Algorithmic Game Theory - handout3

Uriel Feige

21 May, 2024

Course webpage: https://www.wisdom.weizmann.ac.il/∼feige/agt2024.html

Homework assignments are an integral part of the course and will be a significant part of the grade. Please hand in the written assignments two weeks after they are given.

Teaching assistant: **Yotam Gafni**.

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

1. A pure strategy s in a two player game is said to be *dominated* if for every mixed strategy t of the other player, strategy s is not a best response with respect to t. Clearly, a dominated strategy cannot be part of a Nash equilibrium. Show that there is a polynomial time algorithm for detecting whether a two player game (given in normal form) has a dominated strategy. (Hence such strategy can be removed prior to attempting to find a Nash equilibrium.)

2. 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.

c1

	b1	b2
a1		B
a2	A	A; C

c2

	b1	b2
a1	C	A; B
a2	B; C	

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.)

3. Recall that problems in PPAD are problems whose input includes an implicit description of a directed graph $G(V, E)$ with at most exponentially many nodes. Every node has at most one incoming edge and at most one outgoing edge. There is a polynomial time algorithm that given the index of a node (between 1 and V) figures out from its index whether it has an incoming edge and whether it has an outgoing edge, and returns the indices of the other vertices at the end of these edges (if they exist). One is given a source node (a node that has an outgoing edge, but no incoming edge), and the goal is to find any other node of degree 1 (with either no incoming edge or no outging edge).

   (For example, the Lemke-Howson algorithm mentioned in class shows that computing 2-player Nash is in PPAD. Given a 2-player game in normal form, each node of the graph corresponds to two pairs of subsets of pure strategies, one subset for the row player and one subset for the column player. A directed edge from node u to v means that if the algorithm is currently considering the pair of subsets corresponding to u (checking whether there is a Nash with them as supports), then the next pair of subsets to consider (if a Nash is not found yet) will be that corresponding to v. It turns out that in the Lemke-Howsen directed graph, each node has at most on incoming edge, and moreover, from the description of the node one can figure out in polynomial time the source of the incoming 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.