CS 573: Algorithmic Game Theory	Lecture date: January 30th, 2008
Instructor: Chandra Chekuri	Scribe: Afsaneh Shirazi

Contents

1 Price of Anarchy (PoA) of Load Balancing on Related/Uniform Machines

In this section we assume that each machine has a different speed.

There are m machines available and n jobs controlled by n selfish players $1, \ldots, n$. Each machine has a different speed. α_j represents the speed of machine j and we sort the machines such that $\alpha_1 \geq \alpha_2 \geq \ldots \geq \alpha_m$. Player i controls job i with weight $w_i \geq 0$ which needs to be scheduled on one of the m machines. Define a load balancing game as $(N, \langle S_i \rangle, \langle u_i \rangle)$, where $N = \{1, \ldots, n\}$ denotes n players (jobs), $S_i = \{1, 2, \ldots, m\}$ is the set of possible strategies of player i, and $u_i : S = S_1 \times \ldots, \times S_n \mapsto \mathbb{R}$ is the payoff function of player i. In a vector of strategies $s \in S = S_1 \times \ldots \times S_n$, entry $s_i \in S_i$ means job i is scheduled on machines s_i.

The social objective is to minimize *makespan* (maximum machine load). For a strategy profile $s = (s_1, \ldots, s_m)$, load of machine j is the total weight of jobs scheduled on machine j divides by the speed of machine j. It is calculated from the equation bellow:

$$l_j(s) = \frac{1}{\alpha_j} \sum_{i:s_i=j} w_i$$

For a Nash equilibrium s if $s_i = j$ (i is on machine j) then

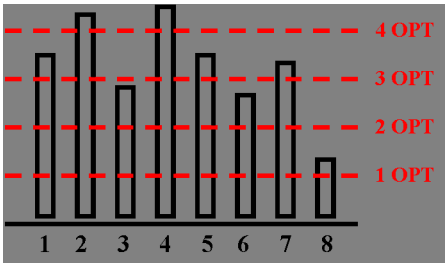
$$l_j(s) \leq l_{j'}(s) + \frac{w_i}{\alpha_{j'}} \qquad \forall j' \neq j$$

P

roof of Nash existence is the same as the proof for the identical machines case.

Theorem 1.1 *PoA for the load balancing game is* $O(\min(\frac{\lg m}{\lg \lg m}, \lg \frac{\alpha_1}{\alpha_m}))$. *Also, there exist instances (games) in which PoA is* $\Omega(\min(\frac{\lg m}{\lg \lg m}, \lg \frac{\alpha_1}{\alpha_m}))$

Proof: We prove the bound of $O(\lg m / \lg \lg m)$ and sketch an argument for the bound of $\lg(\alpha_1/\alpha_m)$. Let s be a Nash equilibrium, and c be $\lfloor \frac{\max_j l_j(s)}{\text{OPT}} \rfloor$. We want to show that $c = O(\frac{\lg m}{\lg \lg m})$.



We define L_k for $k \in \{0, 1, \ldots, c-1\}$ as the longest prefix of $1, 2, \ldots, m$ such that each machine in L_k has load at least kOPT. Recall that $\alpha_1 \geq \alpha_2 \geq \ldots \geq \alpha_m$. In the above example:

$$\begin{aligned} L_0 &= \{1, 2, \ldots, 8\} \\ L_1 &= \{1, 2, \ldots, 8\} \\ L_2 &= \{1, 2, \ldots, 7\} \\ L_3 &= \{1, 2\} \\ L_4 &= \emptyset \end{aligned}$$

We use the next two lemmas in the rest of the proof.

Claim 1.2 $\{1\} \subseteq L_{c-1}$.

Claim 1.3 $|L_k| \geq (k+1)|L_{k+1}|$ *for all* $k \in \{1, \ldots, c-2\}$.

Using the above claims:

$$\begin{aligned} |L_{c-1}| &\geq 1 \\ |L_{c-2}| &\geq (c-1)|L_{c-1}| \\ |L_{c-3}| &\geq (c-2)|L_{c-2}| \end{aligned}$$

Consequently,

$$|L_0| \geq (c-1)!$$

We also know that $|L_0| = m$. So,

$$m \geq (c-1)! \quad \Rightarrow \quad c = O(\frac{\lg m}{\lg \lg m})$$

□

We now prove the claims.

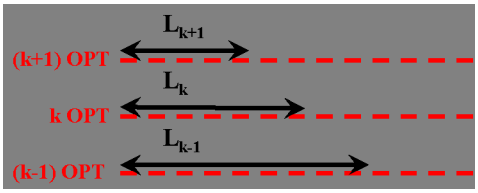
Proof of 1.2. Suppose that $1 \notin L_{c-1}$. Then, $l_1(s) < (c-1)$OPT. By definition of c, there exists some machine $j \neq 1$ such that $l_j(s) \geq c$OPT. The difference between the load of this machine and the first machine is $>$ OPT. Take any job i assigned to machine j. Note that $\frac{w_i}{\alpha_1} \leq$OPT. We can move i to machine 1 which is the fastest machine and it does not add more than OPT to the fastest machine load. So, s is not a Nash equilibrium which is a contradiction with one of our assumptions.

Therefore,

$$l_1(s) \geq (c-1)\text{OPT} \quad \Rightarrow \quad \{1\} \subseteq L_{c-1}$$

□

Proof of 1.3. Let $A_k = \{i | s_i \in L_k\}$ be all jobs scheduled on L_k machines.



Sub-Claim 1.4 *In any optimum solution, all jobs in A_{k+1} can only be scheduled on machines in L_k.*

Proof: Let j_k be the slowest machine in L_k ($j_k = |L_k|$). The load on $j_k + 1$ should be less than kOPT, otherwise $j_k + 1$ would have belonged to L_k. We show that no job in A_{k+1} can be scheduled on $j_k + 1$ which is the fastest machine not in L_k. Let i be a job in A_{k+1}. Since s is a Nash, i should not benefit form moving to machine $j_k + 1$. In other words, the load of $j_k + 1$ plus the load of job i on this machine should be greater than the current load of the machine handling i.

$$\begin{aligned} \frac{w_i}{\alpha_{j_k+1}} + k\text{OPT} &> (k+1)\text{OPT} \quad \Rightarrow \\ \frac{w_i}{\alpha_{j_k+1}} &> \text{OPT} \end{aligned}$$

And this contradicts our assumptions. □

Since the load of all the machine in A_{k+1} is greater than $(k + 1)$OPT, we can write the following inequality about the total weight of jobs in A_{k+1}.

$$\sum_{i \in A_{k+1}} w_i \geq (\sum_{j \leq j_{k+1}} \alpha_j)(k+1)\text{OPT}$$

Based on the sub-claim, these jobs can only be scheduled on L_k. So,

$$\begin{aligned} \text{OPT} \geq \frac{\sum_{i \in A_{k+1}} w_i}{\sum_{j \leq j_k} \alpha_j} &\Rightarrow \text{OPT} \geq \frac{(\sum_{j \leq j_{k+1}} \alpha_j)(k+1)\text{OPT}}{\sum_{j \leq j_k} \alpha_j} \\ &\Rightarrow \sum_{j_{k+1} < j \leq j_k} \alpha_j \geq k \sum_{j \leq j_{k+1}} \alpha_j \\ &\Rightarrow (|L_k| - |L_{k+1}|)\alpha_{j_{k+1}+1} \geq k \sum_{j \leq j_{k+1}} \alpha_j \\ &\Rightarrow (|L_k| - |L_{k+1}|) \geq k \sum_{j \leq j_{k+1}} \frac{\alpha_j}{\alpha_{j_{k+1}+1}} \end{aligned}$$

Since $\alpha_j \geq \alpha_{j_{k+1}+1}$ we have

$$\begin{aligned} &\Rightarrow (|L_k| - |L_{k+1}|) \geq k|L_{k+1}| \\ &\Rightarrow |L_k| \geq (k+1)|L_{k+1}| \end{aligned}$$

□

Proof Sketch for the $O(\lg(\alpha_1/\alpha_m)$ bound: Let c and L_k be defined as in the above proof. We claim the following: $\alpha_{j_{k+2}+1} > 2\alpha_{j_k+1}$ for $k \in \{0, 1, \ldots, c-3\}$. We observe that in an optimum

solution, there must be some job $i \in A_{k+2}$ that is scheduled on a machine outside of L_{k+2}. This implies that $w_i/\alpha_{j_{k+2}+1} \leq \text{OPT}$ since $\alpha_{j_{k+2}+1}$ is the fastest machine not in L_{k+2}. Since the current schedule is a Nash and i did not move to any machine in L_k, each of which has load at most kOPT, we have that $w_i/\alpha_{j_{k+1}+1} > 2\text{OPT}$ (note that all machines in L_{k+2} has load at least $(k+2)$OPT and $j_{k+1}+1$ is the fastest machine in L_k). The claim follows. We leave it as a simple exercise to argue that $c = O(\log m)$ using the claim.

2 Coordination Mechanism: Change the Game to Make Players Behave

We showed that the PoA for related machines is $O(\min(\frac{\lg m}{\lg \lg m}, \lg \frac{\alpha_1}{\alpha_m}))$. Can we improve the PoA by some "mechanism"? The mechanism should be (a) Simple to implement. (b) Local to each machine.

Consider the following rules for scheduling jobs on a machine:

- Shortest first: a machine schedules the jobs on it in the order of increasing job size.
- Longest first
- Random order
- Round Robin

Each rule above is a "policy". Knowing the policy, the selfish players (jobs) will behave differently. The important question is how the policy affects the PoA. Note that game has changed now in terms of the utility functions. Each selfish job wants to finish as early as possible and this of course depends on the policy. The following theorems show how a policy helps reducing PoA.

Theorem 2.1 *Under longest-first policy, PoA for related machines is* $\leq 2 - \frac{2}{m+1}$.

Proof Idea. One can show that the makespan under this policy for any Nash is the same as the makespan of the greedy algorithm that orders jobs in decreasing weight. It is known that this greedy algorithm gives a schedule with makespan at most $2 - \frac{2}{m+1}$ times the optimal makespan. □

It is known that the PoA for shortest-first policy is $\Theta(\lg m)$. However, players may cheat in longest-first policy by stating that their job length is longer than it actually is. It does not happen in shortest first if we reasonably assume that a player would be unhappy if its job is left unprocessed. A mechanism is *truthful* if players have incentive to say their true value. Longest-first is in this sense not a truthful mechanism.

3 Mixed Equilibrium of Identical Machines

So far we have discussed pure strategies. There are m machines available and n jobs controlled by n selfish players $1, \ldots, n$. First consider the identical machines case. Player i controls job i with weight $w_i \geq 0$ which needs to be scheduled on one of the m machines. Define a load balancing game as $(N, \langle S_i \rangle, \langle u_i \rangle)$, where $N = \{1, \ldots, n\}$ denotes n players (jobs), $S_i = \{1, 2, \ldots, m\}$ is the set of

possible strategies of player i, and $u_i : S = S_1 \times \ldots, \times S_n \mapsto \mathbb{R}$ is the payoff function of player i. p_i represents a mixed strategy for player i which is a probability distribution over $S_i = \{1, 2, \ldots, m\}$. p_{ij} is the probability of job i wanting to go on machine j.

For a mixed strategy profile $p = (p_1, \ldots, p_m)$, load of machine j is the total weight of jobs scheduled on machine j. It is calculated from the equation bellow:

$$l_j(p) = \sum_i w_i p_{ij}.$$

Note that $l_j(p)$ is now a random variable. Let $E[l_j(p)]$ denote the expectation of the load of j if the strategy profile is p.

To discuss PoA we need to define a social objective? In the mixed game, since the loads of the machines are random variables, we need to define the maximum load in a different way. There are two reasonable but definitions here. The first is the maximum of the expectations of the loads; formally $\max_j E[l_j(p)]$. The second and more stringent definition is the expectation of the maximum load, formally $E[\max_j l_j(p)]$. It is not hard to see that the second measure is always greater than the first and in fact often much larger. We assume that the social objective is the latter and hence we want to minimize it.

One can show that PoA in the mixed game is $\Theta(\log m / \log \log m)$. Note that this is much worse than the upper bound of 2 we showed for pure equilibria! An example that shows a bad equilibrium that achieves a PoA of $\Omega(\log m / \log \log m)$ is the following. Consider m jobs each of weight $w_i = 1$ to be scheduled on m identical machines. Consider p where $p_i = (\frac{1}{m}, \frac{1}{m}, \ldots, \frac{1}{m})$ for each i. By symmetry one can see that p is a mixed Nash equilibrium. What is the expectation of the maximum loaded machine for this p? This is related to the following well-known ball and bins analysis. Suppose we throw m balls into m bins with each ball choosing independently a bin at random according to the uniform distribution. What is the expectation of load of the maximum loaded bin? It is $\Theta(\log m / \log \log m)$.

We note that if the social objective is to minimize $\max_j E[l_j(p)]$ (the easier objective) then the PoA is again upper bounded by 2.

The following table summarizes the known results on the PoA for the load balancing scenarios we discussed. The result for the mixed equilibria on related machines is quite technical.

	Pure	Mixed
Identical	$2 - \frac{2}{m+1}$	$\Theta(\frac{\lg m}{\lg \lg m})$
related	$\Theta(\min(\frac{\lg m}{\lg \lg m}, \frac{\lg \alpha_1}{\alpha_m}))$	$\Theta(\frac{\lg m}{\lg \lg \lg m})$

The price of stability (PoS) for all of the above scenarios is 1 because there is a pure equilibrium that achieves the minimum makespan. This is not completely obvious. One starts with an assignment of jobs to machines that minimizes makespan and then alters it to make it a Nash equilibrium without increasing the makespan — this process is called *Nashification*. It is useful exercise to see how to do this; the identical machines case is easy.