Is NASH \textit{PPAD}-complete?

Rachael Colley

ILLC- University of Amsterdam

9$^{\text{th}}$ October 2018
Introduction

What does it mean for a problem to be complete?

- This has to be with respect to a complexity class.
- It has to be a member of that complexity class.
- It has to be a *hard* problem with respect to the other problems in this class. We do this by a reduction.
Introduction

Overview:

- NASH is in $PPAD$
  - Some potential complexity classes
  - Lemke-Howson Algorithm
- NASH is $PPAD$-hard
  - A reduction to the problem BROUWER
- Consequences
Why not \( NP \)?

\textbf{NASH:}
Given a finite two player game in strategic form, find a Nash Equilibrium.

\textbf{Decision Problems}

- Problems in \( NP \) are answered with ‘yes’ or ‘no’.
- In fact \textsc{Nash} requires a search algorithm to be able to find the \textit{exact} equilibria.
The complexity class \textit{FNP}

Functional \textit{NP} - \textit{FNP}

- \textit{FNP} stands for a complexity class ‘functional \textit{NP}’
- A problem in \textit{FNP} takes as input an instance of an \textit{NP} problem, and the responsibility of the algorithm is to output a solution or a ‘witness’.
- Take the \textit{FNP} version of SAT, known as FSAT.
- Similarly for \textit{NASH} we want that the output is the equilibria.
The complexity class \textit{FNP}

\textbf{Theorem}

\textit{If NASH is FNP-complete then }\textit{NP = coNP}

\textbf{Comments}

\begin{itemize}
  \item For \textit{NASH} to be \textit{FNP}-complete, this means that \textit{NASH} must also be \textit{FNP}—hard. This means that \textit{NASH} is among the hardest problems in the complexity class.
  \item \textit{NP = coNP} is thought to be very unlikely.
\end{itemize}
The complexity class \textit{FNP}

**Theorem:** If \textit{NASH} is \textit{FNP}-complete then \( NP = coNP \)

**Proof sketch:** There is a reduction from \textit{FSAT} to \textit{NASH}. Then the following algorithms exist:

1. A polynomial algorithm \( A \) that maps every \textit{FSAT} formula \( \varphi \) to a two player game \( A(\varphi) \).
2. A polynomial algorithm \( B \) that maps every MNE, \((\vec{x}, \vec{y})\) of a game \( A(\varphi) \) to either:
   - A satisfying assignment \( B(\vec{x}, \vec{y}) \) of \( \varphi \), if one exists.
   - the string ‘no’ otherwise.

We see here the asymmetry between these problems, this leads to \( NP = coNP \).
The complexity class *TFNP*

In the previous proof we saw a difference in the certificates in FSAT and NASH.

This comes down to the fact that in NASH we are guaranteed at least one witness, where this is not the case in FSAT.
The complexity class *TFNP*

Syntactic Vs. Semantic complexity classes

- **Syntactic complexity classes:**
  - These classes are characterised by some concrete computational model. The concrete reason characterises the whole class. For example:
  - $P$ contains all decision problems that can be solved by a deterministic Turing machine in polynomial time.
  - This is nice as we can have *hard* problems for syntactic classes, as we know the characteristics of all problems.

- **Semantic complexity classes:**
  - This set of classes do not have a generic reason for membership.
  - Such as *TFNP*, where there is a promise. We can fully characterise this semantic class.
The complexity class *TFNP*

**Definition: TFNP**

A problem $P$ is in TFNP if and only if:

- There is a deterministic polynomial time algorithm that can determine whether $P(x, y) = 1$ holds given both input $x$ and witness $y$. (Syntactic/FNP)

- For every $x$, there exists a $y$ such that $P(x, y)$ holds. (Semantic)
The complexity class \( TFNP \)

Another example: Factorising

- A witness of an input can be checked in polynomial-time and due to it outputting a solution then this must be in \( FNP \).
- For any \( x \in \mathbb{N}^+ \) it can always be written in its prime factorisation, thus we are guaranteed a solution.
The complexity class \textit{PPAD}

\textit{PPAD}, a syntactic subclass of \textit{TFNP}

- Definition of a parity-graph: A parity graph is a graph such that all paths between the same two vertices have the same parity.
- Definition of \textit{PPAD}: A problem in \textit{TFNP} which can be formulated on a parity-graph. This parity graph, organised into paths. Such that it has source and a sink, which will be the solution to the problem.

- Ideally, \textit{PPAD} = \textit{TFNP} and thus we would have a syntactic definition for \textit{TFNP}. However, it is unlikely for this to be the case.
The complexity class $PPAD$
Mixed Nash Equilibria and $PPAD$

A $PPAD$ problem for $NASH$: The Lemke-Howson Algorithm

- We saw $NASH$ was a $TFNP$ problem.
- Formulated on a graph consisting of paths.
- We start at a source (artificial equilibria)
- End at a sink (Nash equilibria)
NASH is a *PPAD*-hard problem

What we need to prove NASH is a hard problem.

- Gadgets in games
- r-NASH
- BROUWER
- BROUWER’
- The Levin Reduction
NASH is a \textit{PPAD}-hard problem

The problem $r$-NASH

\begin{itemize}
  \item This problem is very similar to NASH, however, we defined NASH to have two players.
  \item $r$-NASH is similar to NASH, however it is defined for $r$ players.
  \item This problem is in \textit{PPAD}, this is shown by a reduction to \textit{END-OF-THE-LINE}.
\end{itemize}

\textbf{END-OF-THE-LINE:}

Intuition: Given a start point/source, find the end point/sink, using a successor and predecessor function such that $P(V_2) = V_1$ and $S(V_1) = V_2$. 
NASH is a \textit{PPAD}-hard problem

The Problem \textsc{Brouwer}

- This is a discrete version of Brouwer’s fixed point theorem.
- \textsc{Brouwer}: Given a Boolean circuit describing the function $\varphi$, find the fixed point of $\varphi$, $x$.
- Input is $\varphi$, i.e. the function represented as a Boolean circuit.
- Output is $x$, the point at which $\varphi(x) \approx x$, or $\varphi(x) - x \leq \epsilon$. 

NASH is a $PPAD$-hard problem

Another way of formulating Brouwer’s Fixed Point Theorem.

**Theorem**

*Of a given function (circuit) $\varphi$, its fixed point will occur in the vicinity of a panchromatic vertex.*

**Definition**

A vertex is panchromatic if from its eight adjacent cubelets, the circuit $\varphi$ produces each of the four displacements $\varphi(x) = x + \delta_i$, where $i \in \{0, 1, 2, 3\}$. 
NASH is a PPAD-hard problem

Reformulating BROUWER

BROUWER’: Given a circuit $\varphi$, find a panchromatic vertex.

Setting of BROUWER’

- The domain of this discrete problem is the unit cube.
- It is split up into $2^{3n}$ equal cubelets. Each cubelet has sides of length $2^{-n}$.
- We describe each cubelet by the coordinates of its centre, this is the input to $\varphi$. 
NASH is a \textit{PPAD}-hard problem

\textbf{Solution to BROUWER’}

For each cubelet $x$, then $\varphi(x) = x + \delta_i$; where $i = 0, 1, 2, 3$, such that for $\alpha < 2^{-n}$:

- $\delta_1 = (\alpha, 0, 0)$
- $\delta_2 = (0, \alpha, 0)$
- $\delta_3 = (0, 0, \alpha)$
- $\delta_0 = (-\alpha, -\alpha, -\alpha)$

The solution will be the panchromatic vertex where all of the displacements are represented in the eight adjacent cubelets.
NASH is a \textit{PPAD}-hard problem

\textbf{Theorem}

\textit{BROUWER} and \textit{BROUWER’} are both \textit{PPAD–complete problems}.

The proof of this can be found in ‘The complexity of computing a Nash equilibrium.’ by Daskalakis et al., 2006.
NASH is a \textit{PPAD}-hard problem

The Levin Reduction

- \((y, g(z)) \in R_1 \text{ iff } (f(y), z) \in R_2\)
- If \(R_1\) and \(R_2\) are search problems and \(C\) is a complexity class then a \(C\) Levin reduction of \(R_1\) to \(R_2\) consists of three steps:
NASH is a \textit{PPAD}-hard problem

The Levin Reduction from \textsc{brouwer’} to \textsc{r-nash}

\[(f(\phi), z) \text{ is in } \textsc{r-nash} \iff (\phi, g(z)) \text{ is in } \textsc{brouwer’}.\]

\begin{tikzpicture}[node distance=2cm, auto]
  \node (input) {input: \(\phi\)};
  \node (output) [right of=input] {output: \(g(z)\)};
  \node (brouwer) [right of=input] {\textsc{brouwer’}};
  \node (r_nash) [right of=output] {\textsc{r-nash}};
  \node (x) [below of=r_nash] {\(x\)};
  \node (z) [below of=input] {\(z\)};
  \draw [->] (input) -- (brouwer) node[midway, above] {\(f(\phi) = x\)};
  \draw [->] (brouwer) -- (r_nash) node[midway, right] {\(g(z)\)};
  \draw [->] (r_nash) -- (z);
  \draw [->] (input) -- (output) node[midway, left] {\(\phi\)};
  \draw [->] (output) -- (z) node[midway, right] {\(z\)};
\end{tikzpicture}

N.B. The function \(g\) can depend on \(x\).
**NASH is a \textit{PPAD}-hard problem**

The Graphical Game
Step 1: From $\varphi$ to $f(\varphi)$.

- Actions: for each $i \in N$ we have that $A_i = \{0, 1\}$. Hence, their mixed strategy can be represented by a number in the interval $[0, 1]$. 

\[
\begin{array}{c}
\text{BROUWER} \\
\varphi \quad \rightarrow \quad f(\varphi) = x \\
g(z) \quad \downarrow \quad z
\end{array}
\]
NASH is a $PPAD$-hard problem

The Graphical Game

Players:

- Three player’s are leaders such that their strategies give coordinates in the unit cube, to be input into $\varphi$.
- Brittle comparators make this point in the unit cube a vertex in the corresponding Brouwer’ game.
- The remaining players represent the Boolean circuit $\varphi$ which can be built up from gadget games.
NASH is a PPAD-hard problem

The Graphical Game

Step 2: From $f(\varphi)$ to $z$, finding the Nash Equilibrium.

- We use the ‘leader’s’ mixed strategies give us a panchromatic vertex.
- From here we can find the inputs for the circuit of $\varphi$.
- Utility: The utilities of the game is set up in such a way that a Nash Equilibrium occurs only when the leaders have given a panchromatic vertex.
- Similarly to the Lemke-Howson algorithm, we use best responses until we reach a Nash Equilibrium, $z$. 
NASH is a \textit{PPAD}-hard problem

The Graphical Game

Step 3: The map from $z$ to $g(z)$, the fixed point of $\varphi$.

- The map of $z$ to the $g(z)$ is clearly a solution for \textsc{Brouwer’} due to the set up of the game.
- The Nash Equilibrium from $r$-NASH must be the solution to \textsc{Brouwer’}.
- At this point all of the displacements $\delta_i$ will be present, like the labelling in the Lemke-Howson algorithm.
NASH is a $PPAD$—complete problem

- We have seen that NASH and $r$–NASH are in the complexity class of $PPAD$.
- We saw a Levin reduction from a $PPAD$ hard problem BROUWER’ to $r$–NASH. Thus $r$ – NASH is a $PPAD$ hard problem.
- Hence, $r$–NASH is a $PPAD$–complete problem.
Consequences

How hard is a $PPAD$ problem?

- It is not understood how hard to compute $PPAD$ problems are to a deterministic Turing Machine.
- There is no unconditional proof whether $PPAD$ problems are polynomial-time solvable.
- There is usually the assumption made that $P \subsetneq PPAD$, made in a similar way to $P \neq NP$. 
Consequences

**NASH is not solvable in Polynomial time**

Players can’t always find MNE due to NASH being $PPAD$—complete.

- This suggests that the concept of Nash Equilibria is an unsuitable for general purposes for behaviour prediction.
- The polynomial-time tractability of computing an equilibria can be used as a necessary condition for its predictable plausibility.
Consequences

Other Problems with Nash Equilibria

- The non-uniqueness of Nash Equilibria limits the predictive power of in many settings.
- For example:

<table>
<thead>
<tr>
<th></th>
<th>Player 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>L</strong></td>
<td>(0,0)</td>
</tr>
<tr>
<td><strong>R</strong></td>
<td>(5,5)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Player 1</th>
<th><strong>T</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>L</strong></td>
<td>(0,0)</td>
</tr>
<tr>
<td><strong>B</strong></td>
<td>(5,5)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Player 1</th>
<th><strong>B</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>L</strong></td>
<td>(5,5)</td>
</tr>
<tr>
<td><strong>R</strong></td>
<td>(0,0)</td>
</tr>
</tbody>
</table>

- If uniqueness were guaranteed then the players would be able to optimise their payoff.
Consequences

Alternatives to Nash Equilibria
Instead of looking for Nash Equilibria of games maybe we should move to alternative notions which are tractable problems. Such as $\epsilon$–approximate MNE.

- Then an $\epsilon$–approximate MNE is where players cannot increase their payoff by more than $\epsilon$ via unilateral deviation.
- When $\epsilon \approx \frac{1}{3}$ then this is known to be in $P$.
- When $\epsilon$ is an arbitrarily small constant then this can be computed in quasi-polynomial time (worst case $2^{O((\log(n)^c))}$).
Conclusion

- The subclasses of $NP$, and the syntactic and semantic subclasses.
- We saw the reduction from Brouwer’ to $r$-NASH. This was done via the Levin reduction.
- As NASH is $PPAD$–complete we saw that this meant that we cannot compute it in polynomial time.
- We saw approximate equilibria and the possible reduction complexity for them to be computed.