Algorithmic Game Theory and Applications

Lecture 9: Computing solutions for General Strategic Games: Part II: Nash Equilibria

Kousha Etessami

recall: Computing Nash Equilibria: a first clue

Recall "Useful corollary for NEs", from Lecture 3:

If x^* is an NE, then if $x_i^*(j) > 0$ then $U_i(x_{-i}^*; \pi_{i,j}) = U_i(x^*)$.

Using this, we can fully characterize NEs:

Proposition 1 In an n-player game, profile x^* is a NE if and only if there exist $w_1, \ldots, w_n \in \mathbb{R}$, such that:

1. $\forall$ players i, & $\forall$ $j \in \mathit{support}(x_i^*)$, $U_i(x_{-i}^*; \pi_{i,j}) = w_i$, &
2. $\forall$ players i, & $\forall$ $j \notin \mathit{support}(x_i^*)$, $U_i(x_{-i}^*; \pi_{i,j}) \leq w_i$.

<u>Note:</u> Any such w_i's necessarily satisfy $w_i = U_i(x^*)$.

Proof Follows easily from what we already know, particularly 1st claim in the proof of Nash's theorem. $\square$

using our first clue

- Suppose we somehow know support sets, $\mathit{support}_1 \subseteq S_1, \ldots, \mathit{support}_n \subseteq S_n$, for some Nash Equilibrium $x^* = (x_1^*, \ldots, x_n^*)$.
- Then, using Proposition 1, to find a NE we only need to solve the following system of constraints:
  1. $\forall$ players i, & $\forall\, j \in \mathit{support}_i$, $U_i(x_{-i}; \pi_{i,j}) = w_i$,
  2. $\forall$ players i, & $\forall\, j \notin \mathit{support}_i$, $U_i(x_{-i}; \pi_{i,j}) \leq w_i$.
  3. $\forall$ players $i = 1, \ldots, n$, $\sum_{j=1}^{m_i} x_i(j) = 1$.
  4. $\forall$ players $i = 1, \ldots, n$, & for $j \in \mathit{support}_i$, $x_i(j) \geq 0$.
  5. $\forall$ players $i = 1, \ldots, n$, & for $j \notin \mathit{support}_i$, $x_i(j) = 0$.
- This system has $\sum_{i=1}^{n} m_i + n$ variables, $x_1(1), \ldots, x_1(m_1), \ldots, x_n(1), \ldots, x_n(m_n), w_1, \ldots, w_n$.
- Unfortunately, for $n > 2$ players, this is a non-linear system of constraints. Let's come back to the case $n > 2$ players later.

two-player case

- In the 2-player case, the system of constraints is an LP!!
  But:

  Question: How do we find $\mathit{support}_1$ & $\mathit{support}_2$?

two-player case

- In the 2-player case, the system of constraints is an LP!!
  But:

  Question: How do we find $\mathit{support}_1$ & $\mathit{support}_2$?

  Answer: Just guess!!

First algorithm to find NE's in 2-player games

Input: A 2-player strategic game Γ, given by rational values $u_1(s, s')$ & $u_2(s, s')$, for all $s \in S_1$ & $s' \in S_2$. (I.e., the input is $(2 \cdot m_1 \cdot m_2)$ rational numbers.)

Algorithm:

- For all possible $support_1 \subseteq S_1$ & $support_2 \subseteq S_2$:
  - Check if the corresponding LP has a feasible solution $x^*, w_1, \ldots, w_n$. (using, e.g., Simplex).
  - If so, STOP: the feasible solution x^* is a Nash Equilibrium (and $w_i = U_i(x^*)$).

Question: How many possible subsets $support_1$ and $support_2$ are there to try?

First algorithm to find NE's in 2-player games

Input: A 2-player strategic game Γ, given by rational values $u_1(s, s')$ & $u_2(s, s')$, for all $s \in S_1$ & $s' \in S_2$. (I.e., the input is $(2 \cdot m_1 \cdot m_2)$ rational numbers.)
Algorithm:

- For all possible $support_1 \subseteq S_1$ & $support_2 \subseteq S_2$:
    - Check if the corresponding LP has a feasible solution $x^*, w_1, \ldots, w_n$. (using, e.g., Simplex).
    - If so, STOP: the feasible solution x^* is a Nash Equilibrium (and $w_i = U_i(x^*)$).

Question: How many possible subsets $support_1$ and $support_2$ are there to try?
Answer: $2^{(m_1+m_2)}$
So, unfortunately, the algorithm requires worst-case exponential time.
But, at least we have our first algorithm.

remarks on algorithm 1

- The algorithm immediately yields:
  Proposition Every finite 2-player game has a rational NE. (Furthermore, the rational numbers are not "too big", i.e., are polynomial sized.)
- The algorithm can easily be adapted to find not just any NE, but a "good" one. For example:
  Finding a NE that maximizes "(util.) social welfare":
  - For each support sets, simply solve the LP constraints while maximizing the objective
    $$f(x,w) = w_1 + w_2 + \ldots + w_n$$
  - Keep track of best NE encountered, & output optimal NE after checking all support sets.

- The same algorithm works for any notion of “good” NE that can be expressed via a linear objective and (additional) linear contraints: (e.g.: maximize Jane’s payoff, minimize John’s, etc.)
- Note: This algorithm shows that finding a NE for 2-player games is in “**NP**”.

Towards another algorithm for 2-players

Let A be the $(m_1 \times m_2)$ payoff matrix for player 1,
B be the $(m_2 \times m_1)$ payoff matrix for player 2,
w_1 be the m_1-vector, all entries $= w_1$,
w_2 be the m_2-vector, all entries $= w_2$.

Note: We can safely assume $A > 0$ and $B > 0$: by adding a large enough constant, d, to every entry we "shift" each matrix > 0. Nothing essential about the game changes: payoffs just increase by d.

We can get another, related, characterization of NE's by using "slack variables" as follows:

Lemma $x^* = (x_1^*, x_2^*)$ is a NE if and only if:

1. There exists a m_1-vector $y \geq 0$, and $w_1 \in \mathbb{R}$, such that
$$Ax_2^* + y = \mathrm{w}_1$$
& for all $j = 1, \ldots, m_1$, $x_1^*(j) = 0$ or $(y)_j = 0$.
2. There exists a m_2-vector $z \geq 0$, and $w_2 \in \mathbb{R}$, such that
$$Bx_1^* + z = \mathrm{w}_2$$
& for all $j = 1, \ldots, m_2$, $x_2^*(j) = 0$ or $(z)_j = 0$.

Proof Again follows by the Useful Corollary to Nash: in a NE x^* whenever, e.g., $x_1^*(j) > 0$, $U(x_{-1}^*; \pi_{1,j}) = U(x^*)$. Let $(y)_j = U(x^*) - U(x_{-1}^*; \pi_{1,j})$. $\square$

rephrasing the problem

The Lemma gives us some "constraints" that characterize NE's:

1. $Ax_2 + y = \mathrm{w}_1$ and $Bx_1 + z = \mathrm{w}_2$
2. $x_1, x_2, y, z \geq 0$.
3. x_1 and x_2 must be probability distributions,
   i.e., $\sum_{j=1}^{m_1} x_1(j) = 1$ and $\sum_{j=1}^{m_2} x_2(j) = 1$.
4. Additionally, x_1 and y, as well as x_2 and z, need to be "**complementary**":
   for $j = 1, \dots, m_1$, either $x_1(j) = 0$ or $(y)_j = 0$,
   for $j = 1, \dots, m_2$, either $x_2(j) = 0$ or $(z)_j = 0$.

   Since everything is ≥ 0, we can write this as

$$y^T x_1 = 0 \quad \text{and} \quad z^T x_2 = 0$$

continuing the reformulation

Note that, because $A > 0$ and $B > 0$, we know that $w_1 > 0$ and $w_2 > 0$ in any solution.

continuing the reformulation

Note that, because $A > 0$ and $B > 0$, we know that $w_1 > 0$ and $w_2 > 0$ in any solution.

Using this, we can "eliminate" w_1 and w_2 from the constraints as follows: Let $x_2' = (1/w_1)x_2$, $y' = (1/w_1)y$, $x_1' = (1/w_2)x_1$, and $z' = (1/w_2)z$.

continuing the reformulation

Note that, because $A > 0$ and $B > 0$, we know that $w_1 > 0$ and $w_2 > 0$ in any solution.

Using this, we can "eliminate" w_1 and w_2 from the constraints as follows: Let $x_2' = (1/w_1)x_2$, $y' = (1/w_1)y$, $x_1' = (1/w_2)x_1$, and $z' = (1/w_2)z$.

Let 1 denote an all 1 vector (of appropriate dimension).

Suppose we find a solution to

$$Ax_2' + y' = 1 \quad \text{and} \quad Bx_1' + z' = 1$$

$x_1', x_2', y', z' \geq 0$, $(y')^T x_1' = 0$, and $(z')^T x_2' = 0$.

continuing the reformulation

Note that, because $A > 0$ and $B > 0$, we know that $w_1 > 0$ and $w_2 > 0$ in any solution.

Using this, we can "eliminate" w_1 and w_2 from the constraints as follows: Let $x_2' = (1/w_1)x_2$, $y' = (1/w_1)y$, $x_1' = (1/w_2)x_1$, and $z' = (1/w_2)z$.

Let 1 denote an all 1 vector (of appropriate dimension).

Suppose we find a solution to

$$Ax_2' + y' = 1 \quad \text{and} \quad Bx_1' + z' = 1$$

$x_1', x_2', y', z' \geq 0$, $(y')^T x_1' = 0$, and $(z')^T x_2' = 0$.

<u>If, in addition,</u> $x_1' \neq 0$ or $x_2' \neq 0$, then, by complementarity <u>both</u> $x_1' \neq 0$ and $x_2' \neq 0$.

continuing the reformulation

Note that, because $A > 0$ and $B > 0$, we know that $w_1 > 0$ and $w_2 > 0$ in any solution.

Using this, we can "eliminate" w_1 and w_2 from the constraints as follows: Let $x_2' = (1/w_1)x_2$, $y' = (1/w_1)y$, $x_1' = (1/w_2)x_1$, and $z' = (1/w_2)z$.

Let 1 denote an all 1 vector (of appropriate dimension).

Suppose we find a solution to

$$Ax_2' + y' = 1 \quad \text{and} \quad Bx_1' + z' = 1$$

$x_1', x_2', y', z' \geq 0$, $(y')^T x_1' = 0$, and $(z')^T x_2' = 0$.

<u>If, in addition,</u> $x_1' \neq 0$ or $x_2' \neq 0$, then, by complementarity <u>both</u> $x_1' \neq 0$ and $x_2' \neq 0$.

In this case we can "recover" a solution x_1, x_2, y, z, and w_1 and w_2 to the original constraints, by multiplying x_1' and x_2' by "normalizing" constants w_1 and w_2, so that each of $x_1 = w_1 x_1'$ and $x_2 = w_2 x_2'$ define probability distributions. These normalizing constants define w_1 and w_2 in our solution.

2-player NE's as Linear Complementarity Problem

Let

$$M = \begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix} \quad u = \begin{bmatrix} x_1' \\ x_2' \end{bmatrix} \quad v = \begin{bmatrix} y' \\ z' \end{bmatrix}$$

"**Our Goal:**" Find a solution u, v, to

$$Mu + v = 1$$

such that $u, v \geq 0$, and $u^T v = 0$.

This is an intance of a Linear Complementarity Problem, a classic problem in mathematical programming (see, e.g., the book [Cottle-Pang-Stone'92]).

2-player NE's as Linear Complementarity Problem

Let

$$M = \begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix} \quad u = \begin{bmatrix} x_1' \\ x_2' \end{bmatrix} \quad v = \begin{bmatrix} y' \\ z' \end{bmatrix}$$

"**Our Goal:**" Find a solution u, v, to

$$Mu + v = 1$$

such that $u, v \geq 0$, and $u^T v = 0$.

This is an intance of a Linear Complementarity Problem, a classic problem in mathematical programming (see, e.g., the book [Cottle-Pang-Stone'92]).

But, we already know one solution: $u = 0$, $v = 1$.

2-player NE's as Linear Complementarity Problem

Let
$$M = \begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix} \quad u = \begin{bmatrix} x_1' \\ x_2' \end{bmatrix} \quad v = \begin{bmatrix} y' \\ z' \end{bmatrix}$$

"**Our Goal:**" Find a solution u, v, to
$$Mu + v = 1$$
such that $u, v \geq 0$, and $u^T v = 0$.

This is an intance of a Linear Complementarity Problem, a classic problem in mathematical programming (see, e.g., the book [Cottle-Pang-Stone'92]).

But, we already know one solution: $u = 0$, $v = 1$.

Our Actual Goal: is to find a solution where $u \neq 0$.

2-player NE's as Linear Complementarity Problem

Let
$$M = \begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix} \quad u = \begin{bmatrix} x_1' \\ x_2' \end{bmatrix} \quad v = \begin{bmatrix} y' \\ z' \end{bmatrix}$$

"**Our Goal:**" Find a solution u, v, to
$$Mu + v = 1$$
such that $u, v \geq 0$, and $u^T v = 0$.

This is an intance of a Linear Complementarity Problem, a classic problem in mathematical programming (see, e.g., the book [Cottle-Pang-Stone'92]).

But, we already know one solution: $u = 0$, $v = 1$.

Our Actual Goal: is to find a solution where $u \neq 0$.

Wait! Doesn't "$Mu + v = 1$" look familiar??

2-player NE's as Linear Complementarity Problem

Let
$$M = \begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix} \quad u = \begin{bmatrix} x'_1 \\ x'_2 \end{bmatrix} \quad v = \begin{bmatrix} y' \\ z' \end{bmatrix}$$

"**Our Goal:**" Find a solution u, v, to
$$Mu + v = 1$$
such that $u, v \geq 0$, and $u^T v = 0$.

This is an intance of a Linear Complementarity Problem, a classic problem in mathematical programming (see, e.g., the book [Cottle-Pang-Stone'92]).

But, we already know one solution: $u = 0$, $v = 1$.

Our Actual Goal: is to find a solution where $u \neq 0$.

Wait! Doesn't "$Mu + v = 1$" look familiar??

Sure! It's just a "Feasible Dictionary" (from lect. 6 on Simplex), with "Basis" the variables in vector v.

2-player NE's as Linear Complementarity Problem

Let
$$M = \begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix} \quad u = \begin{bmatrix} x'_1 \\ x'_2 \end{bmatrix} \quad v = \begin{bmatrix} y' \\ z' \end{bmatrix}$$

"**Our Goal:**" Find a solution u, v, to
$$Mu + v = 1$$
such that $u, v \geq 0$, and $u^T v = 0$.

This is an intance of a Linear Complementarity Problem, a classic problem in mathematical programming (see, e.g., the book [Cottle-Pang-Stone'92]).

But, we already know one solution: $u = 0$, $v = 1$.

Our Actual Goal: is to find a solution where $u \neq 0$.

Wait! Doesn't "$Mu + v = 1$" look familiar??

Sure! It's just a "Feasible Dictionary" (from lect. 6 on Simplex), with "Basis" the variables in vector v.

Question: How do we move from this "*complementary basis*" to one where $u \neq 0$?

2-player NE's as Linear Complementarity Problem

Let
$$M = \begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix} \quad u = \begin{bmatrix} x'_1 \\ x'_2 \end{bmatrix} \quad v = \begin{bmatrix} y' \\ z' \end{bmatrix}$$

"**Our Goal:**" Find a solution u, v, to
$$Mu + v = 1$$
such that $u, v \geq 0$, and $u^T v = 0$.

This is an intance of a <u>Linear Complementarity Problem</u>, a classic problem in mathematical programming (see, e.g., the book [Cottle-Pang-Stone'92]).

<u>But, we already know one solution:</u> $u = 0$, $v = 1$.

Our Actual Goal: is to find a solution where $u \neq 0$.

<u>Wait!</u> Doesn't "$Mu + v = 1$" look familiar??

Sure! It's just a "Feasible Dictionary" (from lect. 6 on Simplex), with "Basis" the variables in vector v.

Question: How do we move from this "<u>*complementary basis*</u>" to one where $u \neq 0$?

Answer: Pivoting!! (in a very selective way)

sketch of the Lemke-Howson Algorithm

1) Start at the “extra” “complementary Basis” $\beta = \{(v)_1, \ldots, (v)_m\}$, where $m = m_1 + m_2$ (with BFS $u = 0, v = 1$). β is **complementary** if for $k \in \{1, \ldots, m\}$, either $(u)_k \notin \beta$ or $(v)_k \notin \beta$ (but not both, since $|\beta| = m$).

sketch of the Lemke-Howson Algorithm

1) Start at the "extra" "complementary Basis" $\beta = \{(v)_1, \ldots, (v)_m\}$, where $m = m_1 + m_2$ (with BFS $u = 0, v = 1$). β is **complementary** if for $k \in \{1, \ldots, m\}$, either $(u)_k \notin \beta$ or $(v)_k \notin \beta$ (but not both, since $|\beta| = m$).
2) For <u>some</u> i, move via pivoting to a "neighboring" "i-almost complementary" basis β'. β' is **i-almost complementary** if for $k \in \{1, \ldots, m\} \setminus \{i\}$, $(u)_k \notin \beta'$ or $(v)_k \notin \beta'$.

sketch of the Lemke-Howson Algorithm

1) Start at the “extra” “complementary Basis” $\beta = \{(v)_1, \ldots, (v)_m\}$, where $m = m_1 + m_2$ (with BFS $u = 0, v = 1$). β is **complementary** if for $k \in \{1, \ldots, m\}$, either $(u)_k \notin \beta$ or $(v)_k \notin \beta$ (but not both, since $|\beta| = m$).
2) For <u>some</u> i, move via pivoting to a “neighboring” “i-almost complementary” basis β'. β' is **i-almost complementary** if for $k \in \{1, \ldots, m\} \setminus \{i\}$, $(u)_k \notin \beta'$ or $(v)_k \notin \beta'$.
3) While (new basis isn’t actually complementary)

- There’s a unique j, such that both $(u)_j$ & $(v)_j$ are not in the new basis: one was just kicked out of the basis.
- If $(u)_j$ was just kicked out, move $(v)_j$ into the basis by pivoting. If $(v)_j$ was just kicked, move $(u)_j$ in. (Selective pivot rules ensure only one possible entering/leaving pair.)
- Newest basis is also i-almost complementary.

sketch of the Lemke-Howson Algorithm

1) Start at the “extra” “complementary Basis” $\beta = \{(v)_1, \dots, (v)_m\}$, where $m = m_1 + m_2$ (with BFS $u = 0, v = 1$). β is **complementary** if for $k \in \{1, \dots, m\}$, either $(u)_k \notin \beta$ or $(v)_k \notin \beta$ (but not both, since $|\beta| = m$).
2) For <u>some</u> i, move via pivoting to a “neighboring” “i-almost complementary” basis β'. β' is **i-almost complementary** if for $k \in \{1, \dots, m\} \setminus \{i\}$, $(u)_k \notin \beta'$ or $(v)_k \notin \beta'$.
3) While (new basis isn't actually complementary)

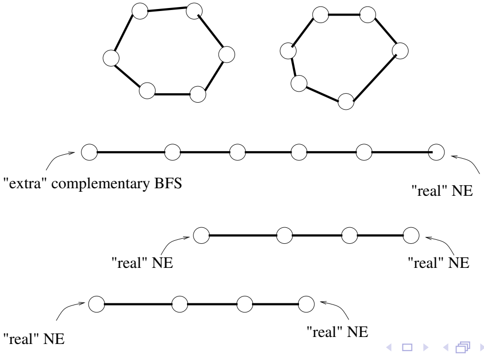
- There's a unique j, such that both $(u)_j$ & $(v)_j$ are not in the new basis: one was just kicked out of the basis.
- If $(u)_j$ was just kicked out, move $(v)_j$ into the basis by pivoting. If $(v)_j$ was just kicked, move $(u)_j$ in. (Selective pivot rules ensure only one possible entering/leaving pair.)
- Newest basis is also i-almost complementary.

4) STOP: we've reached a different complementary basis & BFS. A NE is obtained by “normalizing” $u = [x'_1 \; x'_2]^T$.

We are skipping lots of details related to "degeneracy", etc. (similar to complications that arose in Simplex pivoting).

Question Why should this work?

A key reason: With appropriately selective pivoting rules, each i-almost complementary Basis ("vertex") has 2 neighboring "vertices" unless it is actually a complementary Basis, in which case it has 1. This assures that starting at the "extra" complementary BFS, we will end up at "the other end of the line". Let's see it in pictures:



remarks

▷ The Lemke-Howson (1964) algorithm has a “geometric” interpretation. (See, [von Stengel, Chapter 3, in Nisan et. al. AGT book, 2007]. Our treatment is closer to [McKelvey-McLennan’96], see course web page.)

remarks

- ▷ The Lemke-Howson (1964) algorithm has a “geometric” interpretation. (See, [von Stengel, Chapter 3, in Nisan et. al. AGT book, 2007]. Our treatment is closer to [McKelvey-McLennan’96], see course web page.)
- ▷ The algorithm’s correctness gives another proof of Nash’s theorem $\underline{\textit{for 2-player games only}}$, just like Simplex’s gives another proof of Minimax (via LP-duality).

remarks

$\triangleright$ The Lemke-Howson (1964) algorithm has a “geometric” interpretation. (See, [von Stengel, Chapter 3, in Nisan et. al. AGT book, 2007]. Our treatment is closer to [McKelvey-McLennan’96], see course web page.)

$\triangleright$ The algorithm’s correctness gives another proof of Nash’s theorem *for 2-player games only*, just like Simplex’s gives another proof of Minimax (via LP-duality).

$\triangleright$ How fast is the LH-algorithm? Unfortunately, examples exist requiring exponentially many pivots,
for any permissible pivots (see [Savani-von Stengel’03]).

remarks

▷ The Lemke-Howson (1964) algorithm has a "geometric" interpretation. (See, [von Stengel, Chapter 3, in Nisan et. al. AGT book, 2007]. Our treatment is closer to [McKelvey-McLennan'96], see course web page.)

▷ The algorithm's correctness gives another proof of Nash's theorem *for 2-player games only*, just like Simplex's gives another proof of Minimax (via LP-duality).

▷ How fast is the LH-algorithm? Unfortunately, examples exist requiring exponentially many pivots,
for any permissible pivots (see [Savani-von Stengel'03]).

▷ Is there a polynomial time algorithm to find a NE in 2-player games? This is an *open problem*!

remarks

▷ The Lemke-Howson (1964) algorithm has a "geometric" interpretation. (See, [von Stengel, Chapter 3, in Nisan et. al. AGT book, 2007]. Our treatment is closer to [McKelvey-McLennan'96], see course web page.)

▷ The algorithm's correctness gives another proof of Nash's theorem *for 2-player games only*, just like Simplex's gives another proof of Minimax (via LP-duality).

▷ How fast is the LH-algorithm? Unfortunately, examples exist requiring exponentially many pivots,
for any permissible pivots (see [Savani-von Stengel'03]).

▷ Is there a polynomial time algorithm to find a NE in 2-player games? This is an *open problem*!

▷ However, finding "good" NE's that, e.g., maximize "social welfare" is NP-hard. Even knowing whether there is > 1 NE is NP-hard. ([Gilboa-Zemel'89], [Conitzer-Sandholm'03]).

games with > 2 players

$\triangleright$ Nash himself (1951, page 294) gives a 3 player "poker" game where the only NE is <u>irrational</u>.
So, it isn't so sensible to speak of computing an "exact" NE when the number of players is > 2.

games with > 2 players

▷ Nash himself (1951, page 294) gives a 3 player "poker" game where the only NE is irrational.
So, it isn't so sensible to speak of computing an "exact" NE when the number of players is > 2.

▷ We can try to **approximate** NEs. But there are different notions of approximate NE:
Definition 1: A mixed strategy profile x is called a ϵ-**Nash Equilibrium**, for some $\epsilon > 0$, if $\forall\ i$, and all mixed strategies y_i: $U_i(x) \geq U_i(x_{-i}; y_i) - \epsilon$.
I.e.: *No player can increase its own payoff by more than* ϵ *by unilaterally switching its strategy.*

games with > 2 players

▷ Nash himself (1951, page 294) gives a 3 player "poker" game where the only NE is irrational.
So, it isn't so sensible to speak of computing an "exact" NE when the number of players is > 2.

▷ We can try to **approximate** NEs. But there are different notions of approximate NE:
Definition 1: A mixed strategy profile x is called a ϵ-**Nash Equilibrium**, for some $\epsilon > 0$, if $\forall$ i, and all mixed strategies y_i: $U_i(x) \geq U_i(x_{-i}; y_i) - \epsilon$.
I.e.: *No player can increase its own payoff by more than ϵ by unilaterally switching its strategy.*
Definition 2: A mixed strategy profile x is ϵ-**close** to an actual **NE**, for some $\epsilon > 0$, if there is an actual NE x^*, such that $\|x - x^*\|_\infty \leq \epsilon$, i.e., $|x_i^*(j) - x_i(j)| < \epsilon$ for all i, j.

▷ It turns out these different notions of approximation of an NE have very different computation complexity implications.

What is the complexity of computing an ϵ-NE?

$\triangleright$ It turns out that:
(A) computing an NE for 2-player games, and
(B) computing an ϵ-NE for $>$ 2-player games
are reducible to each other.
Both are at least as hard as ANOTHER-LINE-ENDPOINT:
"Find another end-point of a succinctly given (directed) line graph, with indegree and outdegree ≤ 1*."*.

What is the complexity of computing an ϵ-NE?

$\triangleright$ It turns out that:
(A) computing an NE for 2-player games, and
(B) computing an ϵ-NE for $>$ 2-player games
are reducible to each other.
Both are at least as hard as ANOTHER-LINE-ENDPOINT:
"Find another end-point of a succinctly given (directed) line graph, with indegree and outdegree ≤ 1*."*.

$\triangleright$ [Papadimitriou 1992], defined a complexity class called **PPAD** to capture such problems, where ANOTHER-LINE-ENDPOINT is PPAD-complete.
He took inspiration from ideas in Lemke-Howson algorithm and an algorithm by [Scarf'67] for computing *almost* fixed points.

What is the complexity of computing an ϵ-NE?

$\triangleright$ It turns out that:
(A) computing an NE for 2-player games, and
(B) computing an ϵ-NE for $>$ 2-player games
are reducible to each other.
Both are at least as hard as ANOTHER-LINE-ENDPOINT:
"Find another end-point of a succinctly given (directed) line graph, with indegree and outdegree ≤ 1.".

$\triangleright$ [Papadimitriou 1992], defined a complexity class called **PPAD** to capture such problems, where ANOTHER-LINE-ENDPOINT is PPAD-complete.
He took inspiration from ideas in Lemke-Howson algorithm and an algorithm by [Scarf'67] for computing *almost* fixed points.

$\triangleright$ [Chen-Deng'06] & [Daskalakis-Goldberg-Papadimitriou,'06], showed that computing an NE in 2-player games, & computing a ϵ-NE in $>$ 2-player games, respectively, are PPAD-complete.

$\triangleright$ What about ϵ-close approximating an actual NE??

The complexity of computing an actual NE in games with > 2 players

$\triangleright$ For games with > 2 players, approximating an actual NE, i.e., computing a profile ϵ-close to an actual NE, even for any $\epsilon < 1/2$, is MUCH harder. Not even known to be in **NP**. The best complexity upper bound we know is **PSPACE** (using deep but impractical algorithms for solving nonlinear systems of equations [Gregoriev-Vorobjov'88,Canny'88,Renegar'92]).

$\triangleright$ [Etessami-Yannakakis'07] showed that if we can approximate an actual NE even in NP, that would resolve major open problems in the complexity of numerical analysis (seems unlikely at present). They showed computing or approximating an actual NE is **FIXP**-complete, where FIXP consists of all problems reducible to computing a fixed point for algebraic Brouwer functions defined by operators $\{+, *, -, /, \max, \min\}$ and rational constants.

$\triangleright$ Such fixed point computation problems have many other important applications, in particular, for computation of **market equilibria**.

$\triangleright$ In turns out that PPAD is exactly the *"piecewise linear"* fragment of FIXP, consisting of problems reducible to Brouwer fixed point problems defined by algebraic functions using operators $\{+, -, \max, \min\}$.

$\triangleright$ These results are beyond the scope of this course.
If you are interested to learn more, see:

K. Etessami and M. Yannakakis, "On the Complexity of Nash Equilibria and other Fixed Points", *SIAM Journal on Computing*, 39(2), pp. 2531-2597, 2010.

and the references therein.