Algorithmic Game Theory - handout5

Uriel Feige, Robert Krauthgamer, Moni Naor

3 December 2008

The library should now have a new copy of the Algorithmic Game Theory book, which will be put on the reserve shelf.

The hint in question 1 in handout 2 was misleading. This will be taken into account in the grading.

Reminder: Please remember to register to the fourth Israeli Seminar on Computational Game Theory if you want to attend it. See:

`http://pluto.huji.ac.il/~mfeldman/cgt4_2008.html`

Homework. (Please keep the answers to the following questions short and easy to read.)

1) Consider the following three player game. Player A has strategies a1 and a2, player B has strategies b1 and b2, and player C has strategies c1 and c2. The payoffs are described below. The name of a player appearing in a strategy profile means that the player gets a payoff of 1. Otherwise the payoff is 0. For example, on profile (a1,b2,c2) players A and B each gets a payoff of 1 and player C gets a payoff of 0.

	b1	b2
a1		B
a2	A	A; C

c1

	b1	b2
a1	C	A; B
a2	B; C	

c2

Equivalently, the payoff for each player can be described as follows: if a player plays his first strategy he gets a payoff of 1 iff the two other players play their second strategy. If a player plays his second strategy, he gets a payoff of 1 iff the player preceding him (in the cyclic order A-B-C-A) plays his first strategy.

Find *all* Nash equilibria of this game, and prove that no other Nash equilibrium exists. (For the proof, you may need to solve a system of algebraic equations that expresses the conditions for a profile of strategies being a Nash equilibrium.)

2) Recall that problems in PPAD are problems whose input includes an implicit description of a directed graph with at most exponentially many nodes. There is a polynomial time algorithm that given the name of a node figures out from the implicit description the edges incident with the node. Every node has at most one incoming edge and at most one outgoing edge. One is given a source node (has no incoming edge), and the goal is to find any sink node (has no outgoing edge). The *matching-sink* problem is more specific and requires one to output the sink node that lies on the end of the path of the given source node. Prove that *matching-sink* is NP-hard. (Hint: related to exhaustive search.) Remark: *matching-sink* is in fact PSPACE-complete.