Algorithmic Game Theory - Fall 2013

Lecturer: *Michal* *Feldman*

Last Update: 1 Dec 2013 12:11 p.m. *Deadline: December 16, 2013*

Problem set 2

Question 6:

Consider the weighted generalization of the network connection game, where every player i is associated with weight $w_i > 0$, and instead of sharing the cost equally, players are now assigned cost shares in proportion to their weight. In particular, for a strategy vector S and edge e, let S_e denote the players whose path contains e, and let $W_e = \sum_{i \in S_e} w_i$ be the total weight of these players. Then player i pays $c_e w_i / W_e$ for each edge in his path. Note that if all players have the same weight, this is the original game. Show that, in general, this game does not have an exact potential function.

Question 7:

There are n agents and m machines. Every agent has a task of the same (unit) weight, and can place it on one if the m machines. Agents can have *different cost functions*. Formally, each agent j incurs a cost $c_i^j(k)$ on machine i if a total of k agents are assigned to machine i. Assume that for each fixed j and i, $c_i^j(k)$ is nondecreasing in k.

1. Prove that best-response dynamics need not converge.
2. Give a polynomial-time algorithm for computing a pure Nash equilibrium.
3. Show that if there are only two machines, then best-response dynamics converges to a pure Nash equilibrium.

Question 8:

Consider n machines and m selfish jobs (the players). Each job j has a processing time p_j and a set S_j of machines on which it can be scheduled (i.e., S_j is the strategy space of player j). Once all jobs have chosen machines, the jobs on each machine are processed *serially* from shortest to longest. (you can assume that the p_j's are distinct). For example, if jobs with processing times 1,3, and 5 are schedules on a common machine, then they will complete at times 1, 4, and 9, respectively. Assume that players choose machines in order to minimize their completion times.

Consider the following scheduling algorithm: (1) Sort all the jobs in order from smallest to largest; (2) Schedule the jobs one-at-a-time, assigning a job j to the machine of S_j with minimum load so far (breaking ties arbitrarily). Prove that the pure Nash equilibria of the scheduling game are precisely the possible outputs of this scheduling algorithm (with the different equilibria arising from different ways of breaking ties).

[hint: if you were the smallest player, how is your personal cost affected by the others' decisions?].

Question 9:

Consider the following scheduling setting: there are m machines and n jobs. Each job j has a processing time on each machine. If the machines are *identical*, then the processing time of job j is simply p_j on all machines. If the machines are *unrelated*, then the processing time of job j on machine i is given by p_{ij}. Each job selects a machine among the m machines. Given a strategy for each job, the total load of each machine is the sum of processing times of the jobs assigned to it. Each job seeks to minimize the total load on its chosen machine.

1. Show that determining whether a given pure Nash equilibrium on 2 identical machines is a strong equilibrium can be done in polynomial time.
2. Show that it is Co-NP-hard to determine whether a given pure Nash equilibrium on $m \geq 3$ identical machines is a strong equilibrium.
3. Show that for unrelated machines, it is Co-NP-hard to determine whether a given pure Nash equilibrium on 2 machines is a strong equilibrium. [hint: reduction from Partition].