

Further Solution Concepts & Computational Issues

CMPUT 654: Modelling Human Strategic Behaviour

S&LB §3.4.5, 3.4.7, 4.1, 4.2.3, 4.6

Assignment #1

- **Assignment #1 is released today**
See the website under Assignments (or on the Schedule)
- Due **October 1** before lecture

Recap: Solution Concepts

- **Maxmin strategies** maximize an agent's **guaranteed payoff**
- **Minmax strategies** minimize the other agent's payoff as much as possible
- The **Minimax Theorem**:
 - Maxmin and minmax strategies are the **only** Nash equilibrium strategies in **zero-sum games**
 - Every Nash equilibrium in a zero-sum game has the **same payoff**
- **Dominated strategies** can be removed **iteratively** without strategically changing the game (too much)
- **Rationalizable** strategies are any that are a **best response** to some **rational belief**

Lecture Outline

1. Recap & Logistics
2. ϵ -Nash Equilibrium
3. Correlated Equilibrium
4. Linear Programming
5. Computing Nash Equilibrium
6. Computing Correlated Equilibrium

ϵ -Nash Equilibrium

- In a Nash equilibrium, agents best respond **perfectly**
- What if they are indifferent to **very small** gains in utility?
 - Could reflect modelling error (e.g., unmodelled cost of computational effort)

Definition:

For any $\epsilon > 0$, a strategy profile s is an **ϵ -Nash equilibrium** if, for all agents i and strategies $s'_i \neq s_i$,

$$u_i(s_i, s_{-i}) \geq u_i(s'_i, s_{-i}) - \epsilon.$$

Questions:

For a given $\epsilon > 0$,

1. Is an ϵ -Nash equilibrium guaranteed to **exist**?
2. Is **more than one** ϵ -Nash equilibrium **guaranteed** to exist?

ε -Nash Equilibrium Example

		L	R
U		1, 1	0, 0
D		$1+(\varepsilon/2), 1$	500, 500

Questions:

1. What are the **Nash equilibria** of this game?
2. What are the **ε -Nash equilibria** of this game?

- Every Nash equilibrium is surrounded by a **region** of ε -Nash equilibria
 - Every **numerical algorithm** for computing Nash equilibrium actually computes ε -Nash equilibrium
- However, the reverse is not true! Payoffs from an ε -Nash equilibrium can be **arbitrarily far** from Nash equilibrium payoffs.

Correlated Equilibrium Examples

	Ballet	Soccer
Ballet	2, 1	0, 0
Soccer	0, 0	1, 2

	Go	Wait
Go	-10, -10	1, 0
Wait	0, 1	-1, -1

- In the unique mixed strategy equilibrium of Battle of the Sexes, each player gets a utility of $2/3$
- If the players could first observe a coin flip, they could coordinate on **which** pure strategy equilibrium to play
 - Each would get utility of 1.5
 - **Fairer** than either pure strategy equilibrium, and **Pareto dominates** the mixed strategy equilibrium
- **Correlated equilibrium** is a solution concept in which agents get private, potentially-correlated **signals** before choosing their action
 - In both of these example, each agent sees the **same signal** perfectly, but that is not necessary in general

Correlated Equilibrium

Definition:

Given an n -agent game $G = (N, A, u)$, a **correlated equilibrium** is a tuple (ν, π, σ) , where

- $\nu = (\nu_1, \dots, \nu_n)$ is a tuple of random variables with domains (D_1, \dots, D_n) ,
- π is a joint distribution over ν ,
- $\sigma = (\sigma_1, \dots, \sigma_n)$ is a vector of mappings $\sigma_i : D_i \rightarrow A_i$, and
- for every agent i and mapping $\sigma' : D_i \rightarrow A_i$,

Question: Why do the σ_i 's map to **pure strategies** instead of mixed strategies?

$$\sum_{d \in D_1 \times \dots \times D_n} \pi(d) u_i(\sigma_1(d_1), \dots, \sigma_n(d_n)) \geq \sum_{d \in D_1 \times \dots \times D_n} \pi(d) u_i(\sigma_1(d_1), \dots, \sigma'_i(d_i), \dots, \sigma_n(d_n))$$

Correlated Equilibrium Properties

Theorem:

For every **Nash equilibrium**, there exists a corresponding correlated equilibrium in which each action profile appears with the same frequency. (**how?**)

Theorem:

Any **convex combination** of correlated equilibrium payoffs can be realized in some correlated equilibrium. (**how?**)

Correlated Equilibrium

Another Example

	Ballet	Soccer
Ballet	2, 1	0, 0
Soccer	0, 0	1, 2

	Go	Wait
Go	-10, -10	1, 0
Wait	0, 1	-1, -1

- In our example correlated equilibria, each agent best-responded to the other **at every signal**
 - This is not a requirement of a correlated equilibrium
- Consider this correlated equilibrium, with $D_1 = \{x, y, z\}$ and $D_2 = \{m, r\}$:

$$\begin{aligned} \pi[(x, m)] &= .25 & \sigma_r(x) &= X & \sigma_c(m) &= M \\ \pi[(y, m)] &= .25 & \sigma_r(y) &= Y & & \\ \pi[(z, r)] &= .5 & \sigma_r(z) &= Z & \sigma_r(r) &= R \end{aligned}$$

- Question:** Does the column player best-respond at each signal?
- Question:** What are the **marginal probabilities** for each player's actions?
- Question:** What would happen if the agents played **mixed strategies** with those marginal probabilities?

	L	M	R
X	0,8	3,6	-9,1
Y	0,2	3,9	-12,10
Z	1,0	0,-2	7,7

Linear Programming

Definition:

A **linear program** consists of

- A set of real-valued **variables** $\{x_1, \dots, x_n\}$
- A linear **objective function** defined by **weights** $\{w_1, \dots, w_n\}$
- A set of linear **constraints** of the form $\sum_{j=1}^n a_j x_j \leq b$

Sample:

$$\begin{aligned} & \text{maximize} && \sum_{j=1}^n w_j x_j \\ & \text{subject to} && \sum_{j=1}^n a_{ij} x_j \leq b_i && \forall 1 \leq i \leq m \\ & && x_j \geq 0 && \forall 1 \leq j \leq n \end{aligned}$$

Linear Program Properties

$$\begin{aligned} & \text{maximize} && \sum_{j=1}^n w_j x_j \\ & \text{subject to} && \sum_{j=1}^n a_{ij} x_j \leq b_i && \forall 1 \leq i \leq m \\ & && x_j \geq 0 && \forall 1 \leq j \leq n \end{aligned}$$

- Linear programs can be solved in **polynomial time** by generic algorithms (e.g., ellipsoid algorithm)
 - So writing a problem as a linear program constitutes a **proof** that it is solvable in polynomial time
- Negating weights w_j allows us to **minimize** or **maximize** the objective
- Negating constraint coefficients a_{ij} allows for **greater-than-or-equal** constraints
- Providing both greater-than-or-equal and less-than-or-equal constraints allows for **equality constraints**
- **Cannot** always express **strict inequalities** (although there are tricks)

Computing Nash Equilibrium

- The problem of computing a Nash equilibrium is known to be **computationally hard** (*PPAD*-complete)
 - Even for two-player games!
- But there are some **special cases** that we can compute efficiently

Computing Nash Equilibrium: Zero-Sum Games

$$\begin{aligned} & \text{minimize } U_1^* \\ & \text{subject to } \sum_{a_2 \in A_2} u_1(a_1, a_2) s_2(a_2) \leq U_1^* \quad \forall a_1 \in A_1 \\ & \sum_{a_2 \in A_2} s_2(a_2) = 1 \\ & s_2(a_2) \geq 0 \quad \forall a_2 \in A_2 \end{aligned}$$

- This linear program computes U_1^* , player 1's **minmax value**, and s_2 , player 2's **minmax strategy** against player 1
 - By the minimax theorem, this is player 2's **equilibrium strategy**
- Compute player 1's equilibrium strategy analogously

Computing Maxmin Strategies: Two-Player, General-Sum Games

- We can efficiently compute the maxmin strategies for agents in a two-player **zero-sum game**
- The maxmin strategy for an agent in a **general-sum game** is their best response to an imaginary agent that is **trying to hurt them**
- To compute player 1's maxmin strategy in a general-sum game:
 1. Construct a **zero-sum game** from player 1's payoffs,
 2. Find player 1's minmax strategy in the **constructed game** (using the program from the previous slide)

Computing Nash Equilibrium: Two-Player, General Sum Games

- Finding an equilibrium in general is hard
- But if we already know the **support** of the equilibrium, then we can compute it efficiently in a two-player game:

$$\sum_{a_{-i} \in \sigma_{-i}} s_{-i}(a_{-i}) u_i(a_i, a_{-i}) = v_i \quad \forall i \in \{1,2\}, a_i \in \sigma_i$$

$$\sum_{a_{-i} \in \sigma_{-i}} s_{-i}(a_{-i}) u_i(a_i, a_{-i}) \leq v_i \quad \forall i \in \{1,2\}, a_i \notin \sigma_i$$

$$s_i(a_i) \geq 0 \quad \forall i \in \{1,2\}, a_i \in \sigma_i$$

$$s_i(a_i) = 0 \quad \forall i \in \{1,2\}, a_i \notin \sigma_i$$

$$\sum_{a_i \in A_i} s_i(a_i) = 1 \quad \forall i \in \{1,2\}$$

Questions:

1. Why can't we just set $\sigma_i = A_i$ for every agent and solve **once**?
2. Why can't we just try **every possible support**?
3. Why wouldn't this work for ***n*-player** games?

Computing Nash Equilibrium: General-Sum n -Player Games

- In theory, computing an equilibrium in n -player games and two-player games have **equal computational complexity**
- In practice, **two-player** games tend to be faster to solve:
 - Lemke-Howson pivoting algorithm based on a **linear complementarity program**
- For n -player games, **homotopy-following** methods:
 - Construct a family of parameterized **perturbations** of the game, with $t = 0$ being a trivial game with a known equilibrium, and $t = 1$ being the original game
 - Move t along $[0,1]$, adjusting the equilibrium as you go, until you reach $t = 1$

Computing Correlated Equilibrium

- Correlated equilibria can be found efficiently even in general-sum, n -player games
- Every correlated equilibrium induces a probability distribution over **action profiles**
 - Corresponds to a correlated equilibrium where Nature randomly chooses an action profile, and the agent's **signals** are their **own actions** in that profile
- So finding a **distribution over action profiles** in which each agent would always prefer to play their **recommended action** is sufficient to find a correlated equilibrium (**why?**)

Computing Correlated Equilibrium in Polynomial Time

$$\sum_{a \in A | a_i \in a} p(a) u_i(a) \geq \sum_{a \in A | a_i \in a'} p(a) u_i(a'_i, a_{-i}) \quad \forall i \in N, a_i, a'_i \in A_i$$

$$p(a) \geq 0 \quad \forall a \in A$$

$$\sum_{a \in A} p(a) = 1$$

We could find the social-welfare-optimizing correlated equilibrium by adding an **objective function**:

$$\text{maximize } \sum_{a \in A} p(a) \sum_{i \in N} u_i(a)$$

Summary

- **ϵ -Nash equilibria:** stable when agents have no deviation that gains them more than ϵ
- **Correlated equilibria:** stable when agents have **signals** from a possibly-correlated randomizing device
- **Linear programs** are a flexible encoding that can always be solved in **polytime**
- Finding a Nash equilibrium is **computationally hard** in general
- **Special cases** are efficiently computable:
 - Nash equilibria in zero-sum games
 - Maxmin strategies (and values) in two-player games
 - Correlated equilibrium