CSCE 631 — Algorithmic Game Theory

Lecture 5: Computing Equilibria I

Alan Kuhnle

Summer 2026

Texas A&M University

Today's Plan

Computational Perspective on Nash Equilibria

Why Computation Matters

- Nash's theorem (1950): every finite game has a mixed-strategy NE.
- **Existence $\neq$ efficient computability**.
- Central question: *Can we find a Nash equilibrium in polynomial time?*
- Practical motivation: mechanism design, auctions, network routing all need algorithmic solutions.

Bimatrix Game Recap

Two players: row player chooses $i \in [m]$, column player chooses $j \in [n]$. Payoff matrices $A, B \in \mathbb{R}^{m \times n}$. Mixed strategies $x \in \Delta_m$, $y \in \Delta_n$.

Nash Equilibrium — Formal Definition (Recall)

Definition (Nash Equilibrium)

A pair (x^*, y^*) is a Nash equilibrium if:

$$x^{*\top} A\, y^* \;\geq\; x^\top A\, y^* \quad \forall\, x \in \Delta_m$$

$$x^{*\top} B\, y^* \;\geq\; x^{*\top} B\, y \quad \forall\, y \in \Delta_n$$

- Neither player can unilaterally improve their expected payoff.
- Equivalent: every pure strategy in the *support* of x^* is a best response to y^*, and vice versa.

The Support of a Mixed Strategy

Definition (Support)
The **support** of a mixed strategy $x \in \Delta_m$ is

$$\mathrm{supp}(x) = \{i \in [m] : x_i > 0\}.$$

Best Response Characterization
(x^*, y^*) is an NE if and only if:

1. For all $i \in \mathrm{supp}(x^*)$: $(Ay^*)_i = \max_k (Ay^*)_k$.
2. For all $j \in \mathrm{supp}(y^*)$: $(B^\top x^*)_j = \max_k (B^\top x^*)_k$.

This characterization is the key to the **support enumeration** algorithm.

Support Enumeration Algorithm

Support Enumeration: Idea

1. Enumerate all possible support pairs (S_1, S_2) where $S_1 \subseteq [m]$, $S_2 \subseteq [n]$.
2. For each pair, solve a linear system to find (x, y) consistent with the best-response conditions.
3. Check feasibility: $x \geq 0$, $y \geq 0$, strategies outside the support are not better responses.

Key fact: For a nondegenerate bimatrix game, if (x^*, y^*) is an NE, then $|\mathrm{supp}(x^*)| = |\mathrm{supp}(y^*)|$.

Support Enumeration: The Linear System

Fix supports $S_1 \subseteq [m]$, $S_2 \subseteq [n]$ with $|S_1| = |S_2| = k$.

Row player's conditions (columns in S_2):

$$\sum_{j \in S_2} A_{ij}\, y_j = v_1 \quad \forall\, i \in S_1 \qquad \text{and} \qquad \sum_{j \in S_2} y_j = 1$$

Column player's conditions (rows in S_1):

$$\sum_{i \in S_1} B_{ij}\, x_i = v_2 \quad \forall\, j \in S_2 \qquad \text{and} \qquad \sum_{i \in S_1} x_i = 1$$

This gives $2k + 2$ equations in $2k + 2$ unknowns $(x_{S_1}, y_{S_2}, v_1, v_2)$.

Support Enumeration: Verification

After solving the linear system, verify:

1. $x_i > 0$ for all $i \in S_1$ and $y_j > 0$ for all $j \in S_2$.
2. For all $i \notin S_1$: $(Ay)_i \leq v_1$ (not a better response).
3. For all $j \notin S_2$: $(B^\top x)_j \leq v_2$.

Complexity

Number of support pairs: $\sum_{k=1}^{\min(m,n)} \binom{m}{k}\binom{n}{k}$. For $m = n$: this is $\Theta(4^n/\sqrt{n})$ — **exponential**.

Practical use: works well for small games ($n \leq 15$ or so).

Support Enumeration: Example

Example. Consider the 2×2 game:

$$A = \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix}, \qquad B = \begin{pmatrix} 0 & 1 \\ 3 & 0 \end{pmatrix}$$

Try support $S_1 = \{1, 2\}$, $S_2 = \{1, 2\}$ $(k = 2)$:

$$\begin{aligned} 3y_1 + 0y_2 &= 0y_1 + 1y_2 && \Rightarrow\ 3y_1 = y_2 \\ y_1 + y_2 &= 1 && \Rightarrow\ y_1 = \tfrac{1}{4},\ y_2 = \tfrac{3}{4} \end{aligned}$$

$$\begin{aligned} 0x_1 + 3x_2 &= 1x_1 + 0x_2 && \Rightarrow\ x_1 = 3x_2 \\ x_1 + x_2 &= 1 && \Rightarrow\ x_1 = \tfrac{3}{4},\ x_2 = \tfrac{1}{4} \end{aligned}$$

All positive $\Rightarrow$ valid NE: $x^* = (\frac{3}{4}, \frac{1}{4})$, $y^* = (\frac{1}{4}, \frac{3}{4})$.

Lemke–Howson Algorithm

Lemke–Howson: Overview

- Lemke & Howson (1964): a *path-following* algorithm for bimatrix games.
- Always finds one NE (not necessarily all).
- Operates on the **best-response polytopes** of the two players.
- Idea: follow edges of a *labeled polytope* from a known vertex (the artificial equilibrium) until reaching a fully labeled vertex (a genuine NE).

Key Property

The path is a sequence of *complementary pivots*, analogous to the simplex method for LPs.

Lemke–Howson: Best-Response Polytopes

Define polytopes for an $m \times n$ game:

$$P = \{x \in \mathbb{R}^m_{\geq 0} : B^\top x \leq \mathbf{1}\} \qquad Q = \{y \in \mathbb{R}^n_{\geq 0} : Ay \leq \mathbf{1}\}$$

(after normalizing payoffs to be positive).

Labels: Assign labels $1, \ldots, m+n$ to facets:

- Label i to facet $x_i = 0$ of P and to facet $(Ay)_i = 1$ of Q.
- Label $m+j$ to facet $(B^\top x)_j = 1$ of P and to facet $y_j = 0$ of Q.

Fully labeled: A vertex pair $(x, y) \in P \times Q$ is *completely labeled* if the union of their labels is $\{1, \ldots, m+n\}$.

Nash equilibria $\leftrightarrow$ completely labeled vertex pairs.

Lemke–Howson: The Pivoting Path

1. Start at $(0, 0)$ — the "artificial equilibrium" with labels $\{1, \ldots, m\}$ from P and $\{m+1, \ldots, m+n\}$ from Q.
2. **Drop** a chosen label ℓ.
3. Follow the unique edge leaving the current vertex in the polytope that lost the label.
4. Arrive at a new vertex; it picks up a new label ℓ'.
5. If ℓ' is a **duplicate**, drop it from the *other* polytope and continue.
6. Terminate when label ℓ is picked up again $\Rightarrow$ completely labeled $\Rightarrow$ NE.

Termination: guaranteed by a parity argument on the path graph (odd number of endpoints).

Lemke–Howson: Complexity

- Each pivot is polynomial time (a single simplex-like step).
- But the **number of pivots** can be exponential!
- Savani & von Stengel (2006): constructed bimatrix games where Lemke–Howson takes $\Omega(2^{n/2})$ steps.

Significance

Lemke–Howson was the leading candidate for a poly-time NE algorithm for decades.

Its exponential worst case motivated the search for complexity-theoretic lower bounds.

Complexity of Nash Equilibria: PPAD

Why Is NE Different from Other Search Problems?

- Nash's theorem guarantees existence $\Rightarrow$ this is a **total search problem** (solution always exists).
- Cannot be NP-complete (unless NP = coNP): NP-hard problems might have no solution.
- Need a complexity class for *problems guaranteed to have solutions*.

TFNP (Total Function NP)
The class of search problems in NP where a solution is guaranteed to exist.

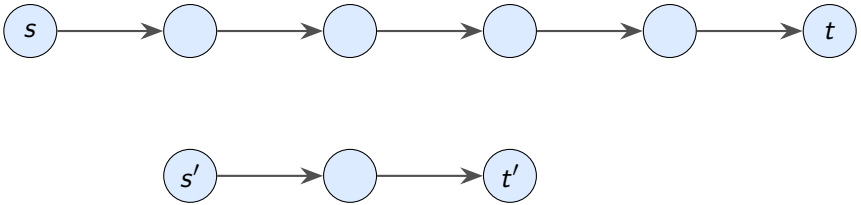
PPAD is a subclass of TFNP based on a specific *existence argument*.

Definition (PPAD — Polynomial Parity Argument, Directed)
Consider a directed graph on $\{0,1\}^n$ defined implicitly by poly-time computable predecessor/successor circuits P, S. Given a **source** node v_0 (known, with $\mathrm{indeg} = 0$), find *another* degree-1 node (a sink or another source).

Why does a solution exist?

- In any directed graph where each node has in-degree ≤ 1 and out-degree ≤ 1: the number of sources equals the number of sinks (mod 2 parity argument).
- Given one source, there must be *at least one other* degree-1 node.

PPAD: Intuition



- Graph is exponentially large (implicit representation).
- Given source s, must find t, s', or t'.
- Could follow the path from s — but path may be exponentially long.
- PPAD-hardness: no shortcut is known.

PPAD-Completeness of Nash Equilibria

Theorem (Daskalakis, Goldberg, Papadimitriou 2009; Chen, Deng 2009)

Finding a Nash equilibrium of a bimatrix game is **PPAD-complete**, even for 2-player games.

Significance:

- No polynomial-time algorithm exists unless PPAD = FP (widely believed to be false).
- This holds even for *two-player* games — the simplest non-trivial case.
- The 3-player case was shown PPAD-complete first (DGP 2006), then reduced to 2 players (CD 2006).

PPAD-Completeness: Proof Strategy

Two directions:

1. **Nash $\in$ PPAD:**
   - Encode Nash equilibrium computation as an "end of the line" problem.
   - The path-following structure of Lemke–Howson provides the directed graph.
2. **PPAD-hardness:**
   - Reduce from a canonical PPAD-complete problem (e.g., Brouwer fixed point) to finding NE.
   - Key step: embed a Brouwer function into game payoffs so that NE corresponds to a fixed point.
   - Technically intricate: requires "arithmetization" of the circuit.

Why Hard Despite Guaranteed Existence?

- **Structural complexity:** NE conditions are complementarity conditions (a strategy is either unused or a best response).
- **Combinatorial explosion:** the number of possible supports is exponential.
- **No convexity:** the set of NE is the union of (possibly exponentially many) convex sets.
- **Contrast with LP:** Linear programming is convex — one optimum, reachable by simplex/interior point in poly time.

Analogy

Finding NE is like finding the end of a very long string in a tangled ball: you know it exists, but you may have to follow the whole string.

Special Case: Zero-Sum Games

Zero-Sum Games Are Easy

Definition (Zero-Sum Game)
A two-player game where $B = -A$. One player's gain is the other's loss.

Key results:

- Von Neumann's Minimax Theorem (1928):

$$\max_{x \in \Delta_m} \min_{y \in \Delta_n} x^\top A y \;=\; \min_{y \in \Delta_n} \max_{x \in \Delta_m} x^\top A y \;=:\; v^*$$

- The common value v^* is the **value of the game**.
- The maxmin and minimax strategies form a Nash equilibrium.
- Solvable in **polynomial time** via linear programming!

LP Formulation: Row Player (Maxmin)

Row player maximizes the worst-case payoff:

$$\max_{x \in \Delta_m} \min_{y \in \Delta_n} x^\top A y \;=\; \max_{x \in \Delta_m} \min_{j \in [n]} (A^\top x)_j$$

Introduce variable v for the minimum:

LP for Row Player

$$\begin{aligned} \max \quad & v \\ \text{s.t.} \quad & \sum_{i=1}^{m} A_{ij}\, x_i \;\geq\; v \quad \forall\, j = 1, \ldots, n \\ & \sum_{i=1}^{m} x_i = 1 \\ & x_i \geq 0 \quad \forall\, i \end{aligned}$$

LP Formulation: Column Player (Minimax)

Column player minimizes the best payoff to the row player:

$$\min_{y \in \Delta_n} \max_{x \in \Delta_m} x^\top A y \;=\; \min_{y \in \Delta_n} \max_{i \in [m]} (Ay)_i$$

LP for Column Player

$$\begin{aligned} \min \quad & w \\ \text{s.t.} \quad & \sum_{j=1}^{n} A_{ij}\, y_j \;\le\; w \quad \forall\, i = 1, \ldots, m \\ & \sum_{j=1}^{n} y_j = 1 \\ & y_j \ge 0 \quad \forall\, j \end{aligned}$$

These two LPs are **dual** to each other $\Rightarrow v^* = w^*$ (strong duality).

LP Duality and the Minimax Theorem

- The row player's LP and the column player's LP are an LP dual pair.
- **Strong duality** of LP directly implies the minimax theorem:

$$\max_{x \in \Delta_m} \min_{y \in \Delta_n} x^\top A y = \min_{y \in \Delta_n} \max_{x \in \Delta_m} x^\top A y$$

- Conversely, the minimax theorem can be used to prove LP strong duality.
- Both are consequences of convex duality / separating hyperplane theorems.

Computational Consequence

We can find NE of zero-sum games in poly time using any LP solver (simplex, ellipsoid, interior point).

Example: Rock-Paper-Scissors

Payoff matrix for the row player:

$$A = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$$

By symmetry: $x^* = y^* = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ and $v^* = 0$.

LP verification: for each column j, $\sum_i A_{ij} x_i^* = \frac{1}{3}(0 + 1 - 1) = 0 \geq v^*$. ✓

This is the *unique* NE of Rock-Paper-Scissors.

Computational Complexity Landscape

Problem	Complexity	Method
2-player zero-sum NE	P	Linear programming
2-player general NE	PPAD-complete	Lemke–Howson / support enum
n-player NE	PPAD-complete	Fixed-point methods
Correlated equilibrium	P	Linear programming
ε-approximate NE	quasi-poly?	TS algorithm

Takeaway: The structure of the game matters enormously for computational tractability.

Summary

Summary

1. **Support enumeration:** correct but exponential-time; practical for small games.
2. **Lemke–Howson:** path-following on best-response polytopes; always terminates but can take exponentially many pivots.
3. **PPAD-completeness:** finding an NE is computationally hard (PPAD-complete for ≥ 2 players); unlikely to have a poly-time algorithm.
4. **Zero-sum games:** the major tractable special case; NE $=$ minimax strategies; solvable via LP in polynomial time.
5. **Next time:** LP formulations in detail, correlated equilibria, and approximate NE.