# L04 Computing NE in two player games

CS 280 Algorithmic Game Theory

Ioannis Panageas

# For known support

Question: Suppose we knew the support of the Nash for both players. Can we compute it?

# For known support

Question: Suppose we knew the support of the Nash for both players. Can we compute it?

Answer: Yes, via Linear Programming!

Let $R, C$ the payoff matrices of row, column players, of size $n \times m$.

# For known support

Question: Suppose we knew the support of the Nash for both players. Can we compute it?

Answer: Yes, via Linear Programming!

Let $R, C$ the payoff matrices of row, column players, of size $n \times m$.

Any Nash equilibrium with support $S, T$ $(x, y)$ must satisfy:

1a) $x_i \geq 0$ for all $i \in [n]$.

2a) $x_i = 0$ for all $i \notin S$.

3a) $\sum_{i \in S} x_i = 1$.

1b) $y_i \geq 0$ for all $i \in [m]$.

2b) $y_i = 0$ for all $i \notin T$.

3b) $\sum_{i \in T} y_i = 1$.

# For known support

Question: Suppose we knew the support of the Nash for both players. Can we compute it?

Answer: Yes, via Linear Programming!

Let $R, C$ the payoff matrices of row, column players, of size $n \times m$.

Any Nash equilibrium with support $S, T$ $(x, y)$ must satisfy:

1a) $x_i \geq 0$ for all $i \in [n]$.

2a) $x_i = 0$ for all $i \notin S$.

3a) $\sum_{i \in S} x_i = 1$.

4a) $(Ry)_i \geq (Ry)_j \; \forall i \in S, j \in [n]$.

1b) $y_i \geq 0$ for all $i \in [m]$.

2b) $y_i = 0$ for all $i \notin T$.

3b) $\sum_{i \in T} y_i = 1$.

4b) $(C^\top x)_i \geq (C^\top x)_j \; \forall i \in T, j \in [m]$.

# A trivial algorithm

LP $(S, T)$

$$(C^\top x)_i \geq (C^\top x)_j \ \forall i \in T, j \in [m].$$
$$(Ry)_i \geq (Ry)_j \ \forall i \in S, j \in [n].$$
$$\textstyle\sum_{i \in S} x_i = 1.$$
$$\textstyle\sum_{i \in T} y_i = 1.$$
$$x_i = 0 \text{ for all } i \notin S.$$
$$y_i = 0 \text{ for all } i \notin T.$$
$$x_i \geq 0 \text{ for all } i \in [n].$$
$$y_i \geq 0 \text{ for all } i \in [m].$$

Algorithm: For all index sets $S, T$, check feasibility of $LP\ (S, T)$. If a feasible solution $(x, y)$ is found, it is a Nash.

# A trivial algorithm

LP $(S, T)$

$$(C^\top x)_i \geq (C^\top x)_j \ \forall i \in T, j \in [m].$$

$$(Ry)_i \geq (Ry)_j \ \forall i \in S, j \in [n].$$

$$\sum_{i \in S} x_i = 1.$$

**Running time $2^{n+m} \cdot \text{poly}(n, m)$!**

**Slow, not polynomial!**

$$y_i \geq 0 \text{ for all } i \in [m].$$

Algorithm: For all index sets $S, T$, check feasibility of $LP\ (S, T)$. If a feasible solution $(x, y)$ is found, it is a Nash.

# Lemke-Howson Algorithm

Assumption: Matrices $R, C$ have non-negative entries. No loss of generality, NE are invariant under shifting.

Basic idea: The Lemke-Howson algorithm maintains a single guess of the supports, and in each iteration we change the guess only a little bit.

# Lemke-Howson Algorithm

Assumption: Matrices $R, C$ have non-negative entries. No loss of generality, NE are invariant under shifting.

Basic idea: The Lemke-Howson algorithm maintains a single guess of the supports, and in each iteration we change the guess only a little bit.

$$P_1 = \{x \in \mathbb{R}^n : \forall i \in [n] \;\; x_i \geq 0 \;\&\; \forall j \in [m] \;\; (x^\top C)_j \leq 1\}.$$

$$P_2 = \{y \in \mathbb{R}^m : \forall i \in [m] \;\; y_i \geq 0 \;\&\; \forall j \in [n] \;\; (Ry)_j \leq 1\}.$$

$$\mathrm{nrml}(x) = \left(\sum_{i\in[n]} x_i\right)^{-1} x \qquad \mathrm{nrml}(y) = \left(\sum_{i\in[m]} y_i\right)^{-1} y$$

# Lemke-Howson Algorithm

Assumption: Matrices $R, C$ have non-negative entries. No loss of generality, NE are invariant under shifting.

Basic idea: The Lemke-Howson algorithm maintains a single guess of the supports, and in each iteration we change the guess only a little bit.

$$P_1 = \{x \in \mathbb{R}^n : \forall i \in [n] \;\; x_i \geq 0 \;\&\; \forall j \in [m] \;\; (x^\top C)_j \leq 1\}.$$
$$P_2 = \{y \in \mathbb{R}^m : \forall i \in [m] \;\; y_i \geq 0 \;\&\; \forall j \in [n] \;\; (Ry)_j \leq 1\}.$$

$$\mathrm{nrml}(x) = \left(\textstyle\sum_{i\in[n]} x_i\right)^{-1} x \qquad \mathrm{nrml}(y) = \left(\textstyle\sum_{i\in[m]} y_i\right)^{-1} y$$

Def. $x$ has label $i$ if $x_i = 0$ or $(x^\top C)_i = 1$. Same for $j$.

# Lemke-Howson Algorithm

Assumption: Matrices $R, C$ have non-negative entries. No loss of generality, NE are invariant under shifting.

Basic idea: The Lemke-Howson algorithm maintains a single guess of the supports, and in each iteration we change the guess only a little bit.

$$P_1 = \{x \in \mathbb{R}^n : \forall i \in [n] \;\; x_i \geq 0 \;\&\; \forall j \in [m] \;\; (x^\top C)_j \leq 1\}.$$

$$P_2 = \{y \in \mathbb{R}^m : \forall i \in [m] \;\; y_i \geq 0 \;\&\; \forall j \in [n] \;\; (Ry)_j \leq 1\}.$$

$$\text{nrml}(x) = \left(\sum_{i \in [n]} x_i\right)^{-1} x \qquad \text{nrml}(y) = \left(\sum_{i \in [m]} y_i\right)^{-1} y$$

Def. $x$ has label $i$ if $x_i = 0$ or $(x^\top C)_i = 1$. Same for $j$.

**Lemma.** *Let $x^* \in P_1$, $y^* \in P_2$, $x^*, y^*$ have all labels and assume $x^*, y^*$ are not zero vectors. It holds that $(nrml(x^*), nrml(y^*))$ is a Nash equilibrium.*

# Lemke-Howson Algorithm

**Lemma.** *Let $x^* \in P_1$, $y^* \in P_2$, $x^*, y^*$ have all labels together and assume $x^*, y^*$ are not zero vectors. It holds that $(nrml(x^*), nrml(y^*))$ is a Nash equilibrium.*

*Proof.*

- For each $i \in [n]$, either $x_i^* = 0$ or $(Ry^*)_i = 1$ ($i$ is best response of row player to $\text{nrml}(y^*)$).
- For each $j \in [m]$, either $y_j^* = 0$ or $({x^*}^\top C)_j = 1$ ($j$ is best response of column player to $\text{nrml}(x^*)$).

# Lemke-Howson Algorithm

**Lemma.** *Let $x^* \in P_1$, $y^* \in P_2$, $x^*, y^*$ have all labels together and assume $x^*, y^*$ are not zero vectors. It holds that $(nrml(x^*), nrml(y^*))$ is a Nash equilibrium.*

*Proof.*

- For each $i \in [n]$, either $x_i^* = 0$ or $(Ry^*)_i = 1$ ($i$ is best response of row player to $\text{nrml}(y^*)$).
- For each $j \in [m]$, either $y_j^* = 0$ or $({x^*}^\top C)_j = 1$ ($j$ is best response of column player to $\text{nrml}(x^*)$).

We conclude that

$$\text{if } x_i^* > 0 \Rightarrow (Ry^*)_i \geq (Ry^*)_j \quad \forall j \in [n]$$

$$\text{if } y_i^* > 0 \Rightarrow ({x^*}^\top C)_i \geq ({x^*}^\top C)_j \quad \forall j \in [m]$$

# Lemke-Howson Algorithm

**Lemma.** *Let $x^* \in P_1$, $y^* \in P_2$, $x^*, y^*$ have all labels together and assume $x^*, y^*$ are not zero vectors. It holds that $(nrml(x^*), nrml(y^*))$ is a Nash equilibrium.*

*Proof.*

- For each $i \in [n]$, either $x_i^* = 0$ or $(Ry^*)_i = 1$ ($i$ is best response of row player to $\text{nrml}(y^*)$).
- For each $j \in [m]$, either $y_j^* = 0$ or $(x^{*\top} C)_j = 1$ ($j$ is best response of column player to $\text{nrml}(x^*)$).

We conclude that

$$\text{if } x_i^* > 0 \Rightarrow (Ry^*)_i \geq (Ry^*)_j \quad \forall j \in [n]$$

$$\text{if } y_i^* > 0 \Rightarrow (x^{*\top} C)_i \geq (x^{*\top} C)_j \quad \forall j \in [m]$$

Hence same is true for $\text{nrml}(x^*), \text{nrml}(y^*)$.

# Lemke-Howson Algorithm

**Lemma.** *Let $x^* \in P_1$, $y^* \in P_2$, $x^*, y^*$ have all labels together and assume $x^*, y^*$ are not zero vectors. It holds that $(nrml(x^*), nrml(y^*))$ is a Nash equilibrium.*

*Proof.*

- For eac[...]nse of row player t[...]
- For eac[...]esponse of column [...]

**They satisfy $\text{LP}(\text{Supp}(x^*), \text{Supp}(y^*))$!**

Inverse is also true!

We conclude that

$$\text{if } x_i^* > 0 \Rightarrow (Ry^*)_i \geq (Ry^*)_j \quad \forall j \in [n]$$

$$\text{if } y_i^* > 0 \Rightarrow (x^{*\top} C)_i \geq (x^{*\top} C)_j \quad \forall j \in [m]$$

Hence same is true for $\text{nrml}(x^*), \text{nrml}(y^*)$.

# Lemke-Howson Algorithm

**Definition** (Vertex). *A vertex of polytope $P_1$ is given by $n$ linearly independent equalities (the rest constraints of $P_1$ are strict inequalties). A vertex for $P_2$ is given by $m$ linearly independent equalities (the rest constraints of $P_1$ are strict inequalties). For $P_1 \cup P_2$ is $n + m$. This is the non-degenerate case.*

# Lemke-Howson Algorithm

**Definition** (Vertex). *A vertex of polytope $P_1$ is given by $n$ linearly independent equalities (the rest constraints of $P_1$ are strict inequalties). A vertex for $P_2$ is given by $m$ linearly independent equalities (the rest constraints of $P_1$ are strict inequalties). For $P_1 \cup P_2$ is $n+m$. This is the non-degenerate case.*

**Algorithm** (Lemke-Howson). *We define the following algorithm:*

1. Initialize $x = \mathbf{0}$ and $y = \mathbf{0}$.
2. $k = k_0 = 1$.
3. **Loop**
4. In $P_1$ find the neighbor vertex $x'$ of $x$ with label $k'$ instead of $k$. Remove label $k$ and add label $k'$.
5. **Set** $x = x'$.
6. **If** $k' = 1$ **STOP**.
7. In $P_2$ find the neighbor vertex $y'$ of $y$ with label $k''$ instead of $k'$. Remove label $k'$ and add label $k''$.
8. **Set** $y = y'$.
9. **If** $k'' = 1$ **STOP**.
10. **Set** $k = k''$.

# Analysis of Lemke-Howson

**Theorem.** *The Lemke-Howson algorithm outputs a Nash equilibrium.*

*Proof.* Define a graph with vertices in $P_1 \cup P_2$. Each vertex $(x, y)$ has:

- One **duplicate** label. This vertex is adjacent to exactly two other vertices, since we can remove the duplicate label from $x$ and pivot in $P_1$, or remove the duplicate label from $y$ and pivot in $P_2$.

# Analysis of Lemke-Howson

**Theorem.** *The Lemke-Howson algorithm outputs a Nash equilibrium.*

*Proof.* Define a graph with vertices in $P_1 \cup P_2$. Each vertex $(x, y)$ has:

- One **duplicate** label. This vertex is adjacent to exactly two other vertices, since we can remove the duplicate label from $x$ and pivot in $P_1$, or remove the duplicate label from $y$ and pivot in $P_2$.
- They have **all labels** exactly once. This vertex has only one neighbor (remove label 1 from whichever vector has it.

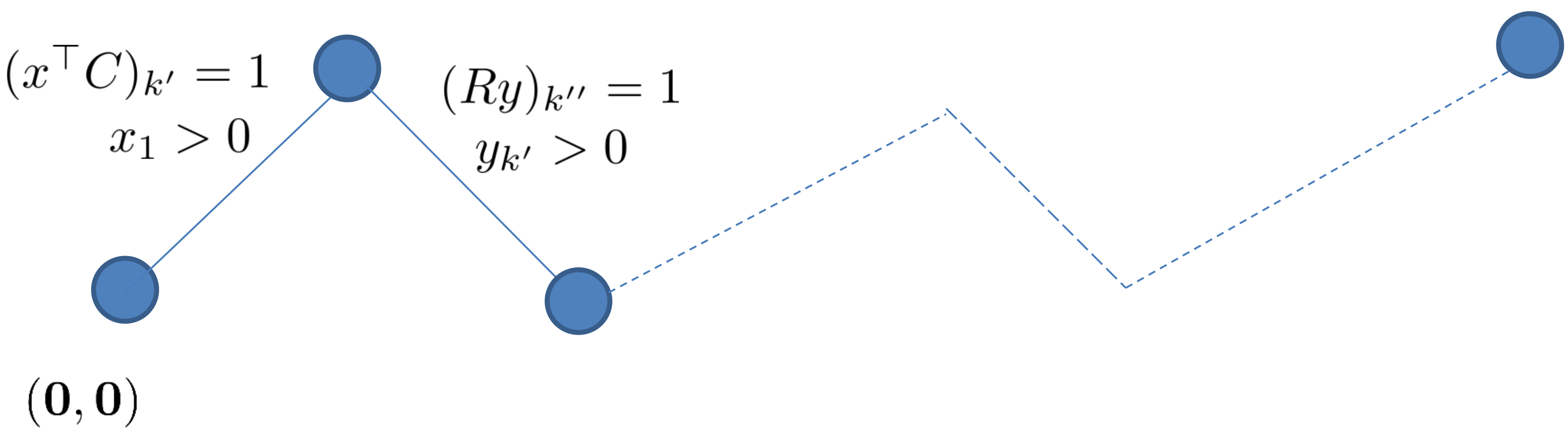
# Analysis of Lemke-Howson

**Theorem.** *The Lemke-Howson algorithm outputs a Nash equilibrium.*

*Proof.* Define a graph with vertices in $P_1 \cup P_2$. Each vertex $(x, y)$ has:

- One **duplicate** label. This vertex is adjacent to exactly two other vertices, since we can remove the duplicate label from $x$ and pivot in $P_1$, or remove the duplicate label from $y$ and pivot in $P_2$.

- They have **all labels** exactly once. This vertex has only one neighbor (remove label 1 from whichever vector has it.



# Analysis of Lemke-Howson

*Proof cont.* Since each vertex in the graph has degree 1 or 2, the graph is a union of cycles and paths!

# Analysis of Lemke-Howson

*Proof cont.* Since each vertex in the graph has degree 1 or 2, the graph is a union of cycles and paths!

1. Lemke-Howson algorithm begins at the configuration $(0, 0)$.
2. $(0, 0)$ has all labels and is therefore an endpoint of a path component.
3. The algorithm will terminate at the other endpoint of the path.
4. The other point is not $(0, 0)$ and cannot be $(x, 0)$ or $(0, y)$.

# Analysis of Lemke-Howson

*Proof cont.* Since each vertex in the graph has degree 1 or 2, the graph is a union of cycles and paths!

1. Lemke-Howson algorithm begins at the configuration $(0, 0)$.
2. $(0, 0)$ has all labels and is therefore an endpoint of a path component.
3. The algorithm will terminate at the other endpoint of the path.
4. The other point is not $(0, 0)$ and cannot be $(x, 0)$ or $(0, y)$.

**From previous lemma, it must be a Nash equilibrium!**

# Other facts

**Corollary (Odd Number).** *For non-degenerate games, the number of Nash equilibria is odd!*

# Other facts

**Corollary (Odd Number).** *For non-degenerate games, the number of Nash equilibria is odd!*

*Proof.* In a graph, the number of vertices with degree odd is even since

$$\sum_v d_v = 2E.$$

# Other facts

**Corollary (Odd Number).** *For non-degenerate games, the number of Nash equilibria is odd!*

*Proof.* In a graph, the number of vertices with degree odd is even since

$$\sum_v d_v = 2E.$$

Hence we have an even number of odd vertices. But $(\mathbf{0}, \mathbf{0})$ is an odd vertex and not a Nash equilibrium!

# Other facts

**Corollary** (Odd Number). *For non-degenerate games, the number of Nash equilibria is odd!*

*Proof.* In a graph, the number of vertices with degree odd is even since

$$\sum_v d_v = 2E.$$

Hence we have an even number of odd vertices. But $(\mathbf{0}, \mathbf{0})$ is an odd vertex and not a Nash equilibrium!

**Theorem** (Savani, von Stengel'04). *The Lemke-Howson algorithm runs in exponential time in worst-case*

# Approximating a Nash eq.

**Definition** ($k$-uniform)**.** *A strategy $x$ is called k-uniform when every coordinate $x_i$ is a multiple of $1/k$.*

Observation: A $k$-uniform strategy has support size at most $k$.

# Approximating a Nash eq.

**Definition** (*k*-uniform). *A strategy $x$ is called k-uniform when every coordinate $x_i$ is a multiple of $1/k$.*

Observation: A $k$-uniform strategy has support size at most $k$.

**Theorem** (Approximate Nash with small support). *Let $\epsilon > 0$. For any two player game, there always exists a k-uniform $\epsilon$-approximate Nash equilibrium for $k = \frac{12 \log n}{\epsilon^2}$.*

# Approximating a Nash eq.

**Definition** (*k*-uniform)**.** *A strategy $x$ is called k-uniform when every coordinate $x_i$ is a multiple of $1/k$.*

Observation: A $k$-uniform strategy has support size at most $k$.

**Theorem** (Approximate Nash with small support)**.** *Let $\epsilon > 0$. For any two player game, there always exists a k-uniform $\epsilon$-approximate Nash equilibrium for $k = \frac{12 \log n}{\epsilon^2}$.*

Remarks:

This was shown by Lipton, Markakis and Mehta using probabilistic method.

It gives a $n^{O\left(\frac{\log n}{\epsilon^2}\right)}$ algorithm. It was shown by Rubinstein that this is tight!