CS 684 Algorithmic Game Theory	Scribe: Yisong Yue
Instructor: Eva Tardos	Friday, September 2nd, 2005

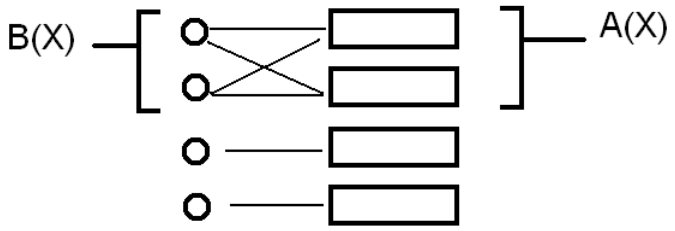
Last Time - covered non-atomic load balancing and proved Nash minimizes maximum load.
Today - prove Nash exists and is unique
Next Time - Nash Networks

Recall from last time...

$Machines\ 1,\ 2, ...m,\ Job\ types\ 1, 2, ...n$
$r_i(L) - monotonic\ increasing,\ continuous\ response\ time\ function\ of\ machine\ i\ given\ load\ L$
$p_j := total\ amount\ of\ job\ type\ j$
$S_j := set\ of\ machines\ accessible\ by\ job\ type\ j$

A solution is a vector X satisfying:
$x_{ij} := amount\ of\ job\ type\ j\ which\ uses\ machine\ i\ (0\ if\ i \not\ni S_j)$
$p_j := \sum_{i=1}^{m} x_{ij},\ \ L_i = \sum_{j=1}^{m} x_{ij},\ \ x \geq 0$
$X\ is\ Nash\ if\ (\forall x_{ij} > 0\ \&\ k \in S_j) \rightarrow (r_i(L_i) \leq r_k(L_k))$

Given Nash Solution X



A(X) := set of machines that have the worst response time
B(X) := set of job types which use A(X)
More formally, let $r_{max} = \max_{i:L_i>0} r_i(L_i)$. Then $A(X) := \{i \mid r_i(L_i) \geq r_{max}\}$ and $B(X) := \{j \mid x_{ij} > 0\ for\ some\ i \in A(X)\}$

Observation: For a Nash Equilibrium X, jobs in $j \in B(X)$ have $S_j \subseteq A(X)$.
This is because jobs only use $A(X)$, the machines with worst response time, only if there are no better options.

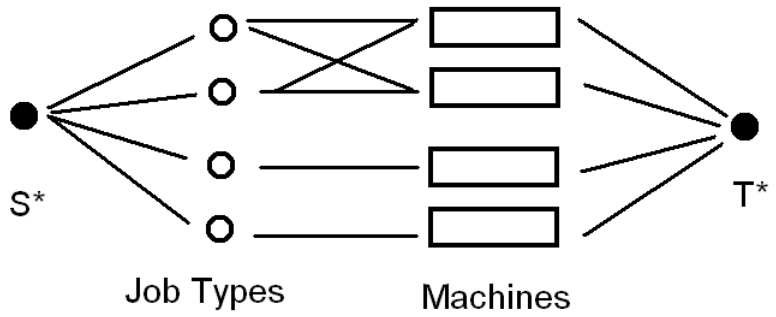
Recall from last time that we proved that

Claim: For any Nash solution X, the value r_{max} is as small as possible among all solutions.
So if we want to find a Nash solution, first we can worry about finding the value r_{max}, or first prove that such a solution exists.

Claim: $\exists$ solution minimizing maximum response time

- The claim is obvious if we define the problem so that splits are integers, as in that case there are only a fininite number of possible assignments of jobs to machines, and among this finite set of option one has smallest maximum response time. The difficulty of proving this statement is that with fcational splits of jobs there are infinitely many assignments. We will not prove that here that the minimum does exists.

If we want to find the minimum r_{max} we first give an **algorithm** to check if a solution with max response time $\leq r$ exists:
- Given m machines, n jobs, r desired max response time, construct a flow network between job types and machines with supersource feeding job types and supersink receiving flow from machines.



(1) Add two new nodes: S* := supersource, T* := supersink, flow travels from S* to T*
(2) Add a node for each machine, and one for each job type
(3) edges connecting S* to Job Types have capacity p_j, respectively
(4) edges connecting Job Types to Machines have capacity ∞
(5) edges connecting Machines to T* have capacity $r_i^{-1}(r) := \max_i L \; s.t. \; r_i(L) \leq r$
(6) The flow value we want is $\sum_j p_j$, which is maximum possible as it fills all edges leaving the supersource S*

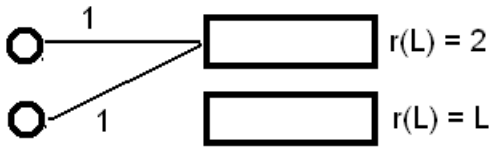
Theorem: There is a flow of value $\sum_j p_j$ in the flow network iff $\exists$ solution with response time $\leq r$ in load balancing problem.

Proof: Given any solution with maximum response time at most r, let the amount of flow from Job Type j to Machine $i = x_{ij}$ of the load balancing solution. Also, given any flow of value $\sum_j p_j$ we can define x_{ij} to be the amount of flow from node of job type j ro machine i.

While we will not need this fact, you may want to note that using the Min Cut / Max Flow Theorem we can derive that a solution with maximum response time does not r exists, if and only if there is a subset of jobs B s.t. total capacity of the machines in $A = \bigcup_{j \in B} S_j$ is less than $\sum_{j \in B} p_j$.

Now we can use binary search to find the smallest possible r which still has a flow solution (remember that this will give a load balancing solution where the response time on each Machine does not exceed r).

Now suppose we've found a flow solution using minimal r. However, such a solution may not be a Nash, as shown by the example below.



We want to give an **algorithm** to convert given load balancing solution X with minimum possible response time r_{max} to a Nash solution. For this algorithm we will assume that the response functions r_i are strictly monotone increasing.

- Let A = $\{i \mid r_i(L_i) \geq r_{max}\}$
- Consider, does there exist j where $x_{ij} > 0$ for some $i \in A(X)$ and $k \notin S_j$ for some $k \ni A(X)$?
- If such a j and machine k exists, then X is not a Nash as this violates the Nash property. To improve the solution we move a bit of x_{ij} to x_{kj}. If we move a small enough amount then machine i will no longer have best response time (as we assumed r_i is strictly monotone increasing), and r_k still does not become a machine with worst response time (as we moved a small enough amount.
- The effect of the change: $A(X) \rightarrow A(X) \backslash \{i\}$
- Repeat this till all jobs in $j \in B(X)$ have $S_j \subseteq A(X)$. Note that $A(X)$ cannot become the empty set, i.e., the maximum response time cannot decrease, as we started by determining what the minimum possible maximum response time is.

- Now consider if the resulting solution is a Nash? It does satisfy the Nash property on machines $A(X)$ and jobs $B(X)$, but the other jobs and machines may not satisfy the Nash property.
- We now create Nash solution by repeating process on remaining machines and jobs. ■

Comments on Uniqueness:
Recall that we know that all Nash solutions mave the minimum possible maxiumum response time. Note that the jobs using the machines of maximum response time don't have any other machines they can use, so in all solutions all these jobs must have maximum response time. This shows that the maximum response time, and the of jobs that have this maximum response time is the same on all Nash equlibria.

In fact, the response time of all jobs is unique, i.e., the same in all Nash solutions. However, the Job Type allocations to Machines is not unique.

Summary:
(1) we converted the load balancing problem to a flow network problem (using r)
(2) we repeatedly solved for max flow in the flow networks until we found the minimal r for which a max flow solution exists
(3) we converted the solution back to a load balancing solution
(4) if the load balancing solution is not Nash (A(X) not minimal), then we repeatedly removed Machines from A(X) until it is minimal
(5) the result is a unique (with respect to r). (6) we recursively find a Nash solution for the machines not in $A(X)$ and the remaining jobs not assigned to machines $A(X)$.

Theorem: Nash solution which MUST exist for any given load balancing problem