1 Computation of Equilibria

1.1 Review

NASH Given a simultaneous-move game, find a mixed strategy Nash equilibrium.

Two-player zero-sum games

- Maximin Strategy: $\tilde{s}_1 \in \arg \max_{s_1} [\min_{k \in A_2} u_1(s_1, k)]$
- Minimax Strategy: $\tilde{s}_2 \in \arg \min_{s_2} [\max_{j \in A_1} u_1(j, s_2)]$
- Maximin and minimax value: $\bar{v}_1 = \min_{k \in A_2} u_1(\tilde{s}_1, k)$ and $\bar{v}_1 = \max_{j \in A_1} u_1(j, \tilde{s}_2)$
- Minimax Theorem: In any two-player, zero-sum game,
  - For each player the set of maximin strategies is identical to the set of minimax strategies.
  - Any maximin strategy for player 1 and any maximin strategy for player 2 is a NE, and these correspond to all NE's.
  - Maximin value = minimax value = expected utility.

General-sum games

- Finding PSNE's for multiple-player general-sum games
- Iterated Elimination of Strictly Dominated Actions
- Support Enumeration Method

1.2 Practice

1. Comprehension Questions

(a) How do you find a MSNE for a two-player zero-sum games? Why is this feasible and why this does not apply to two-player general-sum games?
(b) Briefly explain why the maximin strategy can be solved for both players separately?

(c) Briefly explain why the set of minimax strategies and the set of maximin strategies are identical?

(d) Is it easy to find a pure strategy DSE in a multiple player general sum game? What about finding a pure Nash Equilibrium? If you had to write some code to find the pure NE, how would you do it? Comment on the complexity.

(e) Consider a 2 player general-sum game where player 1 has 2 possible actions and player 2 has 3 possible actions. How many different pairs of support exist when looking for a Nash Equilibrium?

2. 2-player general-sum game. Take the following game:

<table>
<thead>
<tr>
<th></th>
<th>Player 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Left</td>
</tr>
<tr>
<td>Up</td>
<td>(6,4)</td>
</tr>
<tr>
<td>Player 1</td>
<td>Down</td>
</tr>
</tbody>
</table>

(a) Find a pure strategy Nash equilibrium.

(b) Given that there exists a mixed strategy Nash equilibrium with support \{UD,MR\}, formulate the LFP which can be used to solve for this mixed Nash equilibrium. You do not need to solve the LFP.
(c) In the LFP formulation, why we do not insist on positive probability for strategies within the support, and why is this ok?

2 Correlated equilibrium

1. Computation complexity of CE

(a) Briefly, what is the idea for being able to quickly find a CE of a multi-agent, general-sum game?

(b) Is the run-time for computing a CE polynomial in the size of the input? Is it polynomial in the number of agents $n$, or the number of actions $m$?

(c) What is the new problem when using an optimization approach to compute a MSNE?

2. 3 Player Pollution Game. Assume there are 3 players and a river that is polluted. Cleaning the river costs 1 for each player and the benefit from a clear river is 3 for each player. However, at least two players are needed to clean the river, otherwise the effort is wasted. Denote “cleaning” as C and “do not clean” as D.

(a) For a MSNE, suppose agents 1, 2 and 3 are cleaning the river with probabilities $p_1$, $p_2$ and $p_3$ respectively ($s_i(C) = p_i$ and $s_i(D) = 1 - p_i$). Suppose that there is a fully mixed equilibrium, such that each player puts some weight on both actions. Write the inequalities that define the MSNE for player 1.

(b) For a correlated equilibrium with $p$ over the action profiles, write the inequalities that define the CE for player 1.
(c) Briefly explain why computing CE can be solved with LFP, whereas computing MSNE cannot.

3 The Complexity Class PPAD

3.1 Review

Reduction A reduction between two problems $Y$ and $X$ transforms an instance $I_Y$ of $Y$ into an instance $I_X$ of $X$. Suppose we can do this in polynomial time. Then if we can solve $X$ in polynomial-time, we can also solve $Y$ in polynomial time via the reduction. In this sense, “$X$ is at least as hard as $Y$”. One way to use this is that if $Y$ is known to be hard, the reduction shows that $X$ is also hard.

Complexity Classes

- $P$: yes-no decision problems that can be solved in worst case polynomial time, e.g. linear programming, 2SAT, max flow
- $NP$: verifiable in polynomial time
- $NP$-hard: “very hard” problems, not necessarily in $NP$, and at least as hard as the hardest in $NP$, e.g subset sum problem
- $NP$-complete: problems in $NP$ and $NP$-hard (any problem in $NP$ can be reduced to it in polynomial-time), e.g. traveling salesman (decision version)
- $PPAD$: total search problems that can be reduced to End-of-the-Line.
- $PPAD$-hard: problems to which End-of-the-Line can be reduced (and are thus “at least as hard as EOTL”)
- $PPAD$-complete: problems in $PPAD$ and $PPAD$-hard

Nash

1. Nash in general sum games is $PPAD$-Complete. This means that it is conjectured that some instances might take exponential time to solve because it is believed that $PPAD \neq P$.

3.2 Practice

1. Explain briefly what does it mean for a problem to be $P$? $NP$? $NP$-hard?

2. Explain what does it mean for Nash to be $PPAD$-Complete, in terms of how easy it is to find a solution and whether a solution exists.
3. The Battle of the Sexes

<table>
<thead>
<tr>
<th>Player1 \ Player2</th>
<th>L</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>3,2</td>
<td>0,0</td>
</tr>
<tr>
<td>D</td>
<td>0,0</td>
<td>2,3</td>
</tr>
</tbody>
</table>

(a) List all labels $\ell_1$ for player 1 and all labels $\ell_2$ for player 2. Determine which ones are “valid.”

(b) What are the set of vertices in the PPAD graph?
(c) What is the PPAD graph? Use this to confirm the set of all Nash equilibria.

4. Game gadgets

(a) For the multiplication gadget introduced in the Figure 3.11 in the reading and reproduced below, why is it important to design the payoff of agent 4 such that it gets 1 when it takes the opposite action of agent 3 and 0 otherwise? ($x_1, x_2, x_3$ and $x_4$ denote the probability with which each of agents 1, 2, 3 and 4 plays action "1")

\[ u_3(a_3) = \begin{cases} a_1 \cdot a_2, & \text{if } a_3 = 0 \\ a_4, & \text{if } a_3 = 1 \end{cases} \]  
\[ u_4(a_4) = \begin{cases} 1 - a_3, & \text{if } a_4 = 1 \\ a_3, & \text{if } a_4 = 0 \end{cases} \]
(b) Is it important for agent 4 to know the value of $x_1x_2$? Why?

(c) Design a gadget with four agents 1, 2, 3, and 4 to implement $x_4 = \max(x_1 - x_2, 0)$. Agents 1 and 2 are inputs and agent 4 gets payoff 1 when playing the opposite action as agent 3. You need to design the utility function for agent 3! Prove that it “computes” this value of $x_4$ for all $x_1, x_2$ values.