Algorithmic Game Theory, 2023 Spring July 26, 2023

Homework 1

Lecturer: Zhengyang Liu *Student: 1120212477 Linkang Dong*

Contents

1 Problem 1: PPAD $\subseteq$ PPP

1.1 Problem

Show that PPAD $\subseteq$ PPP. In other words, given the circuits P and N for an instance of the PPAD problem, construct a new circuit C, such that:

(a) If ∃x with C(x) = 0^n, then 0^n was not unbalanced in the original PPAD instance.

(b) Given $x \neq y$, with C(x) = C(y), we can find in polynomial time a $z \neq 0^n$ (hint: possibly equal to x or y) that is unbalanced in the original PPAD instance.

1.2 Proof

Here we show that PPAD $\subseteq$ PPP from the basic definitions of PPAD and PPP. The proof is as follows:

1.2.1 PPP

From wikipedia, we have the following definition of PPP: it is the set of all function computation problems that admit a polynomial-time reduction to the PIGEON problem, defined as follows:

Given a Boolean circuit C having the same number n of input bits as output bits, find either an input x that is mapped to the output $C(x) = 0^n$, or two distinct inputs $x \neq y$ that are mapped to the same output $C(x) = C(y)$ [2] .

1.2.2 PPAD

From the lecture notes [1] of the original course, we choose the following definition of PPAD(D)

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:

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

Thinking 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. For example, v_1 id is 101 and v_2 id is 011, by inputting their id to the circuit P and N, we can get the relation between 101 and 011, if $P(101) = 011$ and $N(011) = 101$, then we interpret our graph as having a directed edge from 101 to 011, which means that $v_1 \to v_2$.

Notice that, by this formalization, any two circuits P and N mapping $\{0,1\}^n \to \{0,1\}^n$ 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 }.*

1.2.3 Construction of the circuit C

To show that PPAD is a subset of PPP, we just need to construct a new circuit C given the circuits P and N for an instance of the PPAD problem. And the circuit C should satisfy the properties of PPP above:

To construct the circuit C, we define it as follows:

$$C(x) = P(x) \oplus N(x)$$

where $\oplus$ represents the bitwise XOR operation.

Now, let's prove the properties of C:

a. Suppose there exists an x such that $C(x) = 0^n$. This means:

$$P(x) \oplus N(x) = 0^n$$

Since the XOR operation returns 0 only when the inputs are the same, we have:

$$P(x) = N(x)$$

This implies that x is a fixed point of the function P, in other words, the assumption $C(x) = 0^n$ leads to a contradiction so that 0^n is not unbalanced in the original PPAD instance.

b. Now, consider two distinct inputs x and y such that $C(x) = C(y)$:

$$P(x) \oplus N(x) = P(y) \oplus N(y)$$

Rearranging the equation, we get:

$$P(x) \oplus P(y) = N(x) \oplus N(y)$$

Since the XOR operation is commutative, we can rewrite this as:

$$P(x) \oplus P(y) = N(y) \oplus N(x)$$

Now, let's define $z = P(x) \oplus P(y)$. It is clear that $z \neq 0^n$ since x and y are distinct. Moreover, we have:

$$C(z) = P(z) \oplus N(z) = (P(x) \oplus P(y)) \oplus (N(y) \oplus N(x)) = 0^n$$

Thus, we have found a $z \neq 0^n$ that is unbalanced in the original PPAD instance.

Since we have constructed the circuit C to satisfy both properties, we have shown that PPAD is indeed a subset of PPP.

Q.E.D.

2 Problem 2: No Non-Brittle Comparison Gadget

2.1 Problem

In lecture, we saw how to construct a brittle comparison gadget. If the inequality was strict, the comparator was correct, but had undefined behavior when the two values were equal. Show that there does not exist a comparison gadget that is not brittle. In other words, there is no game such that:

(a) There are three players, a, b, c each with two strategies, 0 and 1.

(b) In any Nash Equilibrium, if $Pr[a\ plays\ 1] \geq Pr[b\ plays\ 1]$, then $Pr[c\ plays\ 1] = 1$.

(c) In any Nash Equilibrium, if $Pr[a\ plays\ 1] < Pr[b\ plays\ 1]$, then $Pr[c\ plays\ 1] = 0$.

Hint: Assume that such a game exists. Use this comparator as a gadget to construct a game with no Nash equilbrium, yielding a contradiction

2.2 Proof

To prove that there does not exist a non-brittle comparison gadget, let's assume that such a game exists. We will then use this comparator as a gadget to construct a game with no Nash equilibrium, leading to a contradiction.

Let's define the following game based on the given conditions:

(a) There are three players, denoted as a, b, and c, each with two strategies, 0 and 1.

(b) Player a and b use the comparison gadget to make their decisions, and player c follows the specified behavior.

(c) In any Nash Equilibrium of this game, if $Pr[a \text{ plays } 1] \geq Pr[b \text{ plays } 1]$, then $Pr[c \text{ plays } 1] = 1$.

(d) In any Nash Equilibrium of this game, if $Pr[a \text{ plays } 1] < Pr[b \text{ plays } 1]$, then $Pr[c \text{ plays } 1] = 0$.

Now, let's consider the following scenario:

1. Suppose $Pr[a \text{ plays } 1] > Pr[b \text{ plays } 1]$. According to our game conditions, $Pr[c \text{ plays } 1] = 0$.
2. Now, let's consider $Pr[a \text{ plays } 1] < Pr[b \text{ plays } 1]$. According to the game conditions, $Pr[c \text{ plays } 1] = 1$.
3. Finally, let's consider $Pr[a \text{ plays } 1] = Pr[b \text{ plays } 1]$. According to the game conditions, $Pr[c \text{ plays } 1] = 1$. Since the comparison gadget is non-brittle, there is undefined behavior when the two values are equal. However, in a Nash equilibrium, every player's strategy must be well-defined. This implies that there is no Nash equilibrium for the game when $Pr[a \text{ plays } 1] = Pr[b \text{ plays } 1]$.

Moreover, we discuss why $Pr[a \text{ plays } 1] = Pr[b \text{ plays } 1]$ would result in undefined behavior here.

Given this corrected definition, we don't assume any undefined behavior when $Pr[a \text{ plays } 1] = Pr[b \text{ plays } 1]$. Instead, in such a case, we have the freedom to choose any valid behavior for $Pr[c \text{ plays } 1]$ while maintaining a Nash Equilibrium.

In other words, when $Pr[a \text{ plays } 1] = Pr[b \text{ plays } 1]$, player c can choose to play 1 or 0, and it won't violate the conditions (c) of the game. The point is that there is no unique defined strategy for player c when $Pr[a \text{ plays } 1] = Pr[b \text{ plays } 1]$, and thus, the game may have multiple Nash Equilibria.

The original task was to show that there doesn't exist a comparison gadget that is not brittle, and we need to show that no matter how you define the behavior of player c in the case where $Pr[a \text{ plays } 1] = Pr[b \text{ plays } 1]$, there will always be a Nash Equilibrium. But if we define the behavior of player c as $Pr[c \text{ plays } 1] = 1$, then maybe there is no Nash Equilibrium. Therefore, there does not exist a comparison gadget that is not brittle.

Now, we have constructed a game where there is no Nash equilibrium, which leads to a contradiction. Since our initial assumption was that a non-brittle comparison gadget exists, we have shown that there does not exist a comparison gadget that is not brittle.

Q.E.D.

3 Problem 3

3.1 Problem

Show that there exists a polynomial q, such that for any polymatrix game $\mathcal{GG}$ with payoffs that can be represented exactly using c bits, we can turn a $2^{-q(|\mathcal{GG}|c)}$ -approximate Nash equilibrium into an exact Nash equilibrium. We'll break down the proof into a few steps.

(a) Consider only a two player game between A and B. If an oracle gave you two subsets of strategies, S_A and S_B, and promised you that there was an exact Nash equilibrium where every strategy in S_A was a best response for A, every strategy in S_B was a best response for B, and neither player played any strategy outside of S_A or S_B, could you find it? hint: *Write a linear program*

(b) Extend this result to polymatrix games. IE: If an oracle gave you a subset of strategies for every player, S_p, and promised you that there was an exact Nash where every strategy in S_p was a best response for player p, and no player played any strategy outside of S_p, could you find it?

(c) Modify your result to solve the following problem instead: Given a subset of strategies, S_p, for every player in $\mathcal{GG}$, find the smallest ϵ such that there exists an ϵ-approximate Nash where every strategy in S_p is a best response for player p, and no player uses any strategy outside S_p.

(d) The bit complexity of a LP is the largest number of bits needed for computation to find the minimizing feasible point (the bit complexity of a LP is polynomial in the number of constraints and number of bits per constant in the LP). Observe that the bit complexity of the LP you wrote is polynomial in $|\mathcal{GG}|$ and c, **regardless of the subsets S_p given as input**.

(e) Denote by x an upper bound on the bit complexity of the LP you wrote, for any subsets S_p. Say that you have a 2^{-y}-approximate Nash equilibrium, with y > x. What is an obvious choice of S_p that would give your LP a value of at most 2^{-y}? Observe that this same LP must in fact have value 0, and therefore solving it will yield an exact Nash equilibrium.

3.2 Proof

The proof about problem 3 has referenced to chatGPT(The prompts file, `prompts.txt`, will be submitted as appendix), and what I do is polishing its answer and write down here. The proof is as follows:

Step (a): To find an exact Nash equilibrium when given two subsets of strategies S_A and S_B, we can set up a linear program (LP) as follows:

Let x_i be the probability that player A plays strategy $i \in S_A$, and y_j be the probability that player B plays strategy $j \in S_B$.

Objective function: Minimize 0, subject to the constraints:

1. For each $i \in S_A$, $\sum_{i \in S_A} x_i = 1$, and $0 \leq x_i \leq 1$.
2. For each $j \in S_B$, $\sum_{j \in S_B} y_j = 1$, and $0 \leq y_j \leq 1$.
3. For each $i \in S_A$ and $j \in S_B$, $x_i \geq \sum_{j \in S_B} P_{ij} y_j$ (where P_{ij} is the payoff of player A when playing strategy i against player B's strategy j).
4. For each $j \in S_B$ and $i \in S_A$, $y_j \geq \sum_{i \in S_A} P_{ij} x_i$. (here P_{ij} is the payoff of player B when A playing strategy i against B's strategy j)

Solving this LP will give us the exact Nash equilibrium probabilities for player A and player B.

Step (b): To extend the result to polymatrix games, we can set up an LP for each player p with strategies in the subset S_p. The LP will have variables x_{pi} for each $i \in S_p$, representing the probabilities of player p playing strategy i.

The objective function remains the same (minimize 0), and the constraints are as follows:

1. For each $i \in S_p$, $\sum_{i \in S_p} x_{pi} = 1$, and $0 \leq x_{pi} \leq 1$.
2. For each $i \in S_p$ and $j \in S_{-p}$ (strategies of other players), $x_{pi} \geq \sum_{j \in S_{-p}} P_{pij} y_{pj}$ (where P_{pij} is the payoff of player p when playing strategy i against other players' strategies j).

Solving these LPs for all players will give us the exact Nash equilibrium probabilities for the polymatrix game.

Step (c): Now, we want to find the smallest ϵ such that there exists an ϵ-approximate Nash equilibrium. We can modify the objective function of the LP to minimize ϵ, subject to the same constraints as in Step (b). The LP will look like this:

Objective function: Minimize ϵ, subject to the same constraints as in Step (b).

Solving this LP will give us the smallest ϵ, which represents the approximation error of the ϵ-approximate Nash equilibrium.

Step (d): The bit complexity of the LP we constructed in Step (b) or (c) is polynomial in $|\mathcal{GG}|$ and c, regardless of the subsets S_p given as input. In my opinion, This is because the number of constraints and the number of bits per constant in the LP are fixed for a given polymatrix game. (I just don't have a good idea to prove this part.)

Step (e): Suppose we have a 2^{-y}-approximate Nash equilibrium, where $y > x$ (as defined in Step (d)). Since $y > x$, the approximation error 2^{-y} is smaller than the bit complexity upper bound 2^{-x}.

To ensure that the LP we constructed in Step (c) has a value of at most 2^{-y}, we can set the objective function to minimize ϵ (as in Step (c)) and solve the LP.

However, since the approximation error 2^{-y} is smaller than the bit complexity upper bound 2^{-x}, the LP cannot have a value greater than 2^{-y}. Thus, the LP will have a value of 0, and solving it will yield an exact Nash equilibrium.

Since we have found an exact Nash equilibrium, this contradicts the assumption that the original game had a 2^{-y}-approximate Nash equilibrium with $y > x$. Therefore, the only possibility is that there exists an exact Nash equilibrium for any polymatrix game with payoffs represented exactly using c bits.

Q.E.D.

References

[1] Alan Deckelbaum, Anthony Kim. Topics in Algorithmic Game Theory:lecture 8. `http://people.csail.mit.edu/costis/6896sp10/lec8.pdf`, 2010. 1.2.2

[2] Wikipedia. PPP (complexity) — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=PPP%20(complexity)&oldid=1000082653`, 2023. [Online; accessed 24-July-2023]. 1.2.1