CS 6840 – Algorithmic Game Theory (**??** pages) Spring 2012

Lecture 42 Scribe Notes: Complexity of Equilibrium

Instructor: Eva Tardos *Cao Ni (cn254)*

1 Overview

We begin with an overall picture of the kinds of complexity problems known. We consider classes of decision problems as follows:

- A traditional **NP** is a yes/no question. The question of whether a (mixed) Nash equilibrium *exists* is trivial: we have proven their existence in previous few lectures
- The question of *finding* the Nash is still interesting. The NP-analogue of finding something is called **FNP**. The input of an FNP is some instance x, and instead of determining if it is achievable (as does NP), in FNP we are interested in finding some output y.

2 Properties of FNP Problems

In FNP, we want to decide if there exists a polynomial function poly($\cdot$) and a polynomial-time algorithm A which, given x and y, returns $A(x, y) =$ yes if (x, y) is satisfiable, and no otherwise.

Additionally, it is required that if there exists y such that $A(x, y) =$ yes, then there also exists y' such that $|y'| \leq$ poly(x). That is, if there is a proper answer then there is also one that's sufficiently short.

Here are two examples of FNP problems:

Example 1. In a game-theoretic setting, x is a finite game and y a proposed Nash.

Example 2. In a Hamiltonian cycle problem, x can be a graph, y a cycle through all nodes.

We define the relative easiness between FNP problems as follows.

Definition. For two FNP problems $\mathcal{P}$ and $\mathcal{P}'$, we say that $\mathcal{P}$ *is an easier problem than* $\mathcal{P}'$, denoted $\mathcal{P} \prec \mathcal{P}'$, if there exists poly-computable functions $f : x \mapsto f(x)$ and $g : y' \mapsto g(y')$ for which

$$A'(f(x), y) = A(x, g(y'))$$

To solve $\mathcal{P}$, we can take input x, convert it to $f(x)$ which is subsequently used as in input to $\mathcal{P}'$, obtain the output y' through $A'(f(x), y')$ and finally use g function to convert it back to the answer to $\mathcal{P}$, $g(y')$. As f, g and A' are all polynomial-computable, so the entire procedure can be finished in polynomial time.

Can Nash be FNP complete? Probably not! Hence we are motivated to study a further subclass of FNP problems, the TFNP.

3 TFNP: "Total" FNP Problems

Definition. A *"Total" FNP (TFNP)* problem is an FNP problem where the answer is guaranteed to be yes.

Recall that we have proven the existence of (mixed) Nash equilibrium in any game. Therefore, finding the Nash is a TFNP problem.

However, proof of the "total" guarantee in general is tricky because it requires a mathematical proof that the answer is always yes.

3.1 Classes of TFNP Problems

Papadimitriou classified the following subclasses of TFNP by style of argument that guarantees positive answer to existence:

1. **PPA** (polynomial-time parity argument): the guarantee is based on a parity argument on a directed graph
2. **PPP** (polynomial-time pigeon-hole principle): the guarantee uses the pigeon-hole principle that for $f : \{1, \ldots, n\} \mapsto \{1, \ldots, n-1\}$, there must exist x, y such that $f(x) = f(y)$.
3. **PLS** (polynomial-time local search): the guarantee is based on the fact that any acyclic graph has a sink (a vertex with no outgoing edges).

3.2 Finding Nash as PPA

Our line of argument that a Nash equilibrium exists sequantially depends on

A1. find a fixed point of a function (Brouwer's fixed point theorem)

A2. find multi-colored simplex (Sperner's lemma)

A3. given a compactly represented graph with all vertices of having degree 0,1 or 2 and a vertex known to have degree 1, find another vertex with degree 1 (or a pair of vertices i, j: i is a neighbour of j but j is not a neighbour of i).

Notice the last step uses a parity argument on a graph. Hence it satisfies the PPA guarantee.

Unfortunately, the requirement that one achievable y implies the existence of a y' such that $|y'| \leq \text{poly(x)}$ means, under the context of Nash equilibrium, that if the game x has a Nash equilibrium y, then there must exist a Nash y' which is polymonial size in the game description. That is not always the case since there are games whose only Nash equilibrium invovles irrational probability distribution. Therefore, the definition of Nash equilibrium and the steps of the existence proof needs to be refined as follows

B1. An ϵ-*Nash equilibrium* in a finite game is a set of mixed strategies where each player has at most ϵ incentive to deviate.

B2. Given a polynomial-time computable function $f : \Delta_n \mapsto \Delta_n$ that is Lipschitz continuous (i.e. for some $c > 0$, $||f(x) - f(y)|| > c||x - y||, \forall x, y$) and $\epsilon > 0$, we can either

- find x such that $||f(x) - x|| \leq \epsilon$, or

- find x such that $f(x) \notin \Delta_n$, or
- find x, y such that $||f(x) - f(y)|| > c||x - y||$

B3. We have an algorithm defining a compactly represented (directed) graph: for any vertex i, the algorithm returns a list of its neighbours. Hence in step A?? we will either find a vertex of degree 1 or vertices i, j such that i is a neighbour of j but j is not a neighbour of i (a mismatch).

3.3 Nash as PPAD-complete

In last 2 classes, we have essentially shown that

$$\text{Nash} \prec \text{Brouwer} \prec \text{Sperner} \in \text{PPA}$$

The one part where our existence proof was weaker is Sperner $\in$ PPA. Sperner did correspond to a PPA problem, but degree 1 nodes corresponded to either a multicolored simplex or lower dimensional simplex on the side. We started from the side using colors $1, \ldots, n-1$. To make sure that the degree 1 node returned is a full dimensional simplex, we can add an extra layer along this side to guarantee that there is exactly one multicolored lower dimensional simplex. To do this color each vertex on this extra layer the lowest color allowed by the coloring. The only multicolored simplex on this side is the one at the corner. Further, the extra layer doesn't create full-dimensional multicolored vertices.

We are not able to prove that Nash is PPA-complete. Nevertheless, it is complete for a subclass of PPA, the PPAD.

Definition. A *Polynomial Parity Arguments on Directed graphs (PPAD)* problem is a PPA problem where each vertex has in-degree of at most 1 and out-degree of at most 1.

Analogous to step B??, we require a graph defined by an algorithm that returns next/previous vertex for any vertex i, having a node 0 with $\text{prev}(0) = \emptyset$. Our goal is to find a different vertex i such that $\text{next}(i) = \emptyset$ or $\text{prev}(i) = \emptyset$ or a mismatch.

Note: for the graph defined in the proof of Sperner's lemma in the last lecture, we can naturally define direction of an edge by keeping a certain color on the left-hand side of any directed edge connecting two simplices.

Fact: PPAD$\prec$PPA and PPAD$\prec$PPP.

In the next class, we will show that finding Nash equilibrium is PPAD-complete, and finding pure-Nash equilibria in potential games is PLS-complete.