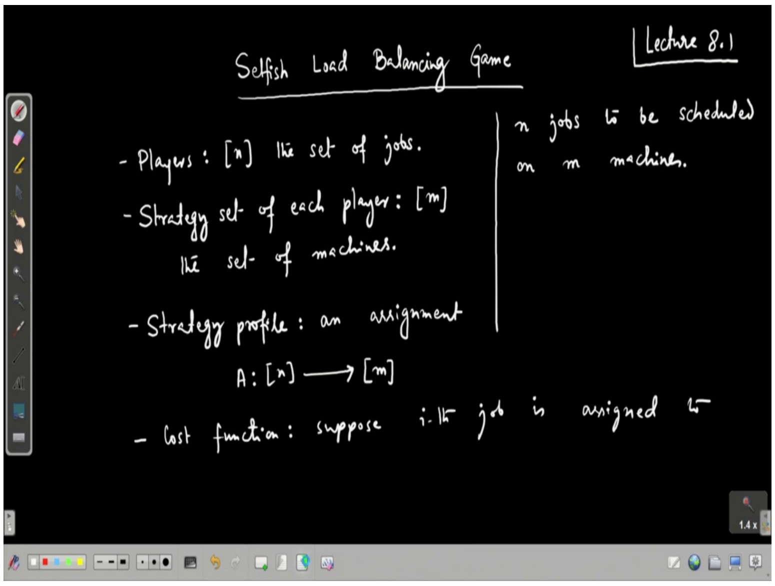
Algorithmic Game Theory

Prof. Palash Dey
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 36
PoA of Selfish Load Balancing Game

Welcome. So, from the last two lectures we have been studying price of anarchy and today we will continue that. In the last lecture we have studied price of anarchy of selfish network routing games and we have proved a very powerful result that the price of anarchy depends only on the amount of non-linearity on the non-linearity of the cost function. It does not depend so much on the network structure.

(Refer Slide Time: 00:53)



Selfish Load Balancing Game

Lecture 8.1

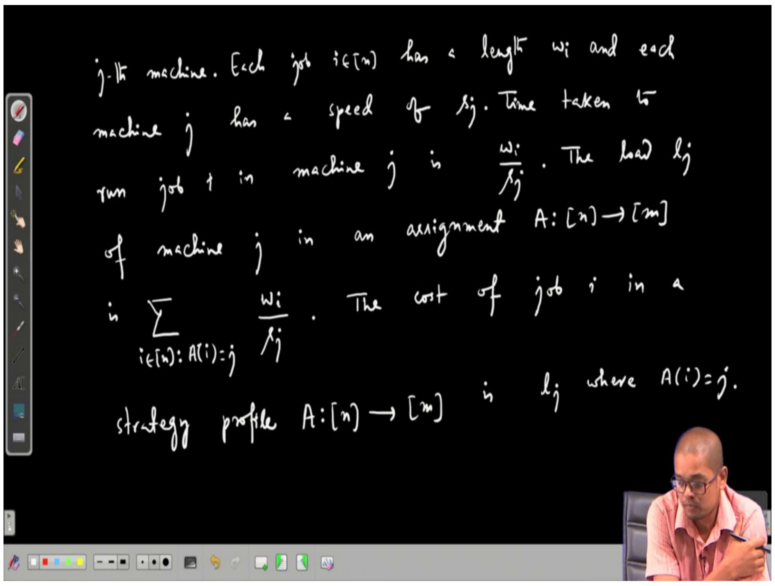
- Players : $[n]$ the set of jobs.
- Strategy set of each player : $[m]$ the set of machines.
- Strategy profile : an assignment $A : [n] \longrightarrow [m]$
- Cost function : suppose i-th job is assigned to

n jobs to be scheduled on m machines.

So, today we will take another example of a game, a very popular game which is called Selfish Load Balancing Game and we will again study price of anarchy of this game; selfish load balancing game. So, what is the game? Who are the players? So, first players: players are they are n players and each player is a job, think of we have n jobs and we are trying to schedule them across m machines.

So, n jobs to be scheduled on m machines, this is the underlying scenario. So, the jobs are the players, n jobs this is the set of jobs and each player has an option of which machine it wants to get executed. So, strategy set of each player is m, the set of machines ok. And what is strategy profile? A strategy profile says which jobs get gets assigned to which machine. So, a strategy profile is an assignment A of jobs to machines, it simply says which jobs has been assigned to which machine ok. And there is cost function, suppose ith job is assigned to jth machine.

(Refer Slide Time: 04:54)



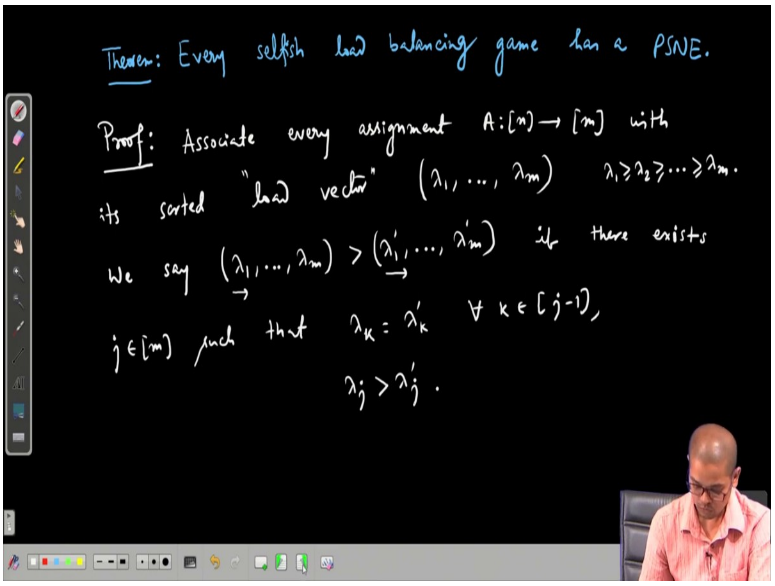
j-th machine. Each job $i \in [n]$ has a length w_i and each machine j has a speed of s_j. Time taken to run job i in machine j is $\frac{w_i}{s_j}$. The load l_j of machine j in an assignment $A : [n] \to [m]$ is $\sum_{i \in [n] : A(i) = j} \frac{w_i}{s_j}$. The cost of job i in a strategy profile $A : [n] \to [m]$ is l_j where $A(i) = j$.

Each job has a length, then the cost of ith job sorry, each job i in n has a length l_j; length say let us call it w_i and each machine j has a speed of s_j. So, time taken to run job i in machine j is $\frac{w_i}{s_j}$ ok. And the load of a machine j in an assignment, the load l_j of machine j in an assignment A is the total amount of time for which the machine runs. So, some of the time taken for the jobs which are assigned to that machine to get them executed.

So, $i \in [n]$ such that this job i has been assigned to machine j. And how much time it will take, for measure for the job i to get executed machine j? $\frac{w_i}{s_j}$. So, this is the load, the some of the times of these jobs to get executed in this machine j is its load. And now we go to the, now we come to the cost function. The cost of job i in a strategy profile, which is an assignment let us call it A, in a strategy profile $A : [n] \rightarrow [m]$, the cost of job i in a strategy profile A is the load of the machine j is l_j, where $A(i) = j$. That job i has been assigned to machine j ok.

So, in the you know in the last example of the selfish network routing game, because it was a potential game it was clear that it that game has a pure strategy Nash equilibrium, which is not so clear in this shellfish load balancing game. So, that is the first we need to show that it has a pure strategy Nash equilibrium, otherwise the price of anarchy concept itself is not defined. We have defined price of anarchy with respect to pure strategy Nash equilibrium.

(Refer Slide Time: 10:10)



Theorem: Every selfish load balancing game has a PSNE.

Proof: Associate every assignment $A:[n] \rightarrow [m]$ with its sorted "load vector" $(\lambda_1, \dots, \lambda_m)$ $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_m$.

We say $(\underset{\rightarrow}{\lambda_1}, \dots, \lambda_m) > (\underset{\rightarrow}{\lambda_1'}, \dots, \lambda_m')$ if there exists $j \in [m]$ such that $\lambda_k = \lambda_k'$ $\forall k \in [j-1]$,

$\lambda_j > \lambda_j'$.

So, an important theorem is every selfish load balancing game has a pure strategy Nash equilibrium proof. You see that this is very important for price of anarchy to make any sense, because in the price of anarchy we need the cost of a strategy profile which is a equilibrium, which is a pure strategy Nash equilibrium, ok.

So, how the proof goes? We associate every assignment A; we need to show that there exists a strategy profile which is pure strategy Nash equilibrium where there is no benefit for any player to deviate unilaterally and strategy profiles are nothing, but assignments. So, we start with strategy profile with assignment and we associate them with its sorted load vector.

What is it? What let us call it say $l_1, \dots, l_m$; not l_1, l we have used. $\lambda_1, \dots, \lambda_m$. So, why sorted? You see you for the assignment you see what is the load of each machine and you sort this ok. So; that means, say $\lambda_1 \geq \lambda_2$. So, λ_1 is not the load of machine 1, it is the highest load.

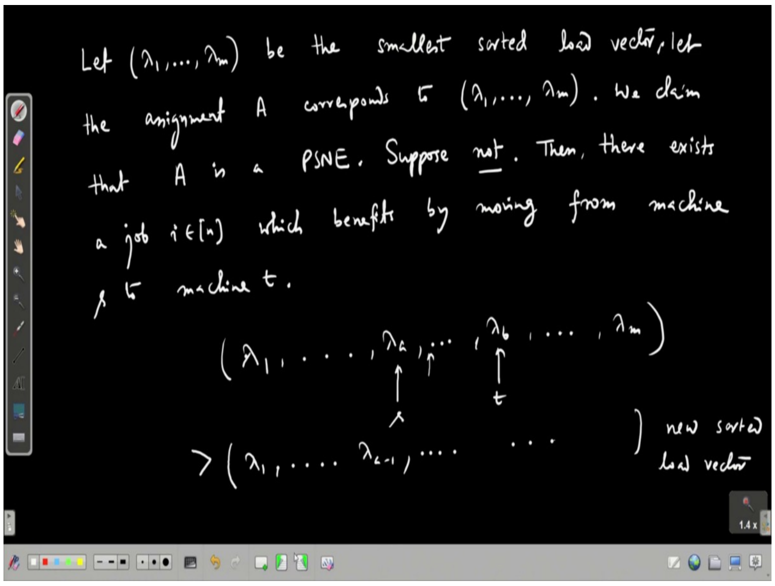
So, λ_1 may be the load of say 10th machine, λ_2 is the second highest load, it may be the load of 20th machine and so on. And you break ties arbitrarily; think of them as just numbers ok. Now, we will compare load vectors, we will define a rule to compare load vectors. So, define we say I have 2 load vectors $\lambda_1, \ldots, \lambda_m$, this is say we say greater than another load vector $\lambda_1^{'}, \ldots, \lambda_m^{'}$, this called lexicographic order.

You start from left and see where in which index they differ first. For example, suppose $\lambda_1 = \lambda_2^{'}$, fine you go to λ_2 check whether is it $\lambda_2 = \lambda_2^{'}$. If suppose it is again yes, then you go to the third index, take the first index from left where they differ, suppose it at the 10th index the differ. $\lambda_{10} \neq \lambda_{10}^{'}$. From $\lambda_1, \ldots, \lambda_9$ is same as $\lambda_1^{'}, \ldots, \lambda_9^{'}$. At tenth index in it differ and then at that index you see which one is more.

So, if $\lambda_{10} > \lambda_{10}^{'}$, then use we say that this tuple $(\lambda_1, \ldots, \lambda_m) > (\lambda_1^{'}, \ldots, \lambda_m^{'})$. So, we say we are defining this thing, this con if we say $(\lambda_1, \ldots, \lambda_m) > (\lambda_1^{'}, \ldots, \lambda_m^{'})$, if there exists an index $j \in [m]$, such that $\lambda_k = \lambda_k^{'}$ for all $k \in [1, j-1]$ and $\lambda_j > \lambda_j^{'}$ ok.

So, for each strategy profile, we have assignments and for each assignments they have we have these load vectors sorted load vectors, among all those sorted load vectors which corresponds to assignments, we see we find out which one is the smallest one, according to this order.

(Refer Slide Time: 15:50)



Let $(\lambda_1, \ldots, \lambda_m)$ be the smallest sorted load vector, let the assignment A corresponds to $(\lambda_1, \ldots, \lambda_m)$. We claim that A is a PSNE. Suppose not. Then, there exists a job $i \in [n]$ which benefits by moving from machine s to machine t.

$$(\lambda_1, \ldots, \underset{\overset{\uparrow}{s}}{\lambda_a}, \ldots, \underset{\overset{\uparrow}{t}}{\lambda_b}, \ldots, \lambda_m)$$

$$> (\lambda_1, \ldots \lambda_{a-1}, \ldots \quad \ldots \quad)$$ new sorted load vector

So, let lambda 1 to lambda m be the smallest sorted load vector, you see which one is smallest. We claim that the strategy profile for whose corresponding assignments load sorted load vector is $(\lambda_1,\ldots,\lambda_m)$ must be a PSNE. So, we claim and let A, the assignment let the assignment A gave rise to, A corresponds to this sorted load vector $(\lambda_1,\ldots,\lambda_m)$.

We claim that A is a PSNE; A is a Pure Strategy Nash Equilibrium. We will show that there is no unilateral beneficial deviation. So, suppose not then there exist a job say $i\in[n]$ which benefits by moving from machine s to machine t; suppose there exists a job. So, you ask what are the indices in this sorted load vector, λ_1 to suppose this is λ_a is the load of this machine s, and because it moved to machine t, t must be less loaded than s. So, t must appear to the right of λ_a, suppose this λ_b.

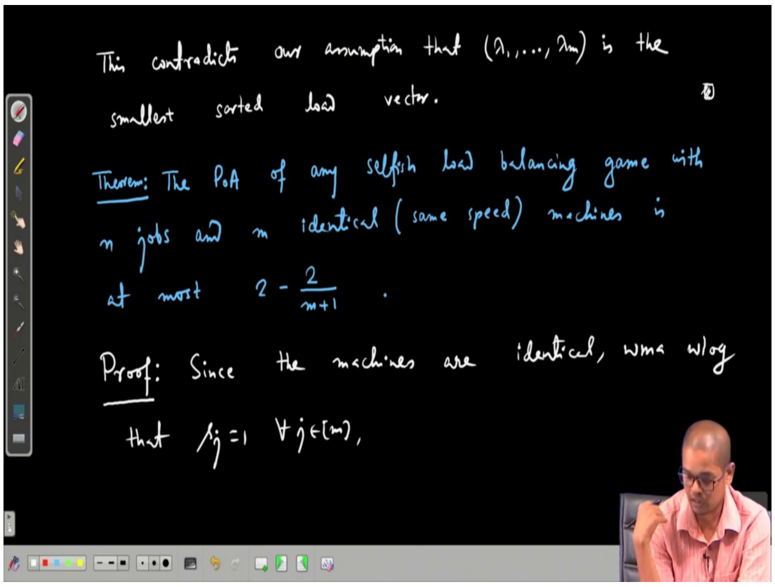
And now so, what is you ask what is the load vector of after job i is moves from machine s to machine t. First we observe that the load of machine t after i is i moves from s to t, it cannot be more than or greater than equal to λ_a, it should be strictly less than λ_a. First observe that the load of every machine except s and t remains same and also the machine, the load of machine t after i moves there, its load must be strictly less than lambda a, then only it is a beneficial move for machine for job i.

So, the new node vector has the first $(a-1)$ entry is same and the a th entry. What could be a th entry? a th entry could be this entry, lambda a minus the time taken for time taken for this executing machine, executing job i in machine A or it could be some other this this thing that λ_{a+1}. But whatever it is, you see that you know this new is the new sorted load vector, this is strictly less than the sorted load vector we started with.

But you know the we assumed that this sorted load vector that we started with $(\lambda_1,\ldots,\lambda_m)$, is this is one of the smallest sorted load vector. There is no other sorted load vector which is smaller than that, then it is a contradiction. Because I have found another sorted load vector which is smaller than that and that cannot happen, because we have defined a complete order.

Any two sorted load vectors we can compare, so this makes the ordering complete and we have a finite set of sorted load vectors, because there are finitely many assignments possible. So, I have a complete order on a finite set. So, the minimum must be unique and if $(\lambda_1, \ldots, \lambda_m)$ is a is a minimum, then there cannot be another minimum and which is a contradiction.

(Refer Slide Time: 22:16)



This contradicts our assumption that $(\lambda_1, \ldots, \lambda_m)$ is the smallest sorted load vector.

Theorem: The PoA of any selfish load balancing game with n jobs and m identical (same speed) machines is at most $2 - \frac{2}{m+1}$.

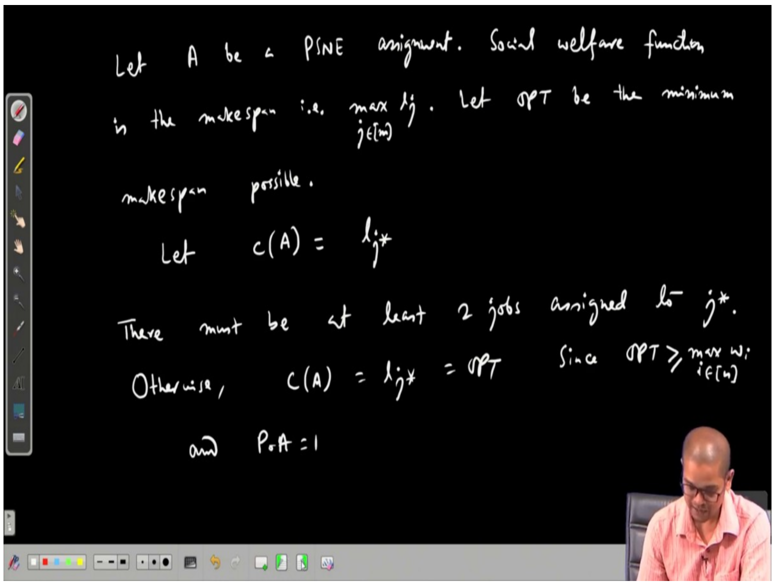
Proof: Since the machines are identical, wma wlog that $s_j = 1 \; \forall j \in [m]$,

Let me write, but this contradicts our assumption that $(\lambda_1, \ldots, \lambda_m)$ is the smallest sorted load vector. So, this results our fundamental question that you know does price of anarchy make sense here, does there exist a PSNE, yes. In this game there exists a PSNE. Now, we will show a price of anarchy bound for this selfish load balancing game. So, here is the theorem, the price of anarchy bound of selfish load balancing game in its full generality is out of scope of this course.

So, we will prove the price of anarchy bound for a special case, for an important special case. The price of anarchy of any selfish load balancing game with n jobs and m identical machines; what do you mean by identical? Their speeds are same. Identical machines is the price of anarchy is at most $2 - \frac{2}{m+1}$.

Proof: first observe that since the machines are identical, identical we may assume without loss of generality, that s_j is 1, the speed of all machines are 1 for all $j \in [m]$. If they are not one we can scale the weight of the jobs appropriately, we can if the speeds are something s_j we can divide the weights of the jobs by s_j and make the machine the speed of the machines to 1, that will not change anything.

(Refer Slide Time: 26:22)



Let A be a PSNE assignment. Social welfare function is the makespan i.e. $\max_{j\in[m]} l_j$. Let OPT be the minimum makespan possible.

Let $c(A) = l_{j^*}$

There must be at least 2 jobs assigned to j^*.

Otherwise, $c(A) = l_{j^*} = OPT$ Since $OPT \geq \max_{i\in[n]} w_i$

and $PoA = 1$

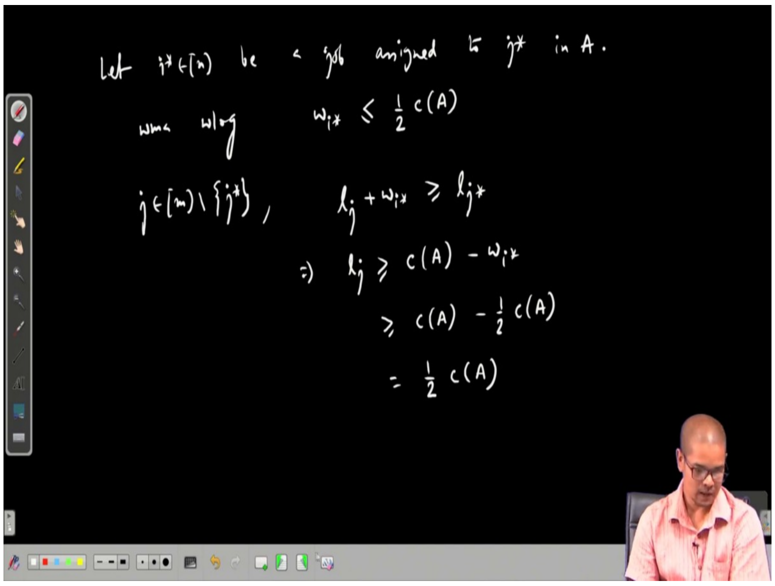
Now, let A be a PSNE assignment. What is the social welfare function? It is the make span, social welfare function is a make span which is maximum load of any machine, that is the social welfare function let me write. Social welfare function is the make span of the schedule, make span that is which is maximum load of any machine, ok.

And say let OPT be the minimum makes span possible. First observe that you know look at this PSNE profile A and ask what is its makes span. So, let $C(A)$, the cost of A, the make span of A is the highest make span and that is the makes highest load of any machine. And suppose that is the load of machine j star. Suppose this l_{j^*} first observe that there must be at least 2 jobs assigned to machine j^*.

Must be at least 2 jobs assigned to the machine gesture. Why? Because if there is only one job, then otherwise we have $C(A)=l_{j^*}$, but j^* equal to will be then OPT. Why? Because OPT, since OPT is greater than equal to, because all jobs must be executed $max_{i\in[n]} w_i$. So, because all jobs must be executed, in particular the maximum weight drop must also be executed, there must be one machine which is executing the maximum weight job.

And if j^* is the maximum makes is the maximum load and if that is executing only 1 job, then j^* is less than equal to the maximum weight of any job and that it must be OPT and in that case and then. So, what is wrong with j^* being equal to up? Nothing wrong, in that case we have price of anarchy is 1 and we have nothing to prove, the statement is already true, because 1 is the minimum price of anarchy that can be attained. So, from the for the remaining proof, let us assume that there are at least 2 jobs assigned to the machine j star in the assignment A.

(Refer Slide Time: 30:35)



Let $i^* \in [n]$ be a job assigned to j^* in A.

wma wlog $w_{i^*} \leq \frac{1}{2} c(A)$

$j \in [m] \setminus \{j^*\}$, $l_j + w_{i^*} \geq l_{j^*}$

$\Rightarrow l_j \geq c(A) - w_{i^*}$

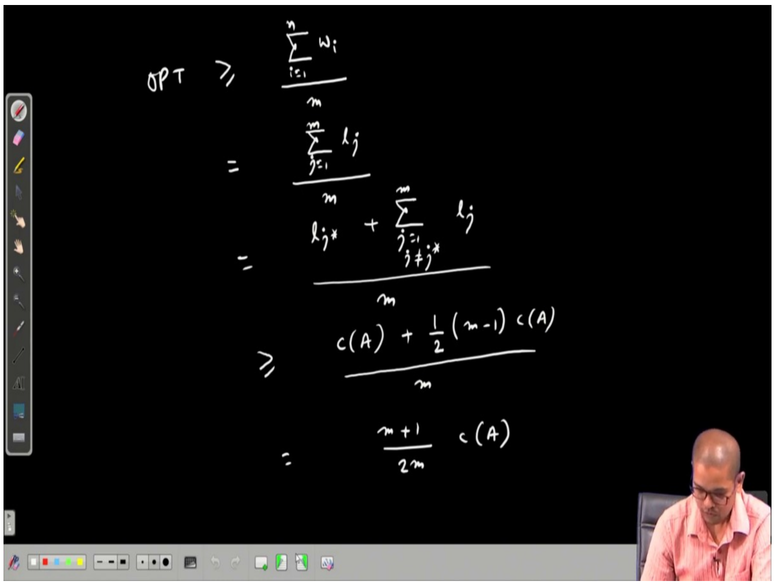
$\geq c(A) - \frac{1}{2} c(A)$

$= \frac{1}{2} c(A)$

So, let $i^* \in [n]$ be a job, assigned to j^* in the assignment A. We also can assume we may assume without loss of generality, that weight of i^* is less than equal to $\frac{1}{2} C(A)$, weight of i^* is less than equal tol_{j^*} and l_{j^*} equal to $C(A)$. Because you know there are at least two jobs, there must be at least one job which takes less than half amount of load and we can call that i start, let us call that i star; ok, very good.

Now you see the load of any other machine. So, for any $j \in [m] \setminus \{j^*\}$, you see l_j this job the i^* has an option to move from machine j^* to machine j. So, $l_j + w_{i^*}$ must be greater than equal to l_{j^*}. If this is not the case then there is a beneficial move for the job i^*. So, that is l_j is greater than equal to,l_{j^*} is $C(A) - w_{i^*}$ and w_{i^*} is at least half this is greater than equal to $C(A) - \frac{1}{2} C(A)$, which is equal to $\frac{1}{2} C(A)$, very good. Now, we will prove the price of anarchy bound.

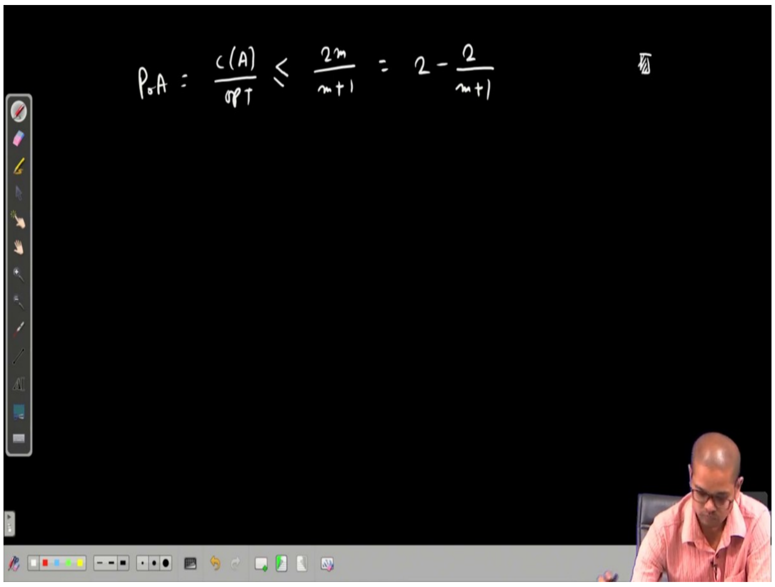
(Refer Slide Time: 32:52)



$$OPT \geq \frac{\sum_{i=1}^{n} w_i}{m} = \frac{\sum_{j=1}^{m} l_j}{m} = \frac{l_{j^*} + \sum_{j=1, j\neq j^*}^{m} l_j}{m} \geq \frac{c(A) + \frac{1}{2}(m-1)c(A)}{m} = \frac{m+1}{2m} c(A)$$

We will bound OPT. What is OPT? OPT is you know the average length of the average weight of the of all the jobs by $[m]$. Now, sum of weights is same as sum of loads. Let us add the sum of loads, we separate out l_{j^*} and keep the others and l_{j^*} is $C(A)$, cost of A and for others it is they are at least $\frac{1}{2}C(A)$, there are $m-1$ of them. So, greater than equal to $\frac{C(A)+\frac{1}{2}(m-1)C(A)}{m}$; this is $\frac{m+1}{2m}C(A)$.

(Refer Slide Time: 33:14)



$$P.A : \frac{c(A)}{OPT} \leq \frac{2m}{m+1} = 2 - \frac{2}{m+1}$$

So, what is the price of anarchy bound? Price of anarchy is cost of A by OPT. And what is cost of A by OPT? Cost of A by OPT is less than equal to $\frac{2m}{m+1}$, is also known as $2-\frac{2}{m+1}$, which is exactly what we need to prove, ok.

Thank you.