Algorithms to Compute a Nash Equilibrium

Lemke-Howson Algorithm – Algebraic Approach

Lemke-Howson Algorithm

- 2-player, general sum games
- Algorithm is for solving linear complementarity programs
- Searches vertices of strategy simplices (like the simplex algorithm for solving LPs)
- Best response condition: Let *B* be the payoff matrix for Player 1. Let *x*, *y* be mixed strategies for player 1, 2. *x* is a best response iff

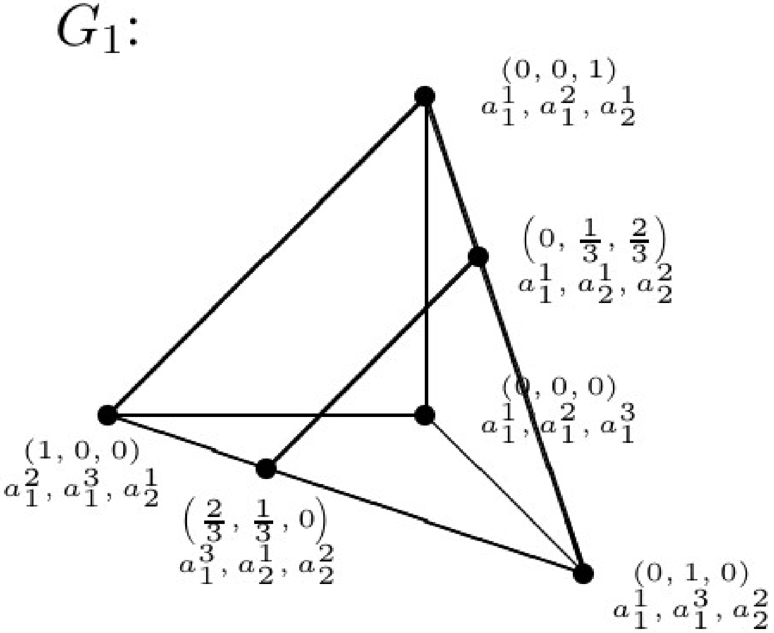
$$x_i > 0 \rightarrow (By)_i = u = max\{ (By)_k \mid k \text{ in } A_1\}$$

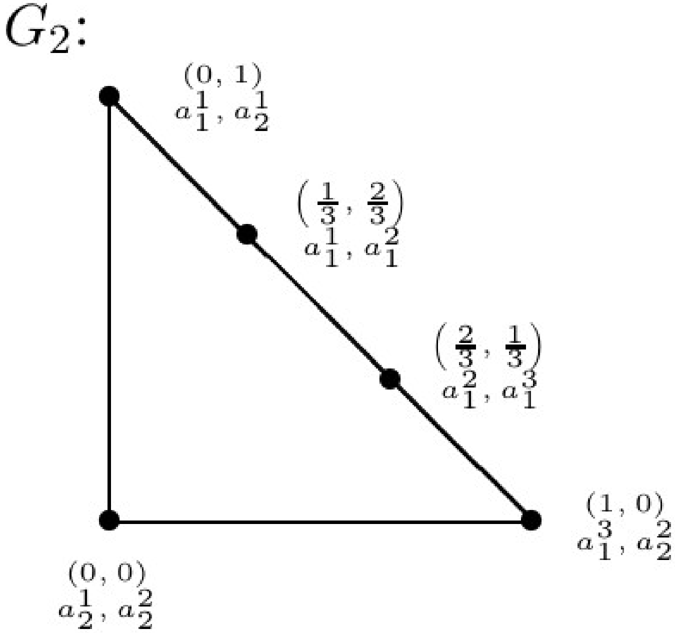
Lemke-Howson – a graphical exposition

$0, 1$	$6, 0$
$2, 0$	$5, 2$
$3, 4$	$3, 3$

Figure 4.1: A game for the exposition of the Lemke–Howson algorithm.

Lemke-Howson – a graphical exposition





Lemke-Howson – Properties

- Guaranteed to find a NE
- Alternative proof of the existence of NE
- Path after initial move is unique. Only nondeterminism is in first move
- All paths from the starting point to a NE can be exponential (algorithm is provably exponential)
- No way to assess how close we are to a NE

Lemke-Howson – Implementation

- How to compute vertices / labels of the strategy simplices?
- We will only compute the vertices along the path traveled in online fashion

Lemke-Howson – Pseudocode

initialize *the two systems of equations at the origin*
arbitrarily pick *one dependent variable from one of the two systems. This variable* enters *the basis.*
repeat
    identify *one of the previous basis variables which must* leave, *according to the minimum ratio test. The result is a new basis.*
    if *this basis is completely labeled* **then**
        return *the basis* // we have found an equilibrium.
    else
        the variable dual to the variable that last left enters *the basis.*

Figure 4.5: Pseudocode for the Lemke–Howson algorithm.

The LCP Formulation

$$\sum_{k \in A_2} u_1(a_1^j, a_2^k) \cdot s_2^k + r_1^j = U_1^* \qquad \forall j \in A_1 \tag{4.14}$$

$$\sum_{j \in A_1} u_2(a_1^j, a_2^k) \cdot s_1^j + r_2^k = U_2^* \qquad \forall k \in A_2 \tag{4.15}$$

$$\sum_{j \in A_1} s_1^j = 1, \quad \sum_{k \in A_2} s_2^k = 1 \tag{4.16}$$

$$s_1^j \geq 0, \quad s_2^k \geq 0 \qquad \forall j \in A_1, \forall k \in A_2 \tag{4.17}$$

$$r_1^j \geq 0, \quad r_2^k \geq 0 \qquad \forall j \in A_1, \forall k \in A_2 \tag{4.18}$$

$$r_1^j \cdot s_1^j = 0, \quad r_2^k \cdot s_2^k = 0 \qquad \forall j \in A_1, \forall k \in A_2 \tag{4.19}$$

Lemke-Howson – Example

$0,1$	$6,0$
$2,0$	$5,2$
$3,4$	$3,3$

Figure 4.1: A game for the exposition of the Lemke–Howson algorithm.

$$\begin{array}{rcllll} r_1 & = & 1 & & -6y'_5 \\ r_2 & = & 1 & -2y'_4 & -5y'_5 \\ r_3 & = & 1 & -3y'_4 & -3y'_5 \end{array} \tag{4.20}$$

$$\begin{array}{rcllll} s_4 & = & 1 & -x'_1 & & -4x'_3 \\ s_5 & = & 1 & & -2x'_2 & -3x'_3 \end{array} \tag{4.21}$$

Lemke-Howson – Example

- Recall: only one of r_1, x_1' can be nonzero
- All slacks nonzero → all probs. = 0.

$$\begin{array}{rclll} r_1 & = & 1 & & -6y_5' \\ r_2 & = & 1 & -2y_4' & -5y_5' \\ r_3 & = & 1 & -3y_4' & -3y_5' \end{array} \qquad (4.20)$$

$$\begin{array}{rcllll} s_4 & = & 1 & -x_1' & & -4x_3' \\ s_5 & = & 1 & & -2x_2' & -3x_3' \end{array} \qquad (4.21)$$

- For first move, arbitrarily pick x_2' to enter
- Since s_5 clashes with x_2', s_5 must leave. (4.21) becomes:

$$\begin{array}{rcllll} s_4 & = & 1 & -x_1' & -4x_3' & \\ x_2' & = & \frac{1}{2} & & -\frac{3}{2}x_3' & -\frac{1}{2}s_5 \end{array} \qquad (4.22)$$

Lemke-Howson – Example

$$
\begin{array}{rclll}
r_1 & = & 1 & & -6y_5' \\
r_2 & = & 1 & -2y_4' & -5y_5' \\
r_3 & = & 1 & -3y_4' & -3y_5'
\end{array}
\tag{4.20}
$$

$$
\begin{array}{rcllll}
s_4 & = & 1 & -x_1' & -4x_3' & \\
x_2' & = & \frac{1}{2} & & -\frac{3}{2}x_3' & -\frac{1}{2}s_5
\end{array}
\tag{4.22}
$$

- By the algorithm rule, since s_5 just left, y_5' must be next to enter
- All of r_1, r_2, r_3 clash with y_5'
- Have to apply the minimum ratio test

$$v = c + qu + T.$$

- *u* is entering variable, *c* is a constant, *T* is term with all other variables
- Variable to leave satisfies *min* |*q/c*|
- In this case, r_1

Lemke-Howson – Example

$$\begin{array}{lcllll} y_5' & = & \frac{1}{6} & & -\frac{1}{6}r_1 \\ r_2 & = & \frac{1}{6} & -2y_4' & +\frac{5}{6}r_1 \\ r_3 & = & \frac{1}{2} & -3y_4' & +\frac{1}{2}r_1 \end{array} \qquad (4.23)$$

$$\begin{array}{lclllll} s_4 & = & 1 & -x_1' & -4x_3' & \\ x_2' & = & \frac{1}{2} & & -\frac{3}{2}x_3' & -\frac{1}{2}s_5 \end{array} \qquad (4.22)$$

- r_1 leaves, yielding 4.23
- So x_1' must enter. Clashes with s_4 only. So s_4 leaves. 4.22 updates to:

$$\begin{array}{lcllll} x_1' & = & 1 & -4x_3' & -s_4 & \\ x_2' & = & \frac{1}{2} & -\frac{3}{2}x_3' & & -\frac{1}{2}s_5 \end{array} \qquad (4.24)$$

- Next, y_4' must enter. r2 and r3 clash, min. ratio gives r2 must leave

Lemke-Howson – Example

$$
\begin{array}{lclll}
y_5' & = & \frac{1}{6} & -\frac{1}{6}r_1 & \\
y_4' & = & \frac{1}{12} & +\frac{5}{12}r_1 & -\frac{1}{2}r_2 \\
r_3 & = & \frac{1}{4} & -\frac{3}{4}r_1 & +\frac{3}{2}r_2
\end{array}
\qquad (4.25)
$$

$$
\begin{array}{lcllll}
x_1' & = & 1 & -4x_3' & -s_4 & \\
x_2' & = & \frac{1}{2} & -\frac{3}{2}x_3' & & -\frac{1}{2}s_5
\end{array}
\qquad (4.24)
$$

- On the LHS, a non-zero variable appears for each action (i.e. either that action is played, or it has a slack and is suboptimal).
- So we've solved the LCP. All non-basis variables are 0, so we get x' = (1, ½, 0); y' = (1/12,1/6). Renormalizing to get a probability distribution, x' = (2/3, 1/3, 0); y' = (1/3,2/3).
- <x', y'> is our Nash equilibrium.

Support-Enumeration Method

Heuristic – Searching the space of supports

- Suppose we already knew the support of the Nash equilibrium. That is, which actions are best response.
- Could we then solve for the probabilities we should assign to each action?
- Yes – we can write an LP
- So, the CNE problem is reduced to guessing the right support

Heuristic – Searching the space of supports

- Suppose we already knew the support of the Nash equilibrium. That is, which actions are best response.
- Could we then solve for the probabilities we should assign to each action?
- Yes – we can write an LP
- So, the CNE problem is reduced to guessing the right support

Feasibility Program

Given a support profile $\sigma = (\sigma_1, \sigma_2)$

$$\sum_{a_{-i} \in \sigma_{-i}} p(a_{-i}) u_i(a_i, a_{-i}) = v_i \qquad \forall i \in \{1,2\}, a_i \in \sigma_i \tag{4.26}$$

$$\sum_{a_{-i} \in \sigma_{-i}} p(a_{-i}) u_i(a_i, a_{-i}) \leq v_i \qquad \forall i \in \{1,2\}, a_i \notin \sigma_i \tag{4.27}$$

$$p_i(a_i) \geq 0 \qquad \forall i \in \{1,2\}, a_i \in \sigma_i \tag{4.28}$$

$$p_i(a_i) = 0 \qquad \forall i \in \{1,2\}, a_i \notin \sigma_i \tag{4.29}$$

$$\sum_{a_i \in \sigma_i} p_i(a_i) = 1 \qquad \forall i \in \{1,2\} \tag{4.30}$$

Eliminating Some Actions

- We can safely prune any actions that are strictly worse than another given the current support:

Definition 4.2.2 (Conditionally strictly dominated action) *An action* $a_i \in A_i$ *is* conditionally strictly dominated, *given a profile of sets of available actions* $R_{-i} \subseteq A_{-i}$ *for the remaining agents, if the following condition holds:* $\exists a_i' \in A_i \ \forall a_{-i} \in R_{-i} : \quad u_i(a_i, a_{-i}) < u_i(a_i', a_{-i})$.

Support-Enumeration Method

- **forall** *support size profiles* $x = (x_1, x_2)$*, sorted in increasing order of, first,* $|x_1 - x_2|$ *and, second,* $(x_1 + x_2)$ **do**
  - **forall** $\sigma_1 \subseteq A_1$ *s.t.* $|\sigma_1| = x_1$ **do**
    - $A'_2 \leftarrow \{a_2 \in A_2 \text{ not conditionally dominated, given } \sigma_1\}$
    - **if** $\nexists a_1 \in \sigma_1$ *conditionally dominated, given* A'_2 **then**
      - **forall** $\sigma_2 \subseteq A'_2$ *s.t.* $|\sigma_2| = x_2$ **do**
        - **if** $\nexists a_1 \in \sigma_1$ *conditionally dominated, given* σ_2 **and***TGS is satisfiable for* $\sigma = (\sigma_1, \sigma_2)$ **then**
          - **return** *the solution found; it is a NE*

Figure 4.6: The SEM algorithm

Faster than Lemke-Howson on most games in the literature.