+1) - T_i(t) < 0 \Longrightarrow P(s_{t+1}) - P(s_t) < 0$.

The difference between a generalized ordinal potential game and an ordinal potential game is in the case that the normalized load of the migrating job does not change, i.e., $T_j(t+1) = T_i(t)$. An ordinal potential requires that in such a case we have $P(s_{t+1}) = P(s_t)$, while for a generalized ordinal potential this is not a requirement, and any relationship can hold between $P(s_{t+1})$ and $P(s_t)$. For all

classes of potential games, it is easy to observe that any improvement policy would converge to a pure equilibrium, since the potential function is bounded.

3. UNRELATED MACHINES

In this section we consider the unrelated machines case with the restricted assignment. We show that this case is a generalized ordinal potential game.

To show convergence we use a lexicographic order similar to the one used by Fotakis et al. [2002]. 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. We first observe 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 3.1. The sorted lexicographic order of the machine loads vector decreases when a job migrates.

We can use the lexicographic order to define a potential function (since it defines a complete order between system states). Therefore we establish the following.

COROLLARY 3.2. *Load balancing of jobs with unrelated machines is a generalized ordinal potential game.*

Note that in the general model, a job that migrates without changing its load might change the system state and therefore the potential function is not an ordinal potential function.

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

COROLLARY 3.3. *For any ESWS 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 3.4. *For any ESWS 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*

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

steps for some constant $c > 0$.

Integer Weights. To bound the convergence time for the integer weight setting, we introduce a generalized ordinal potential function and demonstrate that it decreases when a job migrates. We define the potential of the system at time t, as $P^{un}(t) = \sum_{i=1}^{m} 4^{L_i(t)}$. 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,

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

This establishes that P^{un} is a generalized ordinal potential function.

THEOREM 3.5. *For the system of multiple unrelated machines under the integer weight setting, the function $P^{un}(t) = \sum_{i=1}^{m} 4^{L_i(t)}$ is a generalized ordinal potential function.*

Since in the initial configuration we have that $P^{un}(0) \leq 4^W$ and when we terminate we have $P^{un}(T) \geq m$, we derive the following theorem.

THEOREM 3.6. *For any ESWS 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 3.7. *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. Note that in a second phase all jobs on the most loaded machine might want to stay on it. 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 duration of the second phase is bounded by $m4^{W/m+w_{max}}/2$. Thus, it remains to bound the duration of the first phase, namely T. At any time $t < T$ we have $L_{max}(t) > W/m + w_{max}$. Since $L_{min}(t) \leq W/m$, every job in the maximal 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 $4^{L_{max}(t)}/2 \geq P^{un}(t)/2m$.

Therefore, after T steps we have $P^{un}(T) \leq P^{un}(0)(1-1/2m)^T$. Since $P^{un}(0) \leq 4^W$ and $P^{un}(T) \geq 1$, it follows that $T \leq 4mW$, which establishes the theorem. □

4. RELATED MACHINES

In this section we consider related machines. We show for this case that there is a weighted potential function, where the factor of job J is simply its weight $w(J)$. (Most of the proofs of this section are deferred to Appendix A.)

We first consider restricted assignments and assume that all jobs follow an *improvement* policy. We first define the potential of the system as follows:

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

The following lemma shows that the game is a *weighted potential game*. Since the factor of job J, $c(J)$, is identical to twice its weight $w(J)$, if all the jobs have unit weight then the game is an exact potential game (with potential function $\sum_{i=1}^{m} \frac{(L_i(t))^2}{2S_i} + \sum_{j=1}^{n} \frac{w_j^2}{2S_{M(j,t)}}$).

LEMMA 4.1. *When a job of size w migrates from machine i to machine j at time t then $P^{rl}(t+1) - P^{rl}(t) = 2w(T_j(t+1) - T_i(t)) < 0$.*

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 then the drop is at least $(S_{max})^{-2}$. Since the initial potential is bounded by W^2, we can derive the following Theorem.

THEOREM 4.2. *For any ESWS strategy with an improvement policy, the system of multiple related machines with restricted assignment reaches an ϵ-Nash equilibrium in at most $O(W^2/\epsilon)$ steps, and reaches a Nash equilibrium, assuming both integer weights and speeds, in at most $O(W^2 S_{max}^2)$ steps.*

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

THEOREM 4.3. *For Max Load Machine strategy with best reply policy, the system of multiple related machines with restricted assignment reaches an ϵ-Nash equilibrium in at most*

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

steps.

Discrete Weights. We show that for any K integer weights 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 4.4. 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 $\ell_1, \ldots, \ell_K$, where $\sum_{i=1}^{K} n_i \leq n$ and

$\sum_{i=1}^{K} \ell_i \leq n$, 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 $\sum_{i=1}^{K} n_i \alpha_i = \sum_{i=1}^{K} \ell_i \alpha_i$.

Intuitively, the above definition implies 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 linear integer programming techniques, bounds the size of the equivalent weights. (The linear program and the proof can be found in Appendix C.)

THEOREM 4.5. *For any discrete set of integer weights $w_1, \ldots, w_K$ there exist an equivalent set of integer weights $\alpha_1, \ldots, \alpha_K$ such that for every i $\alpha_i \leq K(cS_{max}n)^{4K}$ 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 [Milchtaich 1996] with a symmetric cost function, and an upper bound of $O(mn^2)$ on the convergence time of a specific strategy was derived. We follow the proof of [Milchtaich 1996] and obtain a better bound in our model. (The proof is in Appendix A.)

THEOREM 4.6. *There exists an ESWS strategy 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 ESWS strategy (different from that of Theorem 4.6).

THEOREM 4.7. *There exists an ESWS 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 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. (Most of the proofs of this section are deferred to Appendix B). First we derive some general properties. The next observation states the minimal load cannot decrease.

Observation 5.1. For a system of multiple identical machines with unrestricted assignment, 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 5.2. For a system of multiple identical machines with unrestricted assignment, 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 5.3. For a system of multiple identical machines with unrestricted assignment, suppose that job J has migrated to its best response machine M at time t. If $J \in A(t')$ for $t' > t$ then another job J' such that $w(J') > w(J)$ migrated to machine M at time t'', and $t < t'' \leq t'$.

PROOF. Since M was the best response machine for J it implies that M is the minimal marginal load machine at time t with respect to J. By observation 5.1, the minimal load never decreases. Thus, the only reason that J wishes to migrate from machine M is that another job(s) migrated to machine M. By Observation 5.2 arrival of a smaller or equal weight job maintains 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 migrated to M between $t+1$ and t'. □

Next we present an upper bound on the convergence time of Max Weight Job strategy. (A similar claim (without proof) appears in [Schuurman and Vredeveld 2001].)

THEOREM 5.4. *For the Max Weight Job strategy with best response policy, the system of multiple identical machines with unrestricted assignment reaches a Nash equilibrium in at most n steps.*

PROOF. By Claim 5.3, once the job has migrated to a new machine, it will not leave it unless a larger job arrives. Since under the Max Weight Job strategy only smaller jobs can arrive in the subsequent time steps, it implies that each job stabilizes after the first migration, and the theorem follows. □

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

THEOREM 5.5. *For Min Weight Job strategy with best response policy, there exists a system configuration that requires at least* $(\frac{n}{K})^K/(2(K!)) \geq (n/K^2)^K$ *steps to the system of multiple identical machines with unrestricted assignment 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 5.6. *For the Min Weight Job and FIFO strategies with best response policy, there exists a system configuration that requires at least* $n^2/4$ *steps to the system of two identical machines with unrestricted assignment to reach a Nash equilibrium.*

PROOF. Consider the following scenario. There are $n/2$ classes of jobs $C_1, \ldots, C_{\frac{n}{2}}$ and each class contains exactly 2 jobs and has weight $w_i = 3^{i-1}$. Notice that a job in C_i has weight $w_i = 3^{i-1}$, which equals 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 denotes 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 5.7. *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 time of Random strategy by $O(n^2)$.

THEOREM 5.8. *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 ESWS strategy, showing that the bound of Theorem 5.5 for the Min Weight Job is not far from the worst convergence time.

THEOREM 5.9. *For any ESWS strategy with best response policy, the system of multiple identical machines with unrestricted assignment 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 ESWS strategy is proportional to the sum of weights.

THEOREM 5.10. *For any ESWS strategy with best response policy, the system of multiple identical machines with unrestricted assignment 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 5.11. *There exists an ESWS strategy with the improvement policy for which the worst case number of steps to 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, the convergence time should be taken into account.

A. PROOFS FROM SECTION 4

PROOF LEMMA 4.1. Let $\Delta(P^{rl}) = P^{rl}(t+1) - P^{rl}(t)$.

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

□

PROOF THEOREM 4.3. Let $P^{rl}(t) = P_1^{rl}(t) + P_2^{rl}(t)$, where , $P_1^{rl}(t) = \sum_{i=1}^{m} \frac{(L_i(t))^2}{S_i}$ and $P_2^{rl}(t) = \sum_{j=1}^{n} \frac{w_j^2}{S_{M(j,t)}}$. Let $T = \frac{W}{\sum_{i=1}^{m} S_i}$. We can rewrite the potential function as follows,

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

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

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

and redefine $P_1^{rl}(t) = \sum_{i=1}^{m} S_i (T_i(t) - T)^2$. By Lemma 4.1 when a job of size w migrates from machine i to machine j at time t then $\Delta(P^{rl}) = P^{rl}(t) - P^{rl}(t+1) = 2w(T_i(t) - T_j(t+1))$. We also define $\delta(t) = \max\{T_{max}(t) - T, T - T_{min}(t)\}$.

We divide the run of the algorithm to two phases. The first phase ends when either $\delta(t) < 2w_{max}$ or $P_1^{rl}(t) < 4mw_{max}^2$ and then the second phase starts. At time t in the first phase we have $P_1^{rl}(t) = \sum_i (T_i(t) - T)^2 S_i \leq m\delta(t)^2 S_{max}$, which implies

that $\sqrt{\frac{P_1^{rl}(t)}{mS_{max}}} \leq \delta(t)$. Since $\Delta(P^{rl}) = 2w(T_i(t) - T_j(t+1)) \geq 2w(\delta(t) - w_{max}) \geq \delta(t)$ we can obtain the following recurrence

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

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

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

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

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

$$P^{rl}(\tau) = O(mw_{max}^2 + nw_{max}^2).$$

Since we can assume that $n \geq m$, we obtain that

$$P^{rl}(\tau) = O(nw_{max}^2).$$

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

Proof Theorem 4.6. We first give the definition of a push out and pull in paths as defined in [Milchtaich 1996].

Definition A.1. A **Push out path** is a triplet (s, MM, JJ) where s is a system state, $MM = M_{i_0}, M_{i_1}, M_{i_2}, \ldots, M_{i_N}$ and $JJ = J_{i_1}, J_{i_2}, \ldots, J_{i_N}$ where the sequence of migration, starting at state s, and following it at step k having job J_{i_k} migrate to machine M_{i_k} from machine $M_{i_{k-1}}$ is a legal execution of an ESWS. Similarly, a **Pull in path** is a triplet (s, MM, JJ) where s is a system state, $MM = M_{i_1}, M_{i_2}, \ldots, M_{i_N}, M_{i_{N+1}}$ and $JJ = J_{i_1}, J_{i_2}, \ldots, J_{i_N}$ where the sequence of migration, starting at state s, and following it at step k having job J_{i_k} migrate from machine M_{i_k} to machine $M_{i_{k+1}}$ is a legal execution of an ESWS.

In contrast to the result in [Milchtaich 1996] we can bound the pull in path by m steps as well rather than nm, since the payoff is symmetric. The algorithm we consider is found in Algorithm 1.

Lemma A.2. *Each iteration of the loop (A - C) terminates after at most m steps.*

Proof. A is a single step. Since both in the pull in path and in the push out path no machine is repeated twice, we obtain the following, B is a push out path through U_1 thus it has at most $|U_1|$ steps. C is a pull in path in U_2 and thus it has at most $|U_2|$ steps. Since $|U_1| + |U_2| + 1 \leq m$ we conclude the Lemma. □

while $A(t) \neq \emptyset$ **do**

A Choose $J \in A(t)$. Let M be the machine with job J, U_1 be the set of machines with load smaller than that of M, and U_2 be the complement of U_1;

B Let J migrate to machine $M' \in U_1$ and continue with a push out path. (Namely, if some job J' now wants to migrate from M', add it to the push out path, and continue.);

C If there exists a job $I \in U_2$ that wants to migrate from M' to M, let it and follow a pull in path. (Namely, if some job I' now wants to migrate to M', add it to the pull in path, and continue.);

end

Algorithm 1: Push out - Pull in Algorithm

It still remains to show that in two times $t < \tilde{t}$ in which we are at start of an iteration, we have that $A(\tilde{t}) \subset A(t)$. We prove for any two consecutive times, t and t' where $t' < t$, in which at start of an iteration we have that $A(t') \subset A(t)$. We note that after performing steps A - C the load is changed only in two machines, the last machines in the pull in and the push out paths. Since in the rest of the machines the load remained unchanged, a job on them became unstable only if it wants to migrate to the last machine in the pull in path, but this is not possible. □

PROOF THEOREM 4.7. Let $S_i = 1 + i/n$. The initial configuration has n unit weight jobs on M_1. Note that the only Nash equilibrium 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 5.5. 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. $w_k = l \sum_{i=1}^{k-1} w_i + 1$. (For example we can set $w_k = \ell^k (k-1)! + k$.) 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 fixing a job selection strategy, the improvement policy to be the best response and the initial configuration, determine uniquely the entire 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 definition and observation.

Definition B.1. 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 B.2. Suppose that at time t 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 $M(J,t) \in MIN_{r+1}(MIN_{r+2}(\ldots(MIN_{m-1}(\{M_1^m\}))))$.

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 B.3. *The duration of a k-phase is at least $l^k/(2(k!))$.*

PROOF. The proof is by induction on the phase index.
Induction 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. After $k-1$ migrations of jobs from C_k, all jobs form the lower classes are concentrated at the same machine, since only one machine does not contain a job from class C_k, and by Observation B.2 it implies that the best response is unique.

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 B.3 after substituting $m-1$ instead of k. □

PROOF THEOREM 5.7. We define a round to be a maximal sequence of jobs that migrate in which no job is repeated twice. Consider a round R and let J be the maximum weight job that wishes to migrate at the beginning of R. There exists a time t during round R in which job J either was selected by FIFO and migrated or become stable. According to Claim 5.3, J will not migrate 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 5.8. We define round to be the sequence of jobs that migrate until the maximum weight job, that wants to migrate, migrates. Consider a round R and let J be the maximum weight job that wishes to migrate at the beginning of R. According to Claim 5.3, J will not migrate in any of the consequent rounds.

Therefore, there are at most $n-k+1$ jobs that want to migrate in the k'th round and using Random strategy its expected time is at most $n-k+1$. Thus, the expected convergence time is bounded by $n(n+1)/2$. □

PROOF THEOREM 5.9. 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 $\sum_{i=1}^{K-1}\left(|C_i|\prod_{j=i+1}^{K}(|C_j|+1)\right)+|C_K|$. Using Lagrange multipliers we obtain 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 5.10. By Claim 5.3, 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', but their total weight is less by at least least one than J' weight, thus PO decreases by at least one. Hence, the total number of migrations resulting from moves of all jobs is bounded by W and the theorem follows. □

PROOF THEOREM 5.11. The initial configuration is as follows: n identical jobs on machine 1 none on the other machines. First we show for $m \le \log(n)$ 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 equilibrium. 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

We first describe the intuition of Theorem 4.5. 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 4.4 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 and to use the fact that integer linear programming is NP.

PROOF THEOREM 4.5. The proof is done by using the fact the integer linear programming is in NP.

THEOREM C.1. *[Hopcroft and Ullman 1979] Let $Ax \geq b$, be an integer linear programming such that A is an $m \times n$ matrix of integers, b is a column of n integers. Then there exists a solution x such that $\|x\|_\infty \leq \ell(c\alpha)^{4\ell}$, where α is the magnitude of the largest element of A and b, $\ell = \min\{n,m\}$, and $c \geq 1$*

Since the proof can be found in the literature, we provide only the main technical lemma required for our setting and its proof here.

LEMMA C.2. *If B is a square submatrix of A, then $|det(B)| \leq (c\alpha)^\ell$, where c is some constant larger than one and $\ell = \min\{n,m\}$*

PROOF. Let B be a square submatrix of size k. If k is greater than ℓ, then the determinant is 0. Otherwise, the determinant is a sum of $k!$ term each is a product

of k elements. Now using the fact that $k \leq \ell$ and each element is bounded by α we obtain the lemma. □

Now we are ready to prove our theorem. Definition 4.4 defines equivalent set of weights. Let $\{w_1, \ldots, w_K\}$ be a set of K integer weights. We 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 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 $\vec{\alpha} = (\alpha_1, \ldots, \alpha_K)$ and $\vec{\beta} = (\beta_1, \ldots, \beta_K)$, 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 $\vec{\alpha}$ and $\vec{\beta}$ 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 the sum of the absolute entries in each row is bounded by $O(n)$, and for related machines by $O(nS_{max})$. Together with Theorem C.1, we conclude the proof. □

REFERENCES

Altman, E., Basar, T., Jimenez, T., and Shimkin, N. 2001. Routing into two parallel links: Game-theoretic distributed algorithms. *Journal of Parallel and Distributed Computing 61(9)*, 1367–1381.

Awerbuch, B., Azar, Y., and Epstein, A. 2005. The price of routing unsplittable flow. In *Proceedings of the 37th Symposium on Theory of Computing*. 57–66.

Azar, Y. 1998. *On-line Load Balancing Online Algorithms - The State of the Art*. Springer, Chapter 8, 178–195.

Boulogne, T., Altman, E., and Pourtallier, O. 2002. On the convergence to Nash equilibrium in problems of distributed computing. *Annals of Operation research 109(1-4)*, 279–291.

Brucker, P., Hurink, J., and Werner, F. 1996. Improving local search heuristics for some scheduling problems: Part I. *Discrete Applied Mathematics 65*, 97 – 122.

Brucker, P., Hurink, J., and Werner, F. 1997. Improving local search heuristics for some scheduling problems: Part II. *Discrete Applied Mathematics 72*, 47 – 69.

Czumaj, A., Krysta, P., and Vocking, B. 2002. Selfish traffic allocation for server farms. In *Proceedings of the 34th Symposium on Theory of Computing*. 287–296.

Czumaj, A. and Vocking, B. 2002. Tight bounds on worse case equilibria. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 413–420.

Even-Dar, E. and Mansour, Y. 2005. Fast convergence of selfish rerouting. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 772–781.

Feldmann, R., Gairing, M., Lcking, T., and B. Monien, M. R. 2003. Nashification and

the coordination ratio for a selfish routing game. In *Proceedings of the 30th International Colloquium on Automata, Languages and Programming (ICALP)*. 514–526.

Finn, G. and Horowitz, E. 1979. Linear time approximation algorithm for multiprocessor scheduling. *BIT 19(3)*, 312–320.

Fotakis, D., Kontogiannis, S., Koutsoupias, E., Mavronicolas, M., and Spirakis, P. 2002. The structure and complexity of Nash equilibria for a selfish routing game. In *Proceedings of the 29th ICALP*. 123–134.

Fotakis, D., Kontogiannis, S., and Spirakis, P. 2004. Selfish unsplittable flows. In *Proceedings of the 31th ICALP*. 593–605.

Fudenberg, D. and Levine, D. 1998. *The theory of learning in games*. MIT Press.

Goldberg, P. 2004. Bounds for the convergence rate of randomized local search in multiplayer games, uniform resource sharing game. In *In Proceedings of the Twenty-Third PODC*. 131–140.

Hopcroft, J. and Ullman, J. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

Kearns, M. and Mansour, Y. 2002. Efficient Nash computation in large population games with bounded influence. In *Proceedings of UAI*.

Koutsoupias, E. and Papadimitriou, C. H. 1999. Worst-case equilibria. In *Proceedings of 16th STACS*. 404–413.

Littman, M., Kearns, M., and Singh, S. 2002. An efficient exact algorithm for singly connected graphical games. In *Proceedings of NIPS*.

Mavronicolas, M. and Spirakis, P. 2001. The price of selfish routing. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing (STOC)*. 510 – 519.

Milchtaich, I. 1996. Congestion games with player-specific payoff functions. *Games and Economic Behavior 13*, 111–124.

Mirrokni, V. and Vetta, A. 2004. Convergence issues in competitive games. In *Proceedings of the 7th International Workshop on Approximation Algorithms for Combinatorial Optimization*.

Monderer, D. and Shapley, L. S. 1996. Potential games. *Games and Economic Behavior 14*, 124–143.

Nash, J. F. 1951. Non-cooperative games. *Annals of Mathematics 54*, 286–295.

Orda, A., Rom, N., and Shimkin, N. 1993. Competitive routing in multi-user communication networks. *IEEE/ACM Transaction on Networking 1*, 614–627.

Papadimitriou, C. H. 2001. Algorithms, games, and the internet. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing (STOC)*. 749–753.

Rosenthal, R. W. 1973. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory 2*, 65–67.

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

Schuurman, P. and Vredeveld, T. 2001. Performance guarantees of local search for multiprocessor scheduling. In *Proceedings IPCO*. 370–382.