Computational Game Theory **Spring Semester, 2011/2012**

Lecture 3: March 7

Lecturer: Amos Fiat *Scribe: Nir Shabat & Erez Shabat*

3.1 Inefficiency Of Equilbria States - Price of Anarchy

In the previous lectures we talked about the inefficiency of equilibrium states. We defined the concepts of the Price of Anarchy (PoA) and the Price of Stability (PoS). We mentioned two problems:

1. Infinitesimal Flows - For which we showed that an equilibrium state can be suboptimal. We talked of Brass Paradox, as a situation where adding another channel or route actually decreases the performance of the system.

2. Load Balancing - Where we showed lower and upper bounds on the PoA in the Identical Machines model, and a lower bound for the model of Related Machines.

3.1.1 Related Machines

We will now show an upper bound on the PoA for the Related Machines model. We will see that the Nash Equilibrium we used in Lecture 2 to achieve a lower bound, is actually as worse as the worst NE possible.

Theorem 3.1 *The Price Of Anarchy for the Related Machines Load Balancing problem with m machines is*

$$\mathcal{O}\left(\frac{\log m}{\log\log m}\right)$$

Proof We'll denote the makespan of some optimal assignment of the jobs as OPT. The following corollary is easily derived from this definition.

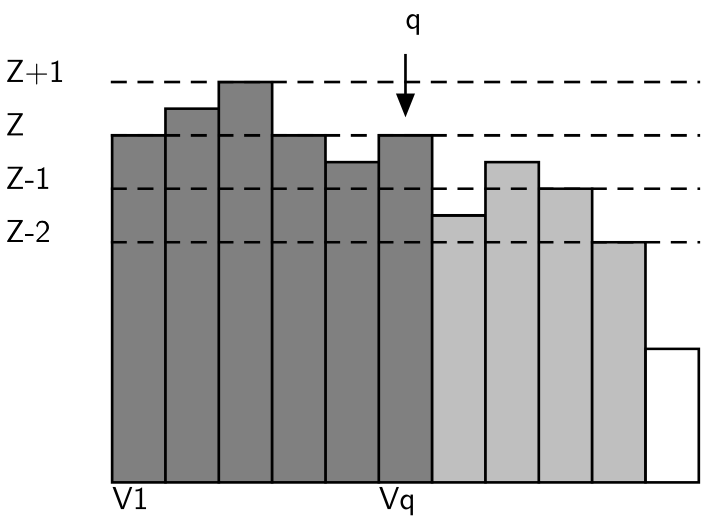


Figure 3.1: NE state for related machines

Corollary 3.2 *No job can take more than OPT processing time on the fastest machine.*

We will normalize the processing time of each job diving it by OPT, so that the makespan of an optimal solution is exactly 1.

Let A be an assignment of jobs to machines which is a NE, and let O be some optimal assignment (makespan wise) of jobs to the machines. We'll examine the list of machines sorted from left to right according to their speed, where V_1 is the fastest machine and V_m is the slowest one (see Figure 3.1). Let the load of the fastest machine V_1 in assignment A be Z.

Lemma 3.3 *No machine has a load greater than $Z + 1$ in assignment A.*

Proof For machine V_1 the lemma is true by the definition of Z. Let V_i be some other machine. Assume, toward a contradiction, that machine V_i has a load greater than $Z + 1$ in A, and let J be some job assigned to V_i. Since the load on V_1 is Z, and according to the corollary, J's processing time on V_1 is at most 1, moving job J to machine V_1 would reduce the cost of J. But this contradicts the assumption that A is a NE. Hence, no machine can have a load greater than $Z + 1$ in A.

The jobs assigned to V_1 in A are assigned to a subset (possible a set of 1 machines) of machines in assignment O.

Lemma 3.4 *Let V_q be the slowest machine which is assigned in O a job that is assigned in A to V_1. Then it must hold that:*

1. *$q \geq Z$*
2. *The loads of machines $V_1 \dots V_q$ in A is at least Z-1*

Proof Since the fastest machine V_1 has a load of Z in A, and the makespan of O is 1, there must be at least Z machines in O running the jobs assigned to V_1 by A. So it must hold that $q \geq Z$.

Assume, toward a contradiction, that the load on machine V_q in A is less than $Z - 1$. Since V_q in O is running some job J assigned to V_1 in A, J's processing time can't be more than 1 on machine V_q or any faster machine. Moving job J to any of the machines $V_2, \dots, V_q$ in A would reduce the cost of J. But this contradicts the assumption that A is a NE. Hence, the load on machines $V_2, \dots, V_q$ in A is at least $Z - 1$.

So far, we know that the total load in A is at least $Z \cdot (Z - 1)$. The jobs comprising that load must be assigned to at least $Z \cdot (Z - 1)$ machines in O. Using similar arguments to the ones used in Lemma 3.4, we can show that each of those $Z \cdot (Z - 1)$ machines have a load of at least $Z - 2$ in A, and so the total load of A is at least $Z \cdot (Z - 1) \cdot (Z - 2)$ and must be assigned to at least that number of machines in O, and so forth until we arrive at the fact that the number of machines m must be at least $Z!$, and by using Lemma 3.3 we get:

$$\text{makespan}(A) \leq Z + 1 = \mathcal{O}\left(\frac{\log m}{\log \log m}\right)$$

A more formal proof can be found in [1, p.524-527]

3.1.2 Other Models

Restricted Assignment

In this model of the load balancing problem, each job can only be assigned to a subset of the machines, i.e., its assignment is restricted.

Example Say there are 4 types of machines (denoted 0 - 3) and 3 types of jobs (denoted 0 - 2), and each machine incurs a load of 1 for each job assigned to it.

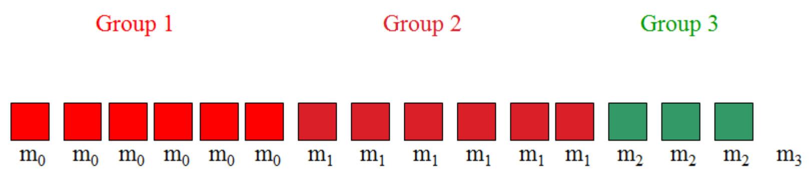


Figure 3.2: Optimal assignment for a load balancing problem with restricted assignment. Makespan of assignment is 1.

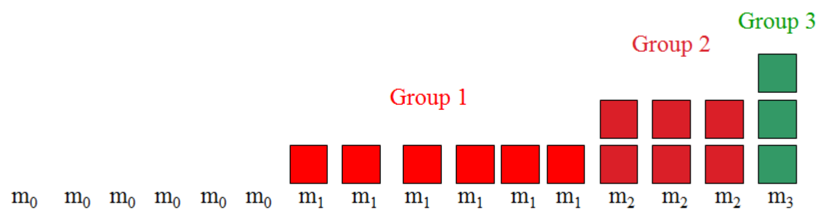


Figure 3.3: NE assignment for a load balancing problem with restricted assignment. Makespan of assignment is 3.

Also, assume that jobs of type x can only be assigned to machine of type $\geq x$, e.g., jobs of type 0 can be assigned to any machine, while jobs of type 2 can only be assigned to machines of type 2 and 3.

Let the input consist of 13 jobs (5 of type 0, 5 of type 1 and 3 of type 2) to be assigned to 13 machines (5 of type 0, 5 of type 1, 3 of type 2 and 3 of type 3).

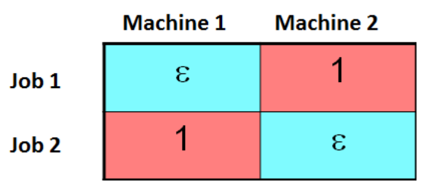
An optimal assignment for the above scenario is shown in figure 3.2, where a NE assignment is shown in figure 3.3.

This example gives a logarithmic lower bound on the PoA for the restricted assignment problem.

Unrelated Machines

Another model is the model of unrelated machines, where each job takes a different processing time on each machine (different load), yet the machines speed is unrelated (i.e. Machine X can be faster than machine Y for job 1, but slower than machine Y for job 2).

Example We have 2 jobs (denoted 1 and 2) and 2 machines (also denoted 1 and 2). Machine 1 can process job 1 in time ϵ and job 2 in time 1, and



	Machine 1	Machine 2
Job 1	ε	1
Job 2	1	ε

Figure 3.4: Unrelated Machine Cost Matrix Example

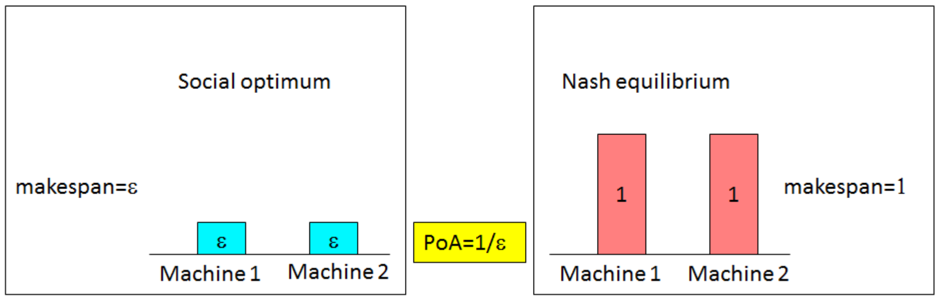


Figure 3.5: Optimal and NE assignment for a load balancing problem of unrelated machines

Machine 2 can process job 2 in time ϵ and job 1 in time 1 (where $\epsilon << 1$) - see figure 3.4.

An optimal assignment for the above case would have a makespan of ϵ, where a NE assignment can have a makespan of 1 (see figure 3.5).

In the above example the PoA is $\frac{1}{\epsilon}$, hence the PoA for the unrelated machines model is unbounded.

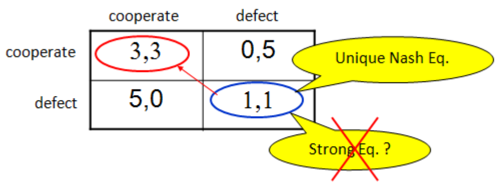
3.2 Inefficiency Of Equilbria States - Strong Price of Anarchy

Definition We say that a vector of strategies forms a *Strong Nash Equilibrium* if no coalition can deviate and **strictly** improve the utility of **all** its members. More formally, we'll say that in a strategy vector s, a subset A of players has a *joint deviation* if there is a strategy $s'_i \in S_i$ for $i \in A$ forming a vector s'_A, such that $u_i(s) < u_i(s'_A, s\_A)$ for all $i \in A$. A strategy vector s is a strong Nash equilibrium if no subset A has a joint deviation.

The strong Nash assumes the game has nontransferable utility, we need to make sure each one of the players in the coalition is increasing his utility. The strong Nash equilibria have a very strong reinforcing property, but in practice very few games have such equilibria.

We can also define the notion of **k-strong equilibrium** in the same way we've defined the Strong equilibrium, with one change, we bound the size of the subset of players A to k, i.e. $|A| \leq k$. From this definition we see that 1-strong equilibrium, is actually a NE. We also see that a k-strong equilibrium, is also a $(k-1)$-strong equilibrium, for every $k \geq 2$. The k-Strong equilibrium is useful when we believe that there is a limit to the size of players that can unite and form a coalition.

Example Prisoner's dilemma does not admit any Strong Eq.



3.2.1 Strong Price of Anarchy

Reminder:

$$\text{PoA} = \frac{\text{worst Nash equilibrium}}{\text{social optimum}}$$

Definition

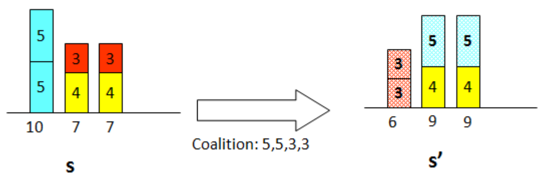
$$\text{SPoA} = \frac{\text{worst Strong equilibrium}}{\text{social optimum}}$$

Determining SPoA requires two parts:

- Proving existence of Strong equilibrium
- Bounding the worst ratio

Because every Strong equilibrium is also a Nash equilibrium, the following inequality holds: SPoA $\leq$ PoA

Example Not every Nash Eq. on identical machines is also a Strong Eq.



On the left side we have a Nash Eq. that is not a Strong Eq. because 5,5,3,3 can form a coalition and improve their state (reduce the load on the machines they're on)

3.2.2 Strong PoA in Related Machines

Definition A vector $(l_1, l_2, \ldots, l_m)$ is smaller than $(l'_1, l'_2, \ldots, l'_m)$ lexicographically if for some i, $l_i < l'_i$ and $l_k = l'_k$ for all $k < i$.

Definition A joint action s is smaller than s' lexicographically (denote $s \prec s'$) if the vector of machines loads L(s), sorted in non-decreasing order, is lexicographically smaller than L(s').

Lemma 3.5 *suppose $L(s)$ and $L(s')$ differ only in the loads of machines in a set $M' \subseteq M$. If for each $M_i \in M'$, $L_i(s) < max_k\{L_k(s')|M_k \in M'\}$, then $s \prec s'$. (The proof is pretty straightforward and can be found in [2, Lemma 3.2]*

Lemma 3.6 *The lexicographically minimal joint action s is a NE (This proof is also straightforward and can be found in [2, Lemma A.1])*

Theorem 3.7 *In any load balancing game, the lexicographical minimal joint action s is a k-Strong Eq. for any k (i.e. is a Strong Nash Eq.).*

Proof Using Lemma 3.6 we know that s is a NE. We'll assume by contradiction that s is not a Strong Eq. i.e. that there is a coalition that can deviate such that each member of the coalition strictly decreases its observed load. Denote the minimal sized coalition Γ and its size $k(k \leq n)$. Denote the joint action after the deviation s'. First we note that all jobs in Γ migrate to a different machine, otherwise, if there was a job that does not migrate, then even without it a coalition would have been formed, contradicting the minimality of Γ. We now prove that if a job wishes to leave machine m then another job wishes to migrate to m and vice-versa.

- First we consider the case that a job J migrates to some other machine M. In such, there must exist another job, say T, migrating out of M, otherwise s isn't a NE, since J can decrease its load by migrating to M by itself.

- Now we consider the second case where a job J migrates from machine m but no other job migrates to m. All other jobs could form a coalition without J and still strictly decrease their observed load, contradicting the minimality of Γ.

We'll look at all the jobs that deviate, only the machines they were on actually change their loads (because all jobs deviate inside that machine group), call that group of machines $M(\Gamma)$. For each machine m in $M(\Gamma)$, there's at least one job that wants to deviate to m. Since each job in the coalition benefits from the deviation, the new load on each machine m in $M(\Gamma)$ must strictly decrease, making $L_m(s') < max_k\{L_k(s)|M_k \in M(\Gamma)\}$. By Lemma 3.5, this implies that $s' \prec s$, contradicting the minimality of s.

3.2.3 Strong PoA in Unrelated Machines

Recall that the PoA for Unrelated Machines is unbounded. It turns out that for the Unrelated Machines model the SPoA cannot be higher than the number of machines m.

Let m denote the number of machines, and let $L_i\,(s)$ be the load of machine M_i in scheduling s. We denote by $c_J\,(s)$ the load observed by job J in scheduling s, and by s_J the machine job J is assigned to in scheduling s, i.e., $c_J\,(s) = L_j\,(s)$ where $M_j = s_J$. Also, denote by OPT the makespan of some optimal scheduling o.

For the rest of the discussion we will assume WLOG that given a scheduling s the machine indices are sorted in a non-decreasing order of the loads under s, i.e., $L_1 \leq \cdots \leq L_m$

Theorem 3.8 *For any job scheduling game with m unrelated machines and n jobs:*

$$SPoA \leq m$$

Proof Claim 3.9 *Let s be a Strong-NE assignment of jobs to machines, then:*

$$L_1\,(s) \leq OPT$$

Proof Assume toward contradiction that $L_1(s) > OPT$. In that case a coalition of all jobs can deviate to the optimal assignment o, where each job will strictly improve its position, in contradiction to the fact that s is a Strong-NE.

Claim 3.10 *Let s be a Strong-NE assignment of jobs to machines, then:*

$$\forall i L_i(s) - L_{i-1}(s) \leq OPT$$

Proof Assume toward contradiction that the claim is false and let j denote the smallest index for which the inequality above does not hold. Denote the set of jobs assigned to machines $M_m, \ldots, M_i$ by s by Γ. Consider a scheduling s' where $s'_J = o_J$ if $J \in \Gamma$ and $s'_J = s_J$ otherwise. Then for every $J \in \Gamma$ we have:

1. $c_J(s) > L_{j-1}(s) + \text{OPT}$ - According to our assumption.

2. $c_J(s') \leq L_{j-1}(s) + \text{OPT}$ - Since the jobs in Γ are scheduled in s' according to an optimal scheduling o, the load on machines $M_{j-1}, \ldots, M_1$ cannot increase by more than OPT.

That is, all jobs in Γ strictly improve their position in s', in contradiction to the fact that s is a Strong-NE.

From claims 3.9 and 3.10 we have that:

$$\text{makespan}(s) = L_m(s) \leq m \cdot \text{OPT}$$

And so:

$$\text{SPoA} = \frac{\text{makespan}(s)}{\text{OPT}} \leq \frac{m \cdot \text{OPT}}{\text{OPT}} = m$$

As stated.

Theorem 3.11 *There exists a job scheduling game of m unrelated machines for which:*

$$SPoA \geq m$$

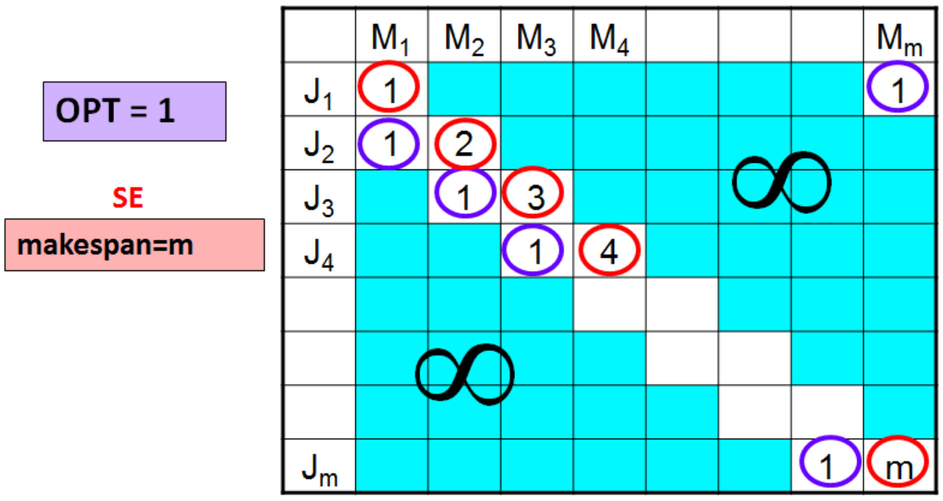


Figure 3.6: Optimal and Strong-NE assignments for a load balancing problem with unrelated machines, with ratio m

Proof Let (M, J, w) be an instance of a job scheduling game, where $M = \{M_1, \ldots, M_m\}$ is a set of unrelated machines, $J = \{J_1, \ldots, J_m\}$ a set of jobs and $w_i(j)$ the weight of job J_j on machine M_i:

$$w_i(j) = \begin{cases} 1 & \text{if } j \equiv (i+1) \pmod m \\ i & \text{if } i = j \\ \infty & \text{otherwise} \end{cases}$$

Define the following 2 schedulings (see figure 3.6):

1. o where $o_1 = M_m$ and $o_j = M_{j-1}$ for $i > 1$, so o has a makespan of 1.

2. s where $s_j = M_j$, so s has a makespan of m

Clearly, o is an optimal assignment, and s is a Strong-NE assignment, and so the theorem is proved.

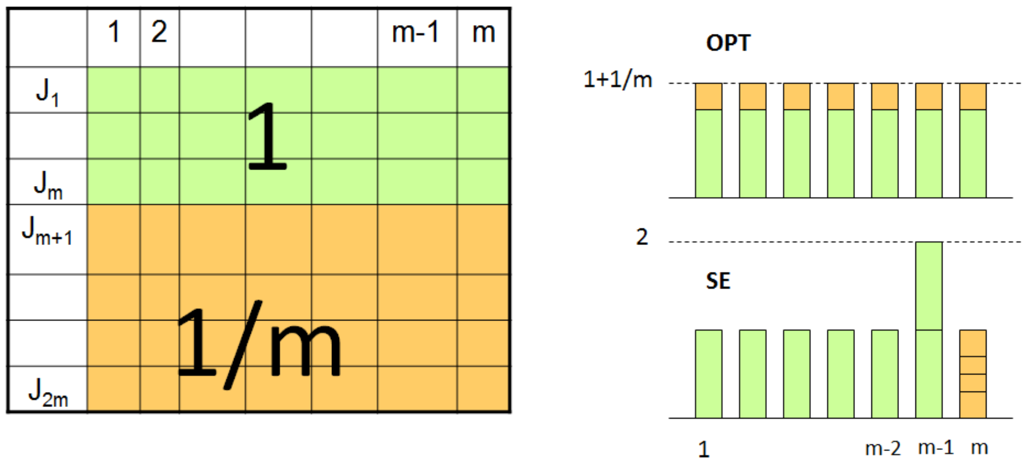


Figure 3.7: Optimal and Strong-NE assignments for a load balancing problem with identical machines

3.2.4 Strong PoA in Identical Machines

Theorem 3.12 *There exists a job scheduling game of m identical machines and n jobs, such that:*

$$SPoA \geq \frac{2}{1+\frac{1}{m}}$$

Proof Let (M, J, w) be an instance of a job scheduling game, where $M = \{M_1, \ldots, M_m\}$ is a set of identical machines, $J = \{J_1, \ldots, J_{2m}\}$ is a set of jobs and $w(j)$ the weight of job J_j:

$$w(j) = \begin{cases} 1 & \text{if } j \leq m \\ 1/m & \text{otherwise} \end{cases}$$

Define the following 2 schedulings (see figure 3.7):

1. o where $o_j = ((j-1)\,\%m) + 1$, so o has a makespan of $1 + 1/m$.
2. s where:

$$s_j = \begin{cases} M_j & \text{if } j \leq m-1 \\ M_{m-1} & \text{if } j = m \\ M_m & \text{otherwise} \end{cases}$$

so s has a makespan of 2.

Clearly, o is an optimal assignment, and s is a Strong-NE assignment, and so the theorem is proved.

Bibliography

[1] Noam Nisan , Tim Roughgarden , Eva Tardos , Vijay V. Vazirani *Algorithmic Game Theory*. Cambridge University Press, New York, 2007.

[2] Nir Andelman, Michael Fledman, Yishay Mansour *Strong Price of Anarchy*, 2006