CS 684 Algorithmic Game Theory	Scribe: Elizabeth Van Nostrand
Instructor: Eva Tardos, Lecture by Ara Hayrapetyan	10/28/2005

Today's lecture is on "Congestion Games with Player-Specific Payoff Functions" by Igal Milchtaich. This paper describes a game where players are allowed to respond differently to different loads on different machines- i.e. job k may respond differently to a load of 5 on machine i than to a load of 5 on machine j, and jobs i and j may react different to the same load on the same machine. Not only is this game more realistic, it is a good example of a game with a Nash equilibrium but no potential function.

1 The Game

The game consists of:

n jobs of unit size

m servers

each job j has associated response function of l_{ji}(x) for load x on machine i, for 1$\leq$ i $\leq$ m

as usual, each job wants to minimize their own response time.

2 Exitance of a Nash

Is there a Nash?

Claim: there exists a Nash equilibrium. Proof by induction on n

Base case: 1 machine, m servers

Proof. The job will choose its best machine, which is clearly a Nash ■

Inductive Hypothesis: if there exists a Nash Equilibrium for n jobs, there is a Nash equilibrium for n+1 jobs.

Proof. Suppose we have a achieved a Nash equilibrium with n jobs.

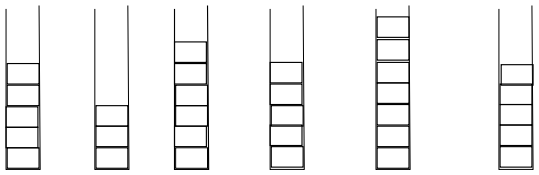

Figure 1: initial configuration

Since this is at equilibrium, we know that $l_{ji}(k_i) \leq l_{jp}(k_p + 1)$, where i is j's machine, p is any other machine, and k_p is the load on machine p. Now we wish to add job n+1. This job will pick its favorite machine m and settle there.

Observation 1: no job on the other machines will wish to move

Proof. It still holds that $l_{ji}(k_i) \leq l_{jp}(k_p+1)$, since all servers have either the same or greater load. ■

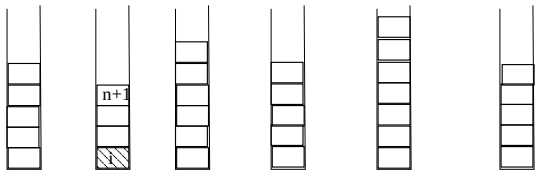

Figure 2: initial configuration with $n + 1$th job added

Observation 2: If no job on machine m wants to move, we have a Nash, so the interesting case is when a job on m now wishes to move

Proof. By defintion, if no job wants to move, the system has arrived at a Nash equilibrium.

We must consider cases where a job on machine m wishes to move. Call this job i (shaded on figure). Note that more than one job may wish to move, but we will only remove one for reasons that will become obvious. If we remove this job but do not add it to another server, the game now looks like this

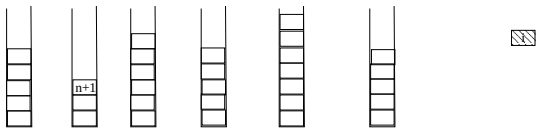

Figure 3: configuration with $n + 1$th job added, and job i removed

Observation 3: every k_p is exactly the same as before, so no job will wish to move. But we must still consider job i. We are in much the same position we were with job n+1.

We now enter the following sequence: place a job, allow a job on that machine to move, place a job, allow a job on that machine to move... If we can show that it terminates, i.e. that no job wants to move, we will have a Nash equilibrium.

Observation 4: on the best response path the loads on all machines will equal the original k_1 through k_m, except for a single machine k_j, which will have (k_j+1). Furthermore, every time a job i moved away from machine j, the load must have been k_j +1

Claim: If during the best response path job i moves on to machine j, then it will always stay on machine j.

Proof. job i chose machine j $\rightarrow l_{ij}(k_j+1) \leq l_{jp}(k_p+1)$, for all p $\neq$ j. This inequality will be true not matter what the later movements, so the job will never want to move again. ■

Since nothing will move more than once, we have reached a Nash equilibrium ■

3 The Nonexistence of a Potential Function

There is no potential function for this game, because it is possible for players making a sequence of choices that cycles, yet each player will lower their own cost.

We did this example in class. It requires that players make an improving move, but not necessarily their best move. There are two players with the following response functions.

$l_{11}(1) \leq l_{12}(\text{x}) \leq l_{13}(\text{y}) \leq l_{11}(2)$

$l_{22}(1) \leq l_{21}(x) \leq l_{23}(y) \leq l_{22}(2)$

Given the following initial conditions, the jobs can cycle as follows

1	2	3	Deviator
-	2	1	1
-	1,2	-	2
-	1	2	1
1	-	2	2
1,2	-	-	1
2	-	1	2
-	2	1	

This example proves that the game is not a potential game. There is another example in which players always make their best move and still cycle, but this example was too complicated to go over in class.