

# CS386 Assignment 1

Due: Tuesday 9/22 10 PM ET

**How this homework works:** each section below offers more problems than you are required to complete. Read through all of them, then answer the number specified at the top of each section.

1. **Dominant Strategies and Pure Nash Equilibria.** *Answer 2 of the 4 problems below.*

Unless stated otherwise, every matrix below is a **payoff matrix** (higher numbers are better for that player) and each cell shows (Player 1's payoff, Player 2's payoff). For each matrix, (a) state whether either player has a dominant strategy, (b) find all pure Nash equilibria (if any), and (c) briefly justify your answer.

(a)

|             | Player 2: A | Player 2: B |
|-------------|-------------|-------------|
| Player 1: A | 2, 2        | 6, 1        |
| Player 1: B | 1, 6        | 4, 4        |

(b)

|             | Player 2: A | Player 2: B |
|-------------|-------------|-------------|
| Player 1: A | 3, 2        | 0, 0        |
| Player 1: B | 0, 0        | 2, 3        |

(c)

|             | Player 2: A | Player 2: B |
|-------------|-------------|-------------|
| Player 1: A | 4, 4        | 0, 3        |
| Player 1: B | 3, 0        | 3, 3        |

(d)

|             | Player 2: A | Player 2: B |
|-------------|-------------|-------------|
| Player 1: A | 5, 5        | 7, 2        |
| Player 1: B | 2, 7        | 3, 3        |

2. **Mixed Nash Equilibria and Support Enumeration.** *Answer both.*

(a) Determine all Nash Equilibria (including mixed).

|             | Player 2: A | Player 2: B |
|-------------|-------------|-------------|
| Player 1: A | 3, -3       | -1, 1       |
| Player 1: B | -2, 2       | 2, -2       |

- (b) Consider the following  $3 \times 3$  **payoff matrix** (cells are (Player 1's payoff, Player 2's payoff)):

|             | Player 2: A | Player 2: B | Player 2: C |
|-------------|-------------|-------------|-------------|
| Player 1: A | 1, 4        | 2, -2       | 3, -1       |
| Player 1: B | -1, -1      | -4, 5       | 5, 0        |
| Player 1: C | 3, -2       | 0, -4       | 3, -1       |

- i. Verify that neither player has a dominant strategy.
- ii. Verify that this game has no pure Nash equilibrium.
- iii. Find the (unique) mixed Nash equilibrium, in which both players mix over all three strategies. Set up the full system of indifference equations for each player and solve it.

### 3. Existence, Computability *Answer 2 of the 3*

- (a) Explain the difference between “an equilibrium exists” and “an equilibrium can be found efficiently,” using Nash’s theorem and PPAD-completeness as your two reference points. Why doesn’t proving existence tell you anything about how hard the search is?
- (b)
  - i. Describe an algorithm for computing all *pure* Nash Equilibria for a given game involving  $n$  players, where player  $i$  has  $m_i$  strategies available to them. Determine its runtime.
  - ii. Do the same for computing all Nash Equilibria (including mixed) for a given game involving **2 players, where player 1 has  $m$  strategies available to them and player has  $n$ .** Determine its runtime.

Note that I’m not asking for the best possible algorithms here - the ‘obvious’ brute force is fine by me.

- (c) Nash’s existence theorem guarantees an equilibrium for any game with **finitely many players**, each with **finitely many pure strategies**. Both finiteness conditions are doing real work in the proof; drop either one and the guarantee can fail.

Construct an explicit game that violates *one* of these two finiteness conditions (either infinitely many pure strategies for at least one player, or infinitely many players) and has **no Nash equilibrium at all**. Then show that no equilibrium exists.

### 4. Proofs. *Answer 2 of the 5 problems below.*

- (a) Consider the following algorithm: successively delete dominated strategies by each player. If the matrix is ever reduced to a single cell, ie a single strategy for each player, declare that strategy a Nash Equilibrium.  
First, observe that this algorithm works for the Prisoners’ Dilemma. Second, observe that if a game lacks dominant strategies, then this algorithm won’t be too useful. Lastly, decide whether this algorithm works in general: that is, if it reduces the game to a single cell, is that cell necessarily a Nash Equilibrium in the original game? If it is, explain why, otherwise give a specific game counter example.
- (b) Consider the following algorithm for a 2-player game given in normal form: pick any starting cell. Alternating between players, each player moves the currently selected cell by selecting the cell corresponding to the *best response* assuming the other player plays the strategy according to the currently selected cell. Determine a payoff matrix of a game

where this algorithm will not terminate despite there being a pure Nash Equilibrium.  
**Hint: I am pretty sure you need at least 3 rows and columns for this. My solution has 3 strategies for each player.**

- (c) Show that if  $s^* = (s_1^*, \dots, s_n^*)$  is a strategy profile in which *every* player  $i$ 's strategy  $s_i^*$  is a dominant strategy for player  $i$ , then  $s^*$  is a pure Nash equilibrium.
  - (d) Show that if  $s^* = (s_1^*, \dots, s_n^*)$  is a mixed Nash equilibrium and pure strategies  $a$  and  $b$  are both in player  $i$ 's support (each played with strictly positive probability under  $s_i^*$ ), then player  $i$ 's expected payoff from playing  $a$  against  $\sigma_{-i}^*$  (everyone else's strategies) equals their expected payoff from playing  $b$  against  $\sigma_{-i}^*$ .
  - (e) Show that if strategy  $a$  is strictly dominated for player  $i$  (there exists another strategy  $a'$  for player  $i$  that yields strictly higher payoff against *every* possible strategy profile of the other players), then  $a$  cannot be played with positive probability in any mixed Nash equilibrium of player  $i$ 's.
5. **I really wanted to include the following question and was trying to figure out how to write it in a way that was do-able by everyone. I failed, so I'm making this one purely optional. That being said, I think it's a good question that sort of addresses why **PPAD** is a thing and why the class **NP** isn't really the right complexity class to consider when talking about computing Nash Equilibria. If you do this problem and come talk to me about it I'll think you're cool.**

#### **Nash Equilibria: In NP, but (Probably) Not NP-Complete.**

At some point, you may have heard about the problem **P** vs **NP**. Fundamentally, this problem comes down to the following: if I can **verify** a solution to a problem quickly (in polynomial time), can I **construct** a solution quickly? To more formally define these classes, I'll include a couple definitions:

- Both **P** and **NP** are classes of *decision problems*. A decision problem is a problem where the output is simply yes or no.
- **P** consists of the problems where I can correctly decide yes or no in a polynomial amount of time. Some examples include:
  - SORTED =  $\{(x_1, x_2, \dots, x_n) \mid x_1 \leq x_2 \leq \dots \leq x_n\}$  - that is, the problem of determining whether a given list of numbers is sorted.
  - SUM =  $\{(x_1, x_2, \dots, x_n, T) \mid \sum_{i=1}^n x_i = T\}$  - that is, the problem of determining whether a given list of numbers sums to a given target number.

Basically, “easy” problems.

- **NP** consists of the problems where I can *verify solutions* in polynomial time. Some examples include:
  - SUBSETSUM =  $\{(x_1, \dots, x_n, T) \mid \text{there exists an } S \subseteq \{x_1, \dots, x_n\} \text{ such that } \sum_{x \in S} x = T\}$  - that is, the problem of determining whether there is a subset from a given list of numbers that sums to a given target value.
  - SAT =  $\{F(x_1, \dots, x_n) \mid F \text{ is a satisfiable expression over boolean variables}\}$  - that is, the problem of determining whether a given boolean expression (eg  $x_1 \wedge (x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_3)$ ) has an assignment of true/false to each variable that makes the expression evaluate to true.

Note that I can *verify* a proposed solution to an instance of SUBSETSUM easily - that is, if someone tells me which subset supposedly sums to  $T$ , then I can just check whether or not that subset does indeed sum to  $T$  quickly (this is just the SUM problem above).

- **Co-NP** consists of problems whose *complements* are in **NP**. For example, something like  $\text{CO-SUBSETSUM} = \{ \langle x_1, \dots, x_n, T \rangle \mid \text{there does not exist an } S \subseteq \{x_1, \dots, x_n\} \text{ such that } \sum_{x \in S} x = T \}$ .
- We say a problem is **NP-Hard** if the problem is so hard that a polynomial time algorithm for it implies a polynomial time algorithm for *every* problem in **NP**. This implication is called a *reduction* - if a polytime algorithm for **B** induces a polytime algorithm for **A**, then we say **A** is polytime reducible to **B**.
- We say a problem is **NP-Complete** if it is both in **NP** and **NP-Hard**. Establishing that problems are **NP-Complete** is certainly not trivial. However, both the problems I listed above (**SUBSETSUM**, **SAT**) are both known to be **NP-Complete**.

It is known that  $\mathbf{P} \subseteq \mathbf{NP}$  (if I can construct a solution in polytime, surely I can verify one in polytime), but the (literally) million dollar question is whether or not  $\mathbf{P} = \mathbf{NP}$ . It is widely believed that  $\mathbf{P} \neq \mathbf{NP}$ : just think about **SUBSETSUM** - it just seems very hard to somehow find the ‘right’ subset (of which there are  $2^n$ ) in a polynomial amount of time, if one even exists! It is also separately widely believed that  $\mathbf{NP} \neq \mathbf{co-NP}$ . This is maybe a little less obvious. One way to think about what **NP** is categorizing is ‘short’ (polynomial) proofs of ‘correctness’. But think about **SUBSETSUM** and **CO-SUBSETSUM**: if **CO-SUBSETSUM** were in **NP**, then there would be a ‘short’ proof that **no** subset sums to the target value. This seems quite hard to do. Anyway, let’s try and see where this whole calculation of Nash equilibria falls into this mess.

- Given a game  $G$  and a fully specified candidate strategy profile  $\sigma$  (a probability distribution over strategies for each player, with rational entries), explain why deciding whether  $\sigma$  is a Nash equilibrium of  $G$  can be done in **polynomial time** in the size of  $G$  and  $\sigma$ . (No new ideas needed: just state clearly what you’d check, and why each check is fast (polynomial time).) This shows that the problem of computing Nash Equilibria is sort of in **NP**. (I say sort of because it’s not exactly a decision problem)
- Now, let’s make a short hand-wavy argument for why finding a Nash equilibrium is (almost certainly) *not* **NP-hard**, and hence not **NP-complete**. Suppose it were: that is, suppose there’s a polynomial-time *deterministic* algorithm  $\mathcal{A}$  that decides some **NP-complete** language  $L$ , where  $\mathcal{A}$  is allowed to call, as a subroutine, some mystical greater-power  $\mathcal{O}$  that takes any game as input and returns *some* Nash equilibrium of it and  $\mathcal{A}$  makes only polynomially many calls to it. (Note that  $\mathcal{O}$  always returns an answer, since by Nash’s theorem an equilibrium always exists — it never says “no such thing.” I’m also assuming this mystical great-power  $\mathcal{O}$  takes at most a polynomial amount of time to respond to us)
  - Explain why every response  $\mathcal{O}$  gives, during any run of  $\mathcal{A}$ , can itself be verified in polynomial time to really be a Nash equilibrium of the game it was asked about. (Use part (a).)
  - Consider the full **transcript** of a run of  $\mathcal{A}$  on some input  $x$ : the sequence of games  $\mathcal{A}$  queries  $\mathcal{O}$  with, together with  $\mathcal{O}$ ’s actual responses on that run. Explain why this transcript has polynomial size, and why a verifier, given  $x$  and this transcript, can check in polynomial time that every response really is a Nash equilibrium of the game it answers.
  - Explain how we can use the above transcripts to show that  $\bar{L}$  (the complement of  $L$ ) is *also* in **NP** (which immediately gives that  $L$  is in **co-NP**).

- iv. Since  $L$  is **NP**-complete, every language in **NP** polynomial-time reduces to  $L$ . Combine this with part (iii) to conclude  $\mathbf{NP} \subseteq \mathbf{co-NP}$ , and hence (by a symmetric argument on complements)  $\mathbf{NP} = \mathbf{co-NP}$ .

So what's the take-away? We already discussed in class why it seemed very unlikely that computing Nash Equilibria could be done in polynomial time. But what's interesting is that the problem doesn't appear to be 'so hard' that it's on the same bar of difficulty as **NP**-Complete problems such as SUBSETSUM, because then we would get  $\mathbf{NP} = \mathbf{co-NP}$  which seems quite morally wrong. Anyway, if you got this far, feel free to come entertain me and chat with me about it.