Overview: Computing an Equil.

- **Tractable**
  - NE: 2-player, m-action, zero-sum
  - PSNE: n-player, m-action
  - Corr. Eq.: n-player, m-action
    - For Normal Form
    - For succinct representations

- **Intractable**
  - NE: 2 player, m-action, general-sum
Two-player, Zero-sum

• Input: normal-form representation

• Theorem. A Nash equilibrium can be computed in polynomial time.

• How? Compute Maximin (or minimax) strategies for each player.
Computing Maximin

\[ \text{LP}_1 : \max_{v_1, x} \quad v_1 \]
\[ \text{s.t. } \sum_{j \in A_1} u_1(j, k) \cdot x_j \geq v_1, \quad \forall k \in A_2 \]
\[ \sum_{j \in A_1} x_j = 1 \]
\[ x_j \geq 0, \quad \forall j \in A_1 \]

\[ \text{(3.5)} \]
\[ \text{(3.6)} \]
\[ \text{(3.7)} \]

\[ x_1 \ldots x_m \] represents the strategy of agent 1
Also write down \( \text{LP}_2 \) to compute maximin \( s_2 \) for agent 2

Computing Pure NE

- Thm. A pure strategy Nash equilibrium can be computed in polynomial time.

- How?
PSNE vs Mixed Strategy NE

- #subsets of (X, Y) vs size of game?

- Support Enumeration Method
  - Search (X, Y) supports
  - Look for NE “consistent” with (X, Y)
NashSupport: Idea

- Input (X,Y). Is there a NE “consistent” with (X,Y)?

  • Consistent
    - Agent 1 indifferent across every action in X, prefer to all not in X
    - Agent 2 indifferent across every action in Y, prefer to all not in Y
    - Agent 1 only uses actions in X
    - Agent 2 only uses actions in Y

A Linear Feasibility Problem

Agent 1 indifferent across all actions in X ... prefers to all outside X
\[ \sum_{k \in A_2} u_1(j, k) \cdot y_k = v_1, \ \forall j \in X \quad \sum_{k \in A_2} u_1(j, k) \cdot y_k \leq v_1, \ \forall j \in A_1 \setminus X \]

Agent 2 doesn’t use actions outside of Y
\[ \sum_{k \in A_2} y_k = 1, \ y_k \geq 0, \ \forall k \in Y, \ y_k = 0, \ \forall k \in A_2 \setminus Y \]

Agent 2 indifferent across all actions in Y
\[ \sum_{j \in A_1} u_2(j, k) \cdot x_j = v_2, \ \forall k \in Y, \ \sum_{j \in A_1} u_2(j, k) \cdot x_j \leq v_2, \ \forall k \in A_2 \setminus Y \]

Agent 1 doesn’t use actions outside of X
\[ \sum_{j \in A_1} x_j = 1, \ x_j \geq 0, \ \forall j \in X, \ x_j = 0, \ \forall j \in A_1 \setminus X \]

Caution: (i) need to enumerate all support pairs, (ii) becomes a nonlinear feasibility program for n>2.
A Direct Approach to FindNash

Recall our definition of a Nash equilibrium:
\[
\sum_{a_{-i} \in A_{-i}} p^*(a_{-i}) u_i(a_i, a_{-i}) \geq \sum_{a_{-i} \in A_{-i}} p^*(a_{-i}) u_i(a_i', a_{-i})
\]
for all \( a_{-i} \) in support, for all \( a_i' \)

Multiply both sides by \( s^*_{i}(a_i) \)
\[
\sum_{a_{-i} \in A_{-i}} u_i(a_i, a_{-i}) \cdot s^*_{i}(a_i) \cdot p^*(a_{-i}) \geq \sum_{a_{-i} \in A_{-i}} u_i(a_i', a_{-i}) \cdot s^*_{i}(a_i) \cdot p^*(a_{-i})
\]
for all \( a_{-i} \), all \( a_i' \)

- Idea: directly encode these inequalities as a nonlinear feasibility problem.

Example: Formulation for 2-players

For agent 1 to be best responding to \( y \) (the strategy of player 2):
\[
\sum_{k \in A_2} u_1(j, k) \cdot x_j \cdot y_k \geq \sum_{k \in A_2} u_1(j', k) \cdot x_j \cdot y_k, \quad \forall j \in A_1, \forall j' \in A_1
\]

For agent 2 to be best responding to \( x \) (the strategy of player 1):
\[
\sum_{j \in A_1} u_2(j, k) \cdot x_j \cdot y_k \geq \sum_{j \in A_1} u_2(j', k) \cdot x_j \cdot y_k, \quad \forall k \in A_2, \forall k' \in A_2
\]

Feasibility:
\[
\sum_{j \in A_1} x_j = 1, \quad \sum_{k \in A_2} y_k = 1, \quad x_j, y_k \geq 0, \forall j \in A_1, \forall k \in A_2
\]

Caution: nonlinear problem, cannot be solved in polynomial time
FindNash is \textbf{PPAD}-complete.

Conjectured that there is \textit{no} efficient algorithm to find a Nash eq. in general games.


Recall our definition of a Correlated equilibrium:

\[
\sum_{a_{-i} \in A_{-i}} p^* (a_{-i} \mid a_i) u_i (a_i, a_{-i}) \geq \sum_{a_{-i} \in A_{-i}} p^* (a_{-i} \mid a_i) u_i (a'_i, a_{-i})
\]

for all \( a_i \) in support, for all \( a'_i \)

Multiply both sides by \( p^* (a_i) \)

\[
\sum_{a_{-i} \in A_{-i}} u_i (a_i, a_{-i}) \cdot p^* (a_i) \cdot p^* (a_{-i} \mid a_i) \geq \sum_{a_{-i} \in A_{-i}} u_i (a'_i, a_{-i}) \cdot p^* (a_i) \cdot p^* (a_{-i} \mid a_i)
\]

for all \( a_i \), all \( a'_i \)

- Idea: directly encode these inequalities as a \textbf{linear} feasibility problem (!)
The Linear Feasibility Problem

Linear constraints to encode best response:

\[
\text{LFP}_{\text{cr}}: \sum_{a_i \in A_i} u_i(a_i, a_{-i}) \cdot p(a_i, a_{-i}) \geq \sum_{a_i' \in A_i} u_i(a_i', a_{-i}) \cdot p(a_i, a_{-i}), \forall i, \forall a_i, a_i' \in A_i
\]

Feasible distribution:

\[
\sum_{a \in A} p(a) = 1, \quad p(a) \geq 0, \quad \forall a \in A
\]

Linear program: can solve in polynomial time

Example:

<table>
<thead>
<tr>
<th>Player 1</th>
<th>Player 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>0, 0</td>
</tr>
<tr>
<td>G</td>
<td>2, 0</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\text{W} & \quad 0p_{11} + 0p_{12} \geq 2p_{11} - 4p_{12} \\
& \quad 2p_{21} - 4p_{22} \geq 0p_{21} + 0p_{22} \\
& \quad 0p_{11} + 0p_{21} \geq 2p_{11} - 4p_{21} \\
& \quad 2p_{12} - 4p_{22} \geq 0p_{12} + 0p_{22} \\
\text{G} & \quad p_{11} + p_{12} + p_{21} + p_{22} = 1, \quad p_{11}, p_{12}, p_{21}, p_{22} \geq 0
\end{align*}
\]

Finding a Correlated Eq.

- Input: normal-form representation of an n-player, m-action game
- **Theorem.** A Correlated eq. can be computed in polynomial time.

- Input: a succinct game, such as a congestion game or “graphical game”
- **Theorem.** A Correlated eq. can be computed in polynomial time.
Complexity of FindNash

- Input: a 2-player, m-action normal form game
- Output: a Nash equilibrium
Complexity classes

- **NP**: class of decision problems for which “yes” instances are easy to verify
  - E.g., 3-colorable
- **NP-hard**: the hardest problems in **NP**
  - Say $X$ is reducible to $Y$ if instance $I_x$ of $X$ can be converted to an instance $I_y$ of $Y$, s.t. a solution to $I_y$ solves $I_x$
    » all in polynomial time
    » $Y$ is “at least as hard” as $X$
- **P**: decision problems solvable in poly time
- Conjectured: $P \neq NP$

What about FindNash?

- **FindNash** is not believed to be NP-hard
  » Even though it is in **NP**!
- FindNash is **PPAD**-complete
  - It is in **PPAD**
  - It is **PPAD**-hard (any **PPAD** problem can reduce to FindNash)
- Conjectured that **PPAD** cannot be solved in polynomial time
- Inbetween that of **P** and **NP**, in that if can be solved, would NOT imply $P=NP$. 

25 26
What is the PPAD class?

- Polynomial parity argument on a directed graph (a class of “total search problems”)

- $X$ is in $\text{PPAD}$ if $X$ reduces to End-of-the-Line
- $X$ is $\text{PPAD}$-hard if End-of-the-Line reduces to $X$

- $EOTL$: Given a “PPAD graph” $G$ and the “standard source” $0$ find a source or sink.
- Special properties:
  - A solution always exists
  - May be an exponential number of vertices, and can only test for a vertex, and walk forward or backwards

Showing FindNash is $\text{PPAD}$ hard
The “PPAD graph” has the parity property

- No isolated vertices
- Each vertex has at most one in-edge, at most one out-edge

- Graph is collection of paths and cycles.
- Graph has a “standard source” => must exist another source or sink.
  - i.e., a solution…

**Input:** A set \( L = \{0,1\}^r \) of labels

- Can check:
  - Is Vertex(\( l \))?
  - Successor(\( l \)), Pred(\( l \))
    - encoded as boolean circuits

- The “standard source” 0
End-of-the-Line

- **Input:** A set \( L = \{0,1\}^r \) of labels
- Can check:
  - Is \( \text{Vertex}(l) \)?
  - Successor\((l)\), Pred\((l)\)
    - encoded as boolean circuits
- The “standard source” 0

- **Output:** return a sink or source \( \neq 0 \)
  \( \rightarrow \) want polynomial time in \( r \)

FindNash is in PPAD

- **Labels** \((l_1,l_2), |l_1|=m_1, |l_2|=m_2\)
- **Vertex** \((l_1,l_2)\) if:
  - Exists \( s_1 \) on actions not in \( l_1 \), causing BR2 on actions in \( l_1 \)
    \( \rightarrow \) or, \( l_1 = A_1 \)
  - Exists \( s_2 \) on actions not in \( l_2 \), causing BR1 on actions in \( l_2 \)
    \( \rightarrow \) or, \( l_2 = A_2 \)
  - Complete or almost-complete
    \( \rightarrow \) \( l_1 + l_2 \) include all actions except \( a_b \)
- **Edge** \((l_1,l_2) \rightarrow (l_1',l_2')\) if:
  - \( l_1 \) is “one different” from \( l_1' \), or \( l_2 \) “one different” from \( l_2' \)

Example \((a_b=M)\)

(direction omitted)
Special properties of Graph

- Standard source 0 is (A1,A2)
- Graph has parity property
  - each vertex has 1 or 2 neighbors
- Sources and sinks “complete”
- Exists a source or sink (!= 0)
  - In fact an odd number
- Source or sink corresponds to a Nash equilibrium
  - Consider (1,1,2). Complete.
  - Thus: a1 in A1 either:
    - in 11 (and not 12): a1 is not in support s1 (and a1 is not best response to s2)
    - not in 11 (in 12): a1 in support s1 (and a1 is best response to s2)
    - s1 is best response to s2
  - Similar for action a2 in A2

Example (a_b=M)

FindNash is PPAD-hard

- Need to reduce EOTL to FindNash
  - Go through BROUWER
  - Given continuous F: [0,1]^3 -> [0,1]^3, find an x s.t. F(x) = x
- Stylized, computational BROUWER:
  - Fix point encoded via an arithmetic circuit
  - Computes F on a very fine mesh in unit cube
- Idea of reduction:
  - Simulate the arithmetic circuits of BROUWER as the Nash equilibrium of a game
Encode circuits via “Game Gadgets”

e.g., MULTIPLY gadget

Claim: $x_4 = x_1 \times x_2$ in every NE

Summary

- FindNash is conjectured to be intractable
  - It is a “total search problem”
  - In particular, it is PPAD-complete
- This casts a shadow over the role of Nash equilibrium in complex settings
  - Can humans have incredible computational powers?
- In contrast, a correlated equilibrium can be computed in polynomial time
  » Even in succinct representations of games that don’t grow exponentially in the number of agents