6.896 Topics in Algorithmic Game Theory March 1, 2010

Lecture 8

Lecturer: Constantinos Daskalakis *Scribe: Alan Deckelbaum, Anthony Kim*

NOTE: The content of these notes has not been formally reviewed by the lecturer. It is recommended that they are read critically.

1 Introduction

We begin by developing a complexity theory of total search problems in FNP, by defining a few complexity classes and discussing relationships between them. We then proceed to lay down the foundations towards showing that NASH is complete for the complexity class PPAD.

2 Complexity Theory of Total Search Problems

We describe several complexity classes inside FNP. We define these classes by taking a combinatorial argument of existence (such as the parity argument or the pigeonhole principle) and defining a total problem that amounts to searching for the object whose existence is guaranteed.

2.1 PPAD

The complexity class PPAD ("Polynomial Parity Argument on Directed Graphs") was proposed by Christos Papadimitriou in 1994 [6]. Its goal is to capture the complexity of searching for an object whose existence is guaranteed by the principle that, *"A directed graph with an unbalanced node (indegree $\neq$ outdegree) must have another unbalanced node."*

Suppose that we describe an exponentially large graph with vertex set $\{0,1\}^n$, where each vertex has in-degree and out-degree at most 1 by providing two circuits, P and N. Each circuit takes as input a node id (a string in $\{0,1\}^n$) and outputs a node id (another string in $\{0,1\}^n$). We interpret our graph as having a directed edge from v_1 to v_2 iff the following two properties hold:

- $P(v_2) = v_1$
- $N(v_1) = v_2$

We can think of the circuit P as returning a "possible previous" node, and the circuit N as returning a "possible next" node. If these circuits agree (that is, if P says that v_1 is previous to v_2, and if N says that v_2 is next after v_1), then we interpret our graph as having a directed edge from v_1 to v_2. Notice that, by this formalization, any two circuits P and N mapping $\{0,1\}^n \rightarrow \{0,1\}^n$ will define some graph. Furthermore, it is important to notice that, with our characterization, we can efficiently determine both the in-neighbor and the out-neighbor (if they exist) of a given vertex v. This was the case in our proof of Sperner's lemma, where we could use local information to efficiently determine the in-neighbor and out-neighbor of a given simplex.

Inspired by the above discussion, we define the problem END OF THE LINE as follows:

Definition 1. *The problem **END OF THE LINE** is defined as follows: Given two circuits P and N as above, if 0^n is an unbalanced node in the graph, find another unbalanced node; otherwise, return "yes."*

Given this definition we can define the class PPAD as the class of all search problems that are polynomial-time reducible to END OF THE LINE:

Definition 2. *The complexity class **PPAD** is the set {search problems in FNP poly-time reducible to END OF THE LINE }.*

Remark 1. *Note that we defined PPAD in terms of its complete problem. This is in the same spirit as defining FNP as all problems poly-time reducible to SAT. There is an alternative definition of PPAD in terms of Turing Machines, but it's more cumbersome to work with and we won't present it.*

Remark 2 (Syntactic vs Semantic Classes). *Note that the END OF THE LINE problem is well-defined and guaranteed to possess a solution for all input circuits P and N that map $\{0,1\}^n$ to $\{0,1\}^n$. This is a very important property of PPAD in that it allows us to enumerate all search problems in the class. (To justify this rigorously we need to look at the Turing machine definition of the class.) The existence of such an enumeration makes our class "syntactic" allowing hardness results to be obtained.*

From our arguments from last lecture, it follows that the following reductions are possible

$$NASH \rightarrow BROUWER \rightarrow SPERNER \in PPAD \subset FNP.$$

where the symbol "$\rightarrow$" represents polynomial-time reducibility.

2.2 Other "Arguments of Existence"

The complexity class PPAD is based on the principle that *"A directed graph with an unbiased node must have another unbalanced node"*. We can define other complexity classes based on different arguments of existence. Examples of other such classes are the following:

- PPA: *"If an undirected graph has a node of odd degree, it must have another."*
- PLS: *"Every directed acyclic graph must have a sink."*
- PPP: *"If a function maps n elements to $n-1$ elements, it must have a collision."*

We now define each of the above classes formally, in a similar manner to our definition of PPAD.

2.2.1 PPA

The class PPA (polynomial-time parity arguments) is an undirected analogue of PPAD, based on the principle that "If an undirected graph has a node of odd degree, it must have another." We start by defining a problem called ODD-DEGREE-NODE. In this problem, we are given a circuit C which takes as input a node id (a string in $\{0,1\}^n$) and as output returns a set of at most d node id's. The circuit C can be thought of as outputting "possible neighbors" of the input node. We interpret two vertices v_1 and v_2 as being connected in the graph iff $v_1 \in C(v_2)$ and $v_2 \in C(v_1)$. The ODD-DEGREE-NODE problem is defined as follows:

Definition 3. ***ODD-DEGREE-NODE**: Given a circuit C as above, if 0^n has odd degree, find another vertex with odd degree. Otherwise, say "yes."*

The class PPA consists of problems which are reducible to ODD-DEGREE-NODE:

Definition 4. *The complexity class **PPA** is the set {search problems in FNP poly-time reducible to ODD DEGREE NODE }.*

2.2.2 PLS (Johnson-Papadimitriou-Yannakakis '88 [5])

The class PLS (polynomial local search) is based on the existence argument that, "Every directed acyclic graph has a sink." We define a problem FIND SINK in a manner analogous to the above. In our problem, we will have two input circuits. The circuit C takes as input a node id and returns as output a set of at most d node id's. We also have another circuit F (the "potential function") which maps a node id to a real number.

The circuits C and F together define a directed acyclic graph, where we put a directed edge from v_1 to v_2 iff both $v_2 \in C(v_1)$ and $F(v_2) > F(v_1)$. (Note that, as before, we make no real restrictions on the circuit C: it is allowed for example to have $v_1 \in C(v_2)$ but $v_2 \notin C(v_1)$). The directed graph resulting from C and F is clearly acyclic, since F increases along directed edges.

We now define the problem FIND SINK:

Definition 5. ***FIND SINK***: *Given circuits C and F as above, find a node x such that $F(x) \geq F(y)$ for all $y \in C(x)$.*

It is clear that the vertex x asked for in problem FIND SINK is a sink of the corresponding DAG.

Definition 6. *The complexity class **PLS** is the set {search problems in FNP poly-time reducible to FIND SINK }.*

2.2.3 PPP (Papadimitriou '94 [6])

The class PPP (polynomial pigeonhole principle) is based on the argument that, "If a function maps n elements to $n-1$ elements, it must have a collision." We define the following computational problem.

Definition 7. ***COLLISION***: *Given a circuit C mapping node id's to node id's, either find an x such that $C(x) = 0^n$, or find $x \neq y$ such that $C(x) = C(y)$.*

We now define the complexity class PPP:

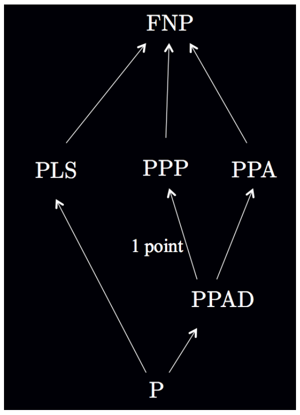
Definition 8. *The complexity class **PPP** is the set {search problems in FNP poly-time reducible to COLLISION }.*

2.3 Relationships Among Classes

We mention some known relationships among the complexity classes defined above. First, it is obvious that the complexity class PPAD is contained in PPA under polynomial-time reductions: we can take our circuits defining a PPAD graph and construct circuits for a PPA graph by simply ignoring the direction of the edges. We continue with a 1-point homework problem.

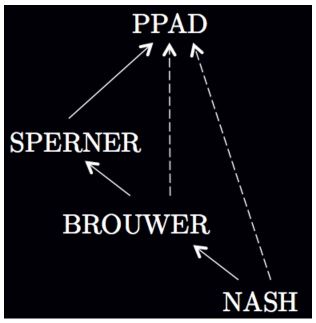
Homework (1 point): Show that the complexity class PPAD is contained in PPP under polynomial-time reductions.

Given the above, we obtain the following relationships between our classes (where an arrow represents polynomial-time reducibility):



3 Fixed-Point Computation, Nash Equilibrium, and PPAD

We have already established in Lectures 5–7 the following results (where dotted arrows represent poly-time reductions that are implied by composing reductions that we have already shown; we have also taken the liberty of using PPAD in place of END OFTHE LINE):



Our goal in the next couple of lectures is to show that the computational problems SPERNER, BROUWER, and NASH are in fact complete for the class PPAD. We do this by showing poly-time reductions in the reverse order of the solid arrows shown above.

3.1 The Masterplan

Our hardness results are established via the following program:

1. Start with the generic PPAD problem, that is END OF THE LINE, and show how to appropriately embed its graph into $[0,1]^3$. Let us name the embedded END OF THE LINE problem " 3D-END OF THE LINE". By construction 3D-END OF THE LINE and END OF THE LINE will be equivalent.
2. Reduce the 3D-END OF THE LINE problem to SPERNER in three dimensions.
3. Proceed to reduce the restricted family 3D-SPERNER instances to a family of piecewise-linear BROUWER instances.
4. Reduce the resulting class of piecewise-linear BROUWER instances to finding a Nash equilibrium of a class of multi-player games.
5. Conclude by showing that the latter is poly-time reducible the 2-player NASH problem.

Remark 3. *Reductions similar to those described in Steps 1, 2 and 3 above were given in [6]. We provide the alternative, easier to use, reductions that were given in [4]. The main barrier in our program outlined above is Step 4 of reducing the special kind of BROUWER problems arising in Step 3 to a multi-player game. This reduction was given in [4], where also a proof of Step 5 was given but with a 4-player game as the final product. This latter construction for Step 5 was improved to 3-player games independently by [1] and [3], while the final touch on Step 5 reducing to 2-player games was given in [2]. In Lecture 10 we provide an alternative construction for Step 5, that follows easily from our construction for Step 4 from [4].*

Remark 4. *The program outlined above will result in showing that BROUWER, SPERNER, and NASH are all complete for PPAD. Given this hardness result, to reduce a generic BROUWER problem to a NASH problem, we can reduce our BROUWER problem to END OF THE LINE and reduce the latter to NASH via the sequence of reductions outlined above.*

4 Reductions

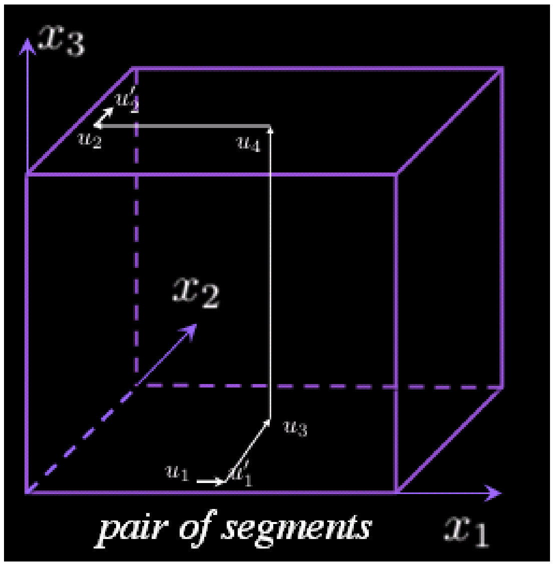
We describe the constructions for Steps 1 and 2 of our program.

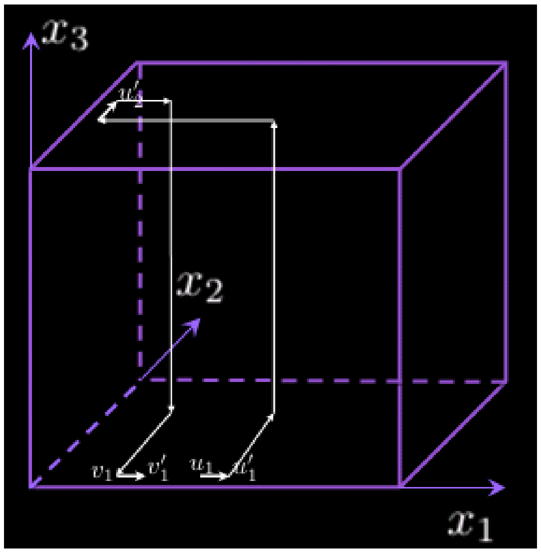
4.1 Step 1: Embedding the END OF THE LINE Graph into 3-d Geometry

We describe an embedding of the END OF THE LINE graph into $[0,1]^3$. To define this embedding, we identify a piece-wise linear, single dimensional subset of the cube corresponding to the PPAD graph. We call that subset L. If the END OF THE LINE graph has vertex set $\{0,1\}^n$, L is defined as follows:

1. Let $m = n + 4$. Divide $[0,1]^3$ into cubelets of size 2^{-m}.
2. For each non-isolated node $u \in \{0,1\}^n$, we add a *main segment* from $u_1 = (8\langle u\rangle + 2, 3, 3) \cdot 2^{-m}$ to $u_1' = (8\langle u\rangle + 6, 3, 3) \cdot 2^{-m}$, an *auxiliary* segment from $u_2 = (3, 8\langle u\rangle + 6, 2^m - 3) \cdot 2^{-m}$ to $u_2' = (3, 8\langle u\rangle + 10, 2^m - 3) \cdot 2^{-m}$, and an orthonormal path connecting the end of main segment and the beginning of auxiliary segment using $u_3 = (8\langle u\rangle + 6, 8\langle u\rangle + 6, 3) \cdot 2^{-m}$ and $u_4 = (8\langle u\rangle + 6, 8\langle u\rangle + 6, 2^m - 3) \cdot 2^{-m}$ as breakpoints. Note that $\langle u\rangle$ = (integer corresponding to u). See left diagram below.
3. For each edge between u and v, we add an orthonormal path connecting the end of the auxiliary segment of u with the beginning of the main segment of v using $(8\langle v\rangle + 2, 8\langle u\rangle + 10, 2^m - 3) \cdot 2^{-m}$ and $(8\langle v\rangle + 2, 8\langle u\rangle + 10, 3) \cdot 2^{-m}$ as breakpoints. See right diagram below.
4. For node 0^n, we perturb the corresponding main segment ; the main segment goes from $0_1 = (2, 2, 2) \cdot 2^{-m}$ to $0_1' = (6, 2, 2) \cdot 2^{-m}$ and the orthonormal path connecting the end of main segment and the beginning of auxiliary segment uses $0_3 = (6, 6, 2) \cdot 2^{-m}$ and $0_4 = (6, 6, 2^m - 3) \cdot 2^{-m}$ as breakpoints. Note that the auxiliary segment does not change.

Note that the main segment corresponding to node 0^n is placed closer to the boundary. This is for a technical reason that will be clear when we do our reduction in Step 2. In particular, it is **not** needed for this step. Call L the orthonormal single-dimensional subset of the cube defined by the above construction.





We make the following observations about our construction of L:

Claim 1. *Two points p, p' of L are closer than $3 \cdot 2^{-m}$ in Euclidean distance only if they are connected by a part of L that has length $8 \cdot 2^{-m}$ or less.*

Claim 2. *Given the circuits P and N of the END OF THE LINE instance, and a point x in the cube, we can decide in polynomial time if x belongs to L.*

Claim 3. *Node u is a sink if the given END OF THE LINE graph iff L is disconnected at u_2'. Node u is a source iff L is disconnected at u_1.*

From Claims 2 and 3, it follows that solving END OF THE LINE is equivalent to finding a point $\neq 0_1$ where L is discontinuous. Let us call this problem 3D-END OF THE LINE. The input to this problem is a pair of circuits, and the difference from END OF THE LINE is that we envision these circuits as defining an orthonormal line in $[0,1]^3$ (rather than a graph).

4.2 Reduction from 3D-END OF THE LINE to SPERNER

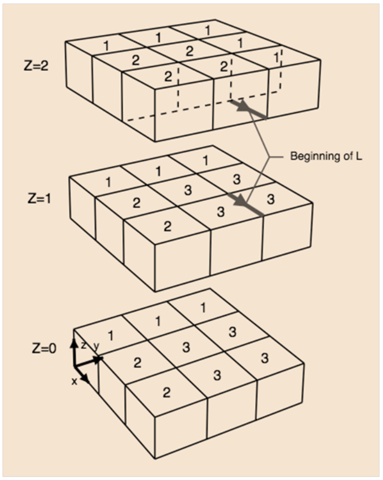
We reduce 3D-END OF THE LINE to SPERNER. Rather than coloring the vertices of the primal subdivision, that is all points whose coordinates are integer multiples of 2^{-m}, we work with the dual subdivision, coloring the centers of the cubelets of the primal subdivision. It will be easy to check that our coloring will be a legal coloring of the dual subdivision.

Let $K_{i,j,k}$ denote the center of the cubelet whose least significant corner (i.e. the corner with the smallest ℓ_1 norm) has coordinates $(i,j,k)\cdot 2^{-m}$. We color the boundary (of the dual subdivision) as follows:

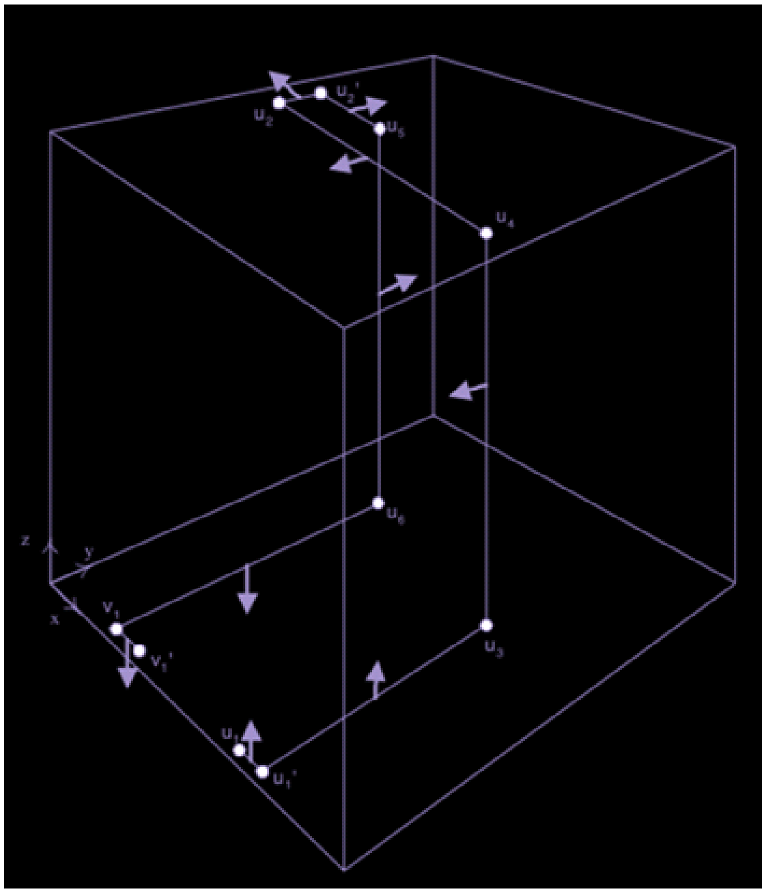
1. Color $K_{i,j,k}$ with color 0 if i, j, or k equals $2^m - 1$.
2. Of the remaining uncolored centers, color $K_{i,j,k}$ with color 1 if $i = 0$.
3. Of the remaining uncolored centers, color $K_{i,j,k}$ with color 2 if $j = 0$.
4. Of the remaining uncolored centers, color $K_{i,j,k}$ with color 3 if $k = 0$.

Clearly, the above coloring is a legal coloring for the dual graph (on centers of cubelets). Note that this coloring is **not** the envelope coloring we used in previous lectures and that the color names are permuted.

We proceed to describe the coloring of the rest of the cubelet centers (those that are not on the boundary of the dual subdivision). If a cubelet does not touch L, then its center is colored 0. The cubelets touching L at any given point are colored with the colors 1, 2 and 3 in a way that "protects" the line from color 0. More specifically, colors 1, 2, and 3 are placed in a clockwise arrangement around L for an observer who is walking on L in the direction corresponding to the direction of the END OF THE LINE graph. Of the four cubelets, two are colored with color 3, one with color 1, and the other with color 2. Note that the placement of the main segment of 0^n close to the boundary is quite convenient, as we can color the cubelets adjacent to the beginning of L in a way protecting the start of L from color 0, as shown in the figure.



Now, how do we decide which two out of the four cubelets around L are colored with color 3? And how do we color at the points where L bends? We start with the first question. Take a look at the diagram below.



The arrow points in the direction in which the two cubelets colored with color 3 lie. The diagram shows how this direction "evolves" as we proceed walking along L. Moreover, the coloring of the extra two cubelets that appear at point where L bends is implied.

We note however that there is an important directionality issue. Consider the subset of L corresponding to an edge (u, v) of the END OF THE LINE graph as in the diagram. At the main segment corresponding to u the pair of cubelets colored with color 3 lies above L, while at the main segment corresponding to v they lie below L. The change in the direction of the arrow along one edge of the END OF THE LINE graph implies that there is no consistent rule by which we can locally decide where the two cubelets colored with 3 lie. We resolve this issue by virtue of the following important observation.

Claim 4. *Wlog, we can assume that all edges (u, v) of the PPAD graph join an odd u (as a binary number) with an even v (as a binary number) or vice versa.*

Proof: We duplicate each vertex u of the PPAD graph by appending an extra bit to the node id, thus creating two copies $u0$ and $u1$. If node u is non-isolated, we include an edge from $u0$ to $u1$. Moreover, each edge (u, v) in the original PPAD graph now connects $u1$ to $v0$. The resulting END OF THE LINE instance satisfies the desired property and is clearly as hard to solve. $\square$

Given Claim 4, we assume that all edges (u, v) of the PPAD graph join an odd u (as a binary number) with an even v (as a binary number) or vice versa. Hence we can resolve the directionality issue with the following convention: if u is even we place the pair of 3-colored cubelets below the main segment of u, while if u is odd we place it above. To decide where the two cubelets colored 3 lie at any given point of L, we compute the pair of main and auxiliary segments that this part of L joins, and decide the location of the 3-colored cubelets by appealing to the above diagram (corresponding to an edge from an odd to an even node) or its reverse. This convention resolves the directionality issue and agrees with the coloring around the main segment of 0^n.

For convenience let us introduce the concept of a panchromatic node.

Definition 9. *A point x in the unit cube is* panchromatic *iff it is the corner of some cubelet (i.e. it belongs to the primal subdivision of multiples of 2^{-m}), and all colors are present in the cubelets containing x.*

We make the following observations:

Claim 5. *A point in the cube is panchromatic in the described coloring iff it is:*

1. *the endpoint u_2' of the auxiliary segment of a sink vertex u of the END OF THE LINE graph, or*
2. *the endpoint u_1 of the main segment of a source vertex $u \neq 0^n$ of the END OF THE LINE graph.*

Claim 6. *Given the description (P, N) of the 3D-END OF THE LINE instance, there is a polynomial-size circuit computing the coloring of each center $K_{i,j,k}$.*

From Claims 5 and 6, the reduction of the 3D-END OF THE LINE problem to SPERNER follows. The only thing we need to justify is that panchromatic nodes correspond to panchromatic simplices of the dual subdivision, which is easy to check.

References

[1] Xi Chen and Xiaotie Deng. 3-NASH is PPAD-Complete. *Electronic Colloquium on Computational Complexity,* 134, 2005.

[2] Xi Chen and Xiaotie Deng. Settling the Complexity of 2-Player Nash-Equilibrium. *FOCS* 2006.

[3] Constantinos Daskalakis and Christos H. Papadimitriou. Three-Player Games Are Hard. *Electronic Colloquium on Computational Complexity,* 139, 2005.

[4] Constantinos Daskalakis, Paul W. Goldberg and Christos H. Papadimitriou. The Complexity of Computing a Nash Equilibrium. *STOC 2006.* Journal version in the *SIAM Journal on Computing*, 39(1): 195–259. 2009. Simplified version in the *Communications of the ACM,* 52(2): 89–97, 2009.

[5] David S. Johnson, Christos H. Papadimitriou and Mihalis Yannakakis. How Easy is Local Search? *Journal of Computer Systems Science*, 37(1): 79–100, 1988.

[6] Christos H. Papadimitriou. On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence. *Journal of Computer Systems Science*, 48(3): 498–532, 1994.