CS364A: Problem Set #4

Due in class on Thursday, March 3, 2011

Instructions: Same as previous problem sets.

Problem 16

(a) (5 points) *Algorithmic Game Theory*, Exercise 19.9.

(b) (5 points) *Algorithmic Game Theory*, Exercise 19.10.

Problem 17

(a) (10 points) *Algorithmic Game Theory*, Exercise 19.13.

(b) (10 points) *Algorithmic Game Theory*, Exercise 19.14.

Problem 18

In this problem we will investigate the convergence of best-response dynamics in variants of the load-balancing model discussed in Chapter 20 of the AGT book. Recall that "best-response dynamics" means that you begin at an arbitrary assignment, and as long as the current assignment is not a (pure-strategy) Nash equilibrium, an arbitrary player who can benefit by deviating is allowed to switch machines. A deviating player is assumed to switch to its best option (e.g., in the most basic load-balancing model, a machine with the lightest load).

(a) (15 points, plus 15 more extra credit) Suppose that all players have the same (unit) weight but players can have different cost functions. Formally, each player j incurs a cost $c_i^j(k)$ on machine i if it is among k players assigned to i. Assume that for each fixed j and i, $c_i^j(k)$ is nondecreasing in k.

Solve two of the following four problems for full credit. You will receive extra credit for each problem after the first two that you solve.

(i) Prove that best-response dynamics need not converge.

(ii) Prove that, despite (i), a pure-strategy Nash equilibrium always exists.

(iii) Give a polynomial-time algorithm for computing a pure-strategy Nash equilibrium.

(iv) Show that if there are only two machines, then best-response dynamics converges to a pure-strategy Nash equilibrium.

(b) (10 points) Now suppose that all players have the same cost function but can have different weights. Assume also that every machine has the same cost function (a function from the sum of users' weights to a common cost for all users). Consider restricting best-response dynamics so that at each step, if there are several players who want to deviate, then only the player with largest weight is allowed to deviate (ties are broken arbitrarily). Prove that this process converges to a pure-strategy Nash equilibrium after at most n iterations.

(c) (10 extra credit points) For the same model as in (b), does (unrestricted) best-response dynamics converge to a pure-strategy Nash equilibrium in time polynomial in the input (the number of players, the number of machines, and the logarithm of the maximum player weight)?

Problem 19

(a) **(7 points)** Consider an atomic selfish routing game in which all players have the same source vertex and sink vertex (and each controls one unit of flow). Assume that edge cost functions are nondecreasing, but do not assume that they are affine. Prove that a (pure-strategy) Nash equilibrium (i.e., an equilibrium flow) can be computed in polynomial time.

[Hint: Remember the potential function. You can assume without proof that the minimum-cost flow problem can be solved in polynomial time. If you haven't seen the min-cost flow problem before, you can read about it in any book on "combinatorial optimization". Be sure to discuss the issue of fractional vs. integral flows, and explain how (or if) you use the hypothesis that edge cost functions are nondecreasing.]

(b) **(7 points)** Prove that in an atomic selfish routing network of parallel links, every equilibrium flow minimizes the potential function.

(c) **(6 points)** Show by example that (b) does not hold in general networks, even when all players have a common source and sink vertex.

Problem 20

Recall the set-up for online regret-minimization: there is a fixed set A of actions; each day $t = 1, \ldots, T$ you pick an action $a^t \in A$ (possibly from a probability distribution) based only on information from previous days; and then a cost vector $c^t : A \to [0, 1]$ is unveiled. The goal is to design a (randomized) algorithm that, for every sequence of cost vectors, has small expected average regret. [Recall that the (average, per time-step) regret is the difference between your average cost ($\frac{1}{T}\sum_{t=1}^{T} c^t(a^t)$) and that of the best fixed action ($\min_{a\in A} \frac{1}{T}\sum_{t=1}^{T} c^t(a)$).]

(a) **(5 points)** Suppose that, every day, you pick the action that performed best in the past (i.e., that minimizes the cumulative cost $\sum_{s=1}^{t-1} c^s(a)$ over $a \in A$). Show that, in the worst-case, the average regret of this algorithm is $\Omega(1)$ as $T \to \infty$.

(b) **(5 points)** Let's consider the following randomized pre-processing step: independently for each action a, initialize the starting cumulative cost to a geometric random variable $-X_a$ with parameter ϵ (i.e., to the number of coin flips needed until you get "heads", assuming that the probability of "heads" is ϵ). Then, every day, you pick the action that minimizes the perturbed cumulative cost $-X_a + \sum_{s=1}^{t-1} c^s(a)$ over $a \in A$.

First prove that, for each day t, with probability at least $1 - \epsilon$, the smallest perturbed cumulative cost of an action prior to day t is at least 1 less than the second-smallest perturbed cumulative cost of an action prior to day t.

(c) **(5 points)** As a thought experiment, consider the (unimplementable) algorithm that, every day, picks the action that minimizes the perturbed cumulative cost $-X_a + \sum_{s=1}^{t} c^s(a)$ over $a \in A$, *taking into account the current day's cost vector*. Prove that the average regret of this algorithm is at most $(\max_a X_a)/T$.

(d) **(5 points)** Prove that $\mathbf{E}[\max_a X_a] = O(\epsilon^{-1} \log n)$, where n is the number of actions.

(e) **(5 points)** Use (c) and (d) to prove that, for a suitable choice of ϵ, the algorithm in (b) has expected average regret $O(\sqrt{\frac{\log n}{T}})$, just like the multiplicative weights algorithm covered in class. (Make any assumptions you want about how ties between actions are broken.)