Game Theory and Complexity

Created	@Nov 18, 2020 8:05 AM
Tags	Playing Games for Algo

Game Theory

Game Theory serves as a good formalism to study a certain category of games (mostly non-cooperative multi-player games). It is very well studied field and we shall start of with it and then move forward to other formalisms and study more categories of games like simulations, puzzles, their video game counterparts as well as multi-player video games.

Introduction

> An equilibrium is not always an optimum; it might not even be good.

One very important common motivation of both Game Theory and Computer Science is to be able to **model rationality** and solve cognitive tasks such as **negotiation**.

In this module, we shall introduce Game Theory with a Complexity Theory minded approach. As it turns out, Game Theory has given rise to several interesting challenges with respect to computational . complexity especially in the realm of the complexity classes PPAD and PLS.

Prisoner's Dilemma and Nash Equilibrium

In the movie 'A Beautiful Mind', there is a memorable scene where we find Russell Crowe playing Dr. John Nash say that Adam Smith's statement - "In competition, individual ambition serves the common good" - was incomplete. Instead, Nash postulates that the best result is for everyone in the group doing what's best for himself and the group. This conversation serves as a very informal introduction to the idea behind Nash Equilibrium. Here, we shall now formalise it starting with an example.

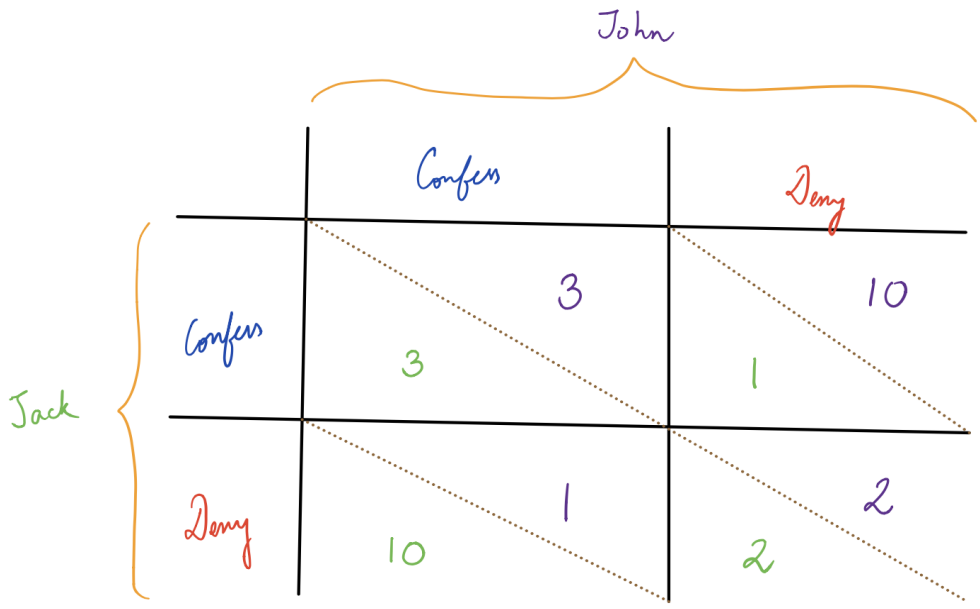
The Prisoner's Dilemma

Suppose that Jack and John are two two individuals charged and convicted with a minor case of money laundering. They both have to serve 2 years of prison each. However, the judge has a suspicion that both of them (they are acquaintances) are involved in some armed burglary (serious felony).

Now, the judge has no evidence present with him at hand. So, he proposes the following to both of them:

- if both of you deny involvement in the burglary, then both of you receive only 2 years of prison each
- if one confesses while the other denies, then the one who confessed gets 1 year while the other gets 10 years of prison
- if both of you confess, then both of you receive 3 years of prison each

Assuming that the Jack and John have no layalty amongst themselves, we should observe that they will pick a non-optimal scenario.



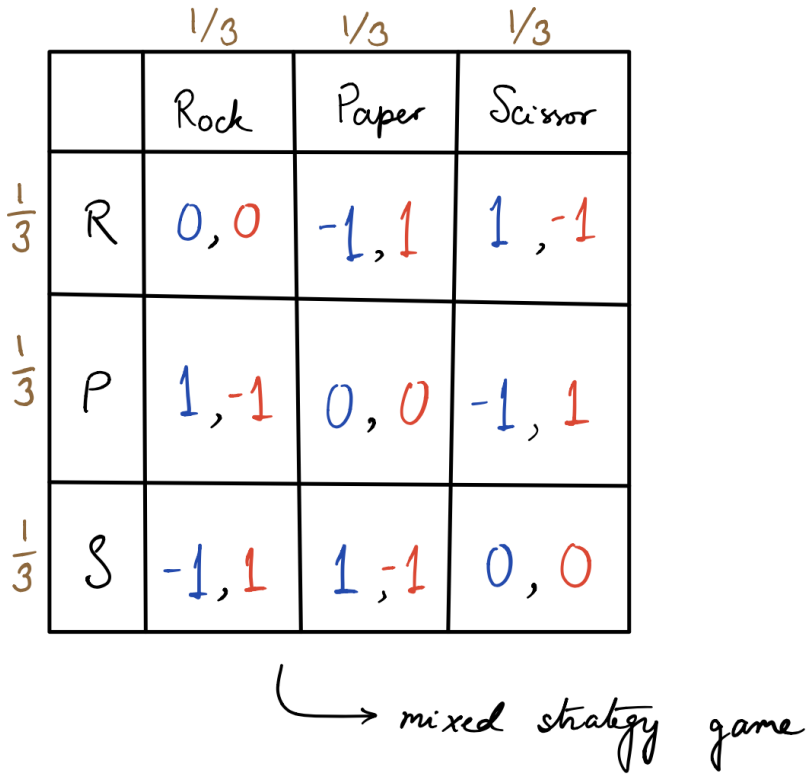
Here, there exists a global optimum in the case where both the prisoners lie. However, given our predisposed knowledge of the situation it might not be the best rational choice for the prisoners.

The best rational choice for the prisoners would be a sub-optimal choice presented in the case where both of them confess. The case serves as a **sub-optimal stable equilibrium** and is referred to as the Nash Equilibrium.

Rock-Paper-Scissors

Let us try the above payoff matrix method shown above for the popular game "Rock-Paper-Scissors".

If you have played the game for a lot of times, you may already have a good intuition as to which scenario will give rise to a Nash Equilibrium.



Given the payoff matrix, it is evident that only when both players randomize with equal probability, we obtain a Nash Equilibrium.

Interestingly, this game also serves as a proof by contradiction for the statement that not all games have a Pure Nash Equilibrium. So what's a Pure Nash Equilibrium? Let's find out.

Games: Definitions and Notations

Games can be defined with a set of **players**, where each player has his own set of allowed **actions** (known as **pure strategies**). Any combination of actions will result in a numerical **payoff** (specified by the game) for each player.

Let the number of players be $1, 2, 3, ..., k$ and S_i denote player i's set of actions. n denotes the size of the largest S_i. If k is a constant we look for algorithms that are $O(n^\kappa)$ where $\kappa \in \mathbb{N}$.

Let us look at the above statements in the light of the game of Rock-Paper-Scissors which was discussed above.

- players, $p = \{1, 2\}$
- $\forall i \in p,\ S_i = \{\text{rock}, \text{paper}, \text{scissor}\}$
- Here, of course $k = 2$, which actually is one of the most studied special case in game theory.
- $n = 3$

Let $S = S_1 \times S_2 \times S_3 \times ... \times S_k$. Then, S is the set of **pure strategy profiles**. Then, each $s \in S$ gives rise to payoff to each player and u_s^i denotes the payoff to player i when all players choose s.

- The list of all possible u_s^i's yields a **normal-form game**, which for a k-player game should be a list of kn^k numbers.
- We have have other models too. For example, a **Bayesian game** is one where u_s^i can be a probability distribution over i's payoff.

Nash Equilibrium

A Nash Equilibrium is a set of strategies - one strategy per player - such that no player has an incentive to unilaterally change his strategy (stable sub-optima). In case of two player zero-sum games, Nash Equilibrium is also optimal.

- **Pure** NE: where each player chooses a pure strategy
- **Mixed** NE: where for each player a probability distribution over his pure strategies is applied (in case of MNE we have expected payoff's associated with each player)

Formal Definition of Nash Equilibrium

From the above, $u_s^i = u_i(s)$ where $s \in S$.

An **individual mixed strategy** is a probability distribution on the set of available strategies. Such as in the last example, selecting one of "rock", "paper", or "scissors" uniformly at random is an example of a mixed strategy.

A **pure strategy** is simply a **special case** of a mixed strategy, in which one strategy is chosen 100% of the time.

- A **Nash equilibrium** is a strategy profile $s = (s_1, s_2, \ldots, s_n)$ with the property that,

$$u_i(s) \geq u_i((s_1, s_2, \ldots, s_i', \ldots, s_n))$$

$\forall i$, where $s_i' \in S_i$ denotes a strategy other than s_i available to player i.

In the event that this inequality is strict, $u_i(s) > u_i((s_1, s_2, \ldots, s_i', \ldots, s_n))\ \forall i$, the profile s is called a **Strong Nash Equilibrium.** Otherwise, s is called a **Weak Nash equilibrium**.

Existence Theorem

Any game with a finite set of players and finite set of strategies has a Nash Equilibrium of Mixed Strategies or MNE.

- Nash's existence theorem guarantees that as long as S_i is finite $\forall i$ and there are a finite number of players (p), at least one Mixed Strategy Nash equilibrium exists.

Some Computational Problems

First Decision Problem

Input: A game in normal form.

Question: Is there a Pure Nash Equilibrium?

Special Case Solutions:

- Determining whether a strategic game has a pure Nash equilibrium is NP-complete given that the game is presented in GNF having a bounded neighborhood or in the case of acyclic-graph or acyclic-hypergraph games.
- Pure Nash Equilibrium existence and computations problems are feasible in logarithmic space for games in standard normal form.

A Second Better Problem

Input: A game in normal form.

Output: Is there a Mixed Nash Equilibrium?

Following the steps of quite a many papers and books written on this topic, we shall call this problem NASH.

Speculation

By Nash's Existence Theorem, this is intrinsically a search problem and not a decision problem.

This might tempt one to look for efficient algorithms for the above problem. There were many attempts in that direction.

> John von Neumann in the 1920s that MNE can be formulated in terms of linear programming for zero-sum games.

But what about games that are not zero-sum?

Several algorithms were proposed over the next half century, but all of them are either of unknown complexity, or known to require, in the worst case, exponential time.

Well then, one might ask, is NASH NP-complete?

Complexity of NASH

Equal-Subsets

Input: $A = \{a_1, \ldots, a_n\}$ where $\Sigma_{\forall i} a_i < 2^n - 1$ and $a_i \in \mathbb{Z}, \forall i$

Output: Two distinct sets $A_1,\ A_2$ such that $\Sigma_{\forall j \in A_1} j = \Sigma_{\forall k \in A_2} k$.

Before we further discuss NASH let's talk about the problem of EQUAL-SUBSETS.

It is evident that **EQUAL-SUBSETS** $\in$ **NP**, just like NASH. But is it NP-hard too or in other words - **is it reducible from SAT**? Is it NP-complete?

Total Search Problems

EQUAL-SUBSETS and NASH form a very specific group of problems which appear to be different from typical NP-complete problems like SAT.

Why? **Because unlike SAT,** these problems have a **guaranteed solution.** These problems (EQUAL-SUBSETS, NASH etc.) thus are Total Search Problems - problems where every instance has a solution. But is this difference enough to conclude that EQUAL-SUBSETS and NASH are not NP-complete?

Proof: **Total Search Problems in NP are not NP-complete**

Suppose Ψ is a total search problem in NP which is also NP-Complete. Then, SAT $\preceq_p \Psi$ and $\exists$ an algorithm A for SAT that runs in polynomial time, provided that it has access to poly-time algorithm A' for Ψ.

Now, suppose A has a non-satisfiable formula ϕ which calls A' some number of times. the algorithm eventually returns the answer NO implying that ϕ is not satisfiable.

Now, further suppose that we replace A' with a similar non-deterministic algorithm which in case of EQUAL-SUBSETS would be guess-and-test. This gives us a non-deterministic poly-time algorithm for SAT. The entire algorithm can recognize this

non-satisfiable formula ϕ, as before and we have a **NP algorithm that recognizes unsatisfiable formulae.**

This implies $\mathbf{NP = co\text{-}NP}$.

And, this is specifically why NASH too is very unlikely to be NP-complete. Below, we have another excellent argument by Megiddo.

> Suppose we have a reduction from SAT to NASH, that is, an efficient algorithm that takes as input an instance of SAT and outputs an instance of NASH, so that any solution to the instance of NASH tells us whether or not the SAT instance has a solution. Then we could turn this into a nondeterministic algorithm for verifying that an instance of SAT has *no* solution: Just guess a solution of the NASH instance, and check that it indeed implies that the SAT instance has no solution.
> The existence of such a non-deterministic algorithm for SAT (one that can verify that an unsatisfiable formula is indeed unsatisfiable) is an eventuality that is considered by complexity theorists almost as unlikely as $\mathbf{P = NP}$. We conclude that NASH is very unlikely to be **NP**-complete.

This implies that NASH and cimilar problems are highly likely to be easier than NP-complete problems. What what is the complexity class we can associate them to? A question still remains as to is it easy or is it hard?

TFNP: The Complexity Class

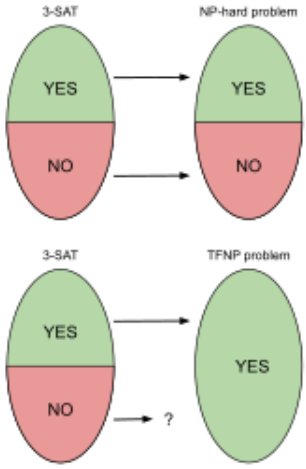
TFNP or Total Function Non-deterministic Polynomial problems are total function problems which can be solved in non-deterministic polynomial time. It is a subset of FNP - a function problem extension of the decision class NP.

This is the complexity class where we shall prove that NASH exists. It also contains the problems FACTORIZATION and EQUAL-SUBSETS.

> TFNP is widely conjectured to contain problems that are computationally intractable.

But, there hasn't been any results yet which show that TFNP problems are NP-hard. Moreover, TFNP-complete problems are not believed to exist.

> The top image shows the



typical form of a reduction that shows a problem is NP-hard. Yes instances map to yes instances and no instances map to no instances. The bottom image depicts the intuition for why it is difficult to show TFNP problems are NP-hard. TFNP problems always have a solution and so there is no simple place to map no instances of the original problem. This is the intuition behind the lack of NP-hardness results for TFNP problems.

Proving TFNP Membership

Proving that various search problems are total often use a "non-constructive step" which is hard to compute (unless the problem itself can be efficiently solved). However, it turns out that for most known total search problems these non-constructive steps are one of very few arguments.

These arguments define sub-classes within TFNP since they have mathematical theorems associated that prove their guarantee of existence of solution. TFNP, thus, is often studies through the lenses of these sub-classes and interestingly, these sub-classes have complete problems by virtue of the argument associated evn though TFNP itself does not have known complete problems.

Arguments and Sub-classes

- “If a function maps n elements to n − 1 elements, then there is a collision.” This is the *pigeonhole principle*, and the corresponding class is **PPP**.
- “If a directed graph has an unbalanced node (a vertex with different in-degree and out-degree), then it must have another.” This is the *parity argument* for directed graphs, giving rise to the class **PPAD** considered in this article.
- “If a graph has a node of odd degree, then it must have another.” This is the parity argument, giving rise to the class **PPA**.
- “Every directed acyclic graph must have a sink.” The corresponding class is called **PLS** for *polynomial local search*.

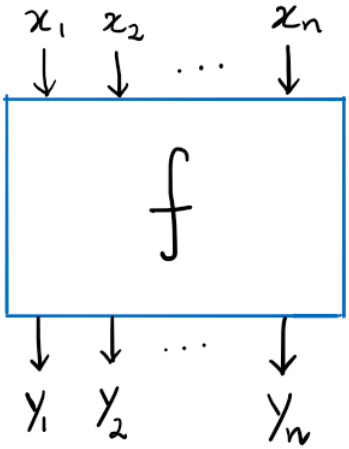
To understand how to define membership to these classes let's go back to our old friend the class NP.

One of the many ways to define NP is *class of all problems that can be reduced into instances of SAT*. Now, we shall define the above subclasses in similar fashion - **define classes** by their **complete problems.**

PPP and EQUAL-SUBSETS

We used the similarity between EQUAL-SUBSETS and NASH to introduce this new class of problems, namely TFNP. Now, it would be quite a shame if we don't actually prove the membership of EQUAL-SUBSETS in TFNP. Moreover, since it is a member of PPP, it will be quite interesting to see how a very simple yet powerful theorem (the pigeonhole principle) that we all have been introduced quite early in high school gives rise to a complexity class of it's own.

PPP stands for *polunomial pigenhole principle*. For the purpose of reductions we construct a gadget in the form of a PIGEONHOLE-CIRCUIT which in itself by definition is a PPP-complete problem.



Input: A boolean circuit C that takes n inputs and n outputs.
Output: A binary vector $\vec{x}$ such that $C(\vec{x}) = 0^n$ or alternatively, 2 vectors $\vec{x}$ and $\vec{x'}$ with $f(\vec{x}) = f(\vec{x'})$.

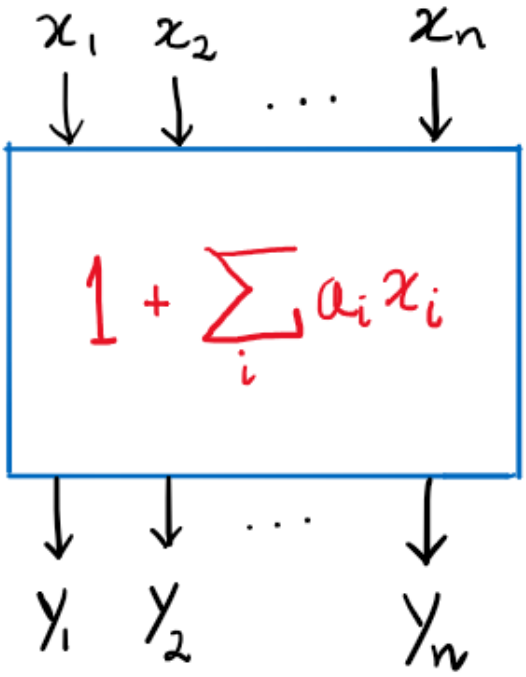
With regard to questions of polynomial time computation, the following are equivalent:

- n inputs/outputs $\rightarrow C$ of size n^2
- n inputs/outputs $\rightarrow C$ of size $O(n^k), k \in \mathbb{N}$
- $n =$ number of gates in C, number of inputs = number of outputs

Definition: A problem Ψ belongs to PPP if Ψ reduces to PIGEONHOLE CIRCUIT in polynomial time. Furthermore, if we can reduce Ψ from PIGEONHOLE CIRCUIT then Ψ is PPP-complete.

EQUAL-SUBSETS belongs to PPP

Taking inspiration from the book Proof without Words, we have:



Example:

Consider, $A = \{2, 3, 4, 5\}$ and $C = 1 + \Sigma_i(a_i x_i) = 1 + \begin{bmatrix} 2 & 3 & 4 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$

Then?

We have two vectors 2 vectors $\vec{x} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ and $\vec{x'} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$ such that, $C(\vec{x}) = C(\vec{x'})$.

Why?

Since, $1 + \begin{bmatrix} 2 & 3 & 4 & 5 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = 1 + \begin{bmatrix} 2 & 3 & 4 & 5 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = 5 = (0101)_b$.

Now, before we delve into the next class and classifying NASH let's refine the problem statement bore by NASH.

Refining NASH

Let's state the original problem that we posed.

Input: A game in normal form.

Output: Is there a Mixed Nash Equilibrium?

Now, in this problem for two-player games *the probabilities used by the players are rational numbers* (given that their utilities are also rational). So, it is clear how to write down the solution of a 2-player game.

However, as pointed out in Nash's original paper, when there are more than two players, there may be **only irrational solutions**.

As a result, the problem of computing a Nash equilibrium has to deal with issues concerning numerical accuracy. Therefore, we introduce the concept of Approximate Nash Equilibrium.

Approximate Nash Equilibrium

Input: A game in normal form where, $u_s^p \in [0, 1]$.

Output: Is there a Mixed ϵ-Nash Equilibrium?

$$\epsilon\textbf{-Nash Eqm: } E[\text{payoff}] + \epsilon \geq E[\text{payoff}] \text{ of best possible response}$$

This should be at least as tractable as finding an exact equilibrium, hence any hardness result for approximate equilibria carries over to exact equilibria.

NASH: "Re"-Definition

> Bounded rationality fixes irrationality.

Given the *description of a game* (by explicitly giving the utility of each player corresponding to each strategy profile), and a *rational number* $\epsilon > 0$, compute a *(mixed) Approximate Nash Equilibrium*.

END-OF-A-LINE Problem

Input: A directed graph G and a specified unbalanced vertex of G.

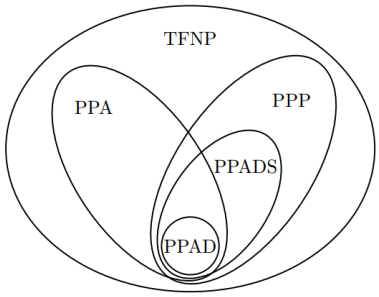
Output: Some other unbalanced vertex.

Now, by parity argument to know that a solution always exists.

> Suppose G has 2^n vertices, one for every bit string of length n (the parameter denoting the size of the problem). For simplicity, we will suppose that every vertex has at most one incoming edge and at most one outgoing edge. The edges of G will be represented by two boolean circuits, of size polynomial in n, each with n input bits and n output bits. Denote the circuits P and S (for predecessor and successor). Our convention is that there is a directed edge from vertex v to vertex v', if given input v, P outputs v' and, vice-versa, given input v', P outputs v. Suppose now that some specific, identified vertex (say, the string $\{0\}^*$) has an outgoing edge but no incoming edge, and is thus unbalanced. With the restriction of at most one incoming and one outgoing edge, the directed graph must be a set of paths and cycles; hence, following the path that starts at the all-zeroes node would eventually lead us to a solution. The catch is, of course, that this may take exponential time. Is there an efficient algorithm for finding another unbalanced node without actually following the path?

PPAD: The Class

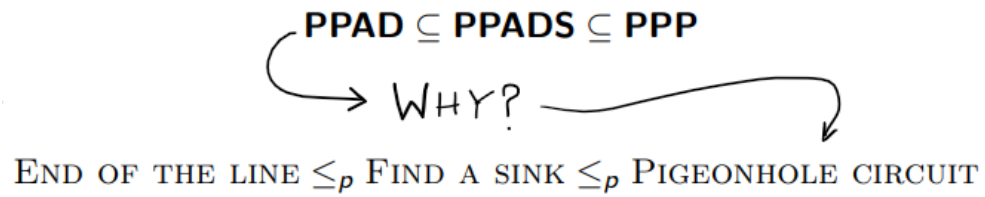
> PPA... what?
>
> **- Papadimitriou**



PPAD (which stands for *Polynomial Parity Arguments on Directed graphs*) is the class of all problems *reducible to END-OF-A-LINE*.

As expected, PPAD problems are believed to be hard, but obtaining PPAD-completeness is a weaker evidence of intractability than that of obtaining NP-completeness.

Arguably, the most celebrated application of the complexity of total search problems is the characterization of the computational complexity of finding a Nash Equilibrium in terms of PPAD-completeness. This problem lies in the heart of game theory and economics. The proof that Nash Equilibrium is PPAD-complete initiated a long line of research in the intersection of computer science and game theory and revealed connections between the two scientific communities that were unknown before.



Does PPAD contain hard problems?

In the absence of a proof P $\neq$ NP, we cannot confirm whether PPAD has hard problems. However, END-OF-A-LINE does appear to be a hard problem. Why?

- END-OF-A-LINE is a hard problem in the query complexity model, i.e., when S, P are given as black-box functions. We are interested in the computational variant where we have access to the circuits. But, except for very few exceptions, we don't know any way of looking inside a circuit and extracting useful information from it.
- Moreover under certain cryptographic assumptions (e.g., indistinguishablility obfuscation) the computational variant of END-OF-LINE is hard.

NASH is PPAD-complete

Here, we shall provide an outline to give an intuition behind why NASH is PPAD-complete.

Two parts of the proof are:

- Nash is in PPAD i.e., $\text{NASH} \preceq_p \text{END-OF-A-LINE}$
- $\text{END-OF-A-LINE} \preceq_p \text{NASH}$ or, NASH is PPAD-hard

Part 1

To prove that NASH is in PPAD, we can use a lot from Nash's original proof of existence, which utilizes Brouwer's fixed point theorem - it is essentially a reduction from NASH to BROUWER (problem of finding an approximate Brouwer fixed point of a continuous function).
Now, BROUWER can be reduced, under certain continuity conditions, to END-OF-A-LINE, and is therefore in PPAD.

$$\text{NASH} \preceq_p \text{BROUWER} \preceq_p \text{END-OF-A-LINE}$$

Let G be a normal form game with k players, $p = \{1, 2, \ldots, k\}$, and strategy sets $S_i = [n],\ \forall i \in [k]$ and let $\{u_s^i : i \in [k], s \in S\}$ be the utilities of the players.

Also let $\epsilon < 1$. In time polynomial in $|G| + \log(1/\epsilon)$, we will specify two circuits S and P each with $N = \text{poly}(|G|, \log(1/\epsilon))$ input and output bits and $P(0^N) = 0^N \neq S(0^N)$, so that, given any solution to END-OF-A-LINE on input S, P, one can

construct in poly-time an ϵ-approximate Nash Equilibrium of G. This is enough for reducing NASH to END-OF-A-LINE with some further observations.

The details of construction of S and P as well as the complete proof has properly been presented in the original 2008 Paper by Daskalakis et. al.

Part 2:

For this part we shall use the result that BROUWER is PPAD-complete and simulate the BROUWER with a game, which effectively reduces BROUWER to NASH.

The PPAD-complete class of Brouwer functions that appear above have the property that their function F can be efficiently computed using arithmetic circuits that are built up from standard operators such as addition, multiplication and comparison. The circuits can be written down as a *data flow graph*, with one of these operators at every node. The key is to simulate each standard operator with a game, and then compose these games according to the *data flow graph* so that the aggregate game simulates the Brouwer function, and an ϵ-Nash equilibrium of the game corresponds to an approximate fixed point of the Brouwer function.

We shall not be adding an elaborate reduction here, which can be found in the original paper as well.

2-NASH is PPAD-complete

Previously, we had discussed why we study Approximate Nash Equilibrium. We had mentioned how for a two-player game, approximation is not required necessarily. Here, we shall concern ourselves with that apparent special case.

Input: A 2-player game in normal form.
Output: Is there a Mixed Nash Equilibrium?

Original version of NASH with 2 players.

Let, this problem be called 2-NASH.

Remember that unlike our redefined NASH, 2-NASH is not mandated to be approximate.

Proof

- 2-NASH to BROUWER: As in before, this is guaranteed by John Nash's original work on the proof of existence of MNE.
- END-OF-A-LINE to BROUWER: This too is evident from the previous discussions.
- BROUWER to IMITATION-GAME: We shall try to reduce IMITATION-GAME from BROUWER. Here, consider the following utility functions.

$$u_1(x, y) = -||x - y||_2^2, \text{ and } u_2(x, y) = -||f(x) - y||_x^2$$

Only when, $y = x = f(y)$ this game has a Nash Equilibrium. However, now both players have infinite actions associated. So, we make the space discrete and both players have exponentially many actions. To fix these, we first reduce to a similar game with $2n$ players with the following utility functions.

$$u_{2i-1}(x_{2i-1}, y_{2i-1}) = -||x_{2i-1} - y_{2i-1}||^2, \text{ and } u_{2i}(x, y_{2i}) = -||f_{2i}(x) - y_{2i}||_x^2$$

- BROUWER to $2n$-NASH: We have $2n$ utility functions as described above and effectively reduce the actions per player to a constant number of actions.
- $2n$-NASH to 2-NASH using LAWYER'S-GAME: Let, Alice and Bob be the lawyers for the players $2i-1$ and $2i$ respectively. Since, lawyer's choose actions of players we have the following.

$$\text{Alice's actions: } S_1 \times S_3 \times \ldots \times S_{2n-1}$$
$$\text{Bob's actions: } S_2 \times S_4 \times \ldots \times S_{2n}$$

We can furthermore define the utilities of lawyers and make some necessary concessions or arrangements such as forcing the lawyer's to use all of their players. We use HIDE-AND-SEEK to achieve this. Effectively, Alice incentivizes to choose the same index as Bob whereas Bob tries to hide.

$$(u_A)_{MP}(i,j) = \begin{cases} M & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} = -(u_B)_{MP}(i,j)$$

Here, M is a sufficiently large number. Condition of MNE: when Alice and Bob play uniformly at random.

Conclusions

Game Theory encompases a **large genre** of games and auctions. It serves as a formalism for these too. And, we have shown how Nash Equilibrium a fundamental concept in Game Theory is likely to be hard as well as related the field to the study of complexity classes and effectively to computation itself. However, it is **not possible to model** all forms of games especially **simulations and puzzles** (which include 2D platform video games) into Game Theory.

> How about games such as chess?

We can capture this and other similar games (go, checkers) in the present framework by considering two players, Black and White, each with a huge action set containing all possible maps from positions to moves - but of course such formalism

(Game Theory) is **not very helpful for analyzing complexity of chess and similar games**.