1 Motivation
Motivation

- **We know:** In finite strategic games, mixed-strategy Nash equilibria are guaranteed to **exist**.
- **We don’t know:** How to systematically **find them**?
- **Challenge:** There are **infinitely many** mixed strategy profiles to consider. How to do this in finite time?

This chapter:

- Computation of mixed-strategy Nash equilibria for **finite zero-sum games**.
- Computation of mixed-strategy Nash equilibria for **general finite two player games**.
2 Linear Programming
Digression:
We briefly discuss linear programming because we will use this technique to find Nash equilibria.

Goal of linear programming:
Solving a system of linear inequalities over $n$ real-valued variables while optimizing some linear objective function.
Example

Production of two sorts of items with time requirements and profit per item. Objective: Maximize profit.

<table>
<thead>
<tr>
<th></th>
<th>Cutting</th>
<th>Assembly</th>
<th>Postproc.</th>
<th>Profit per item</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x) sort 1</td>
<td>25</td>
<td>60</td>
<td>68</td>
<td>30</td>
</tr>
<tr>
<td>(y) sort 2</td>
<td>75</td>
<td>60</td>
<td>34</td>
<td>40</td>
</tr>
<tr>
<td>per day</td>
<td>≤ 450</td>
<td>≤ 480</td>
<td>≤ 476</td>
<td>maximize!</td>
</tr>
</tbody>
</table>

Goal: Find numbers of pieces $x$ of sort 1 and $y$ of sort 2 to be produced per day such that the resource constraints are met and the objective function is maximized.
Example (ctd., formalization)

\[ x \geq 0, \ y \geq 0 \quad (1) \]
\[ 25x + 75y \leq 450 \quad \text{(or } y \leq 6 - \frac{1}{3}x) \quad (2) \]
\[ 60x + 60y \leq 480 \quad \text{(or } y \leq 8 - x) \quad (3) \]
\[ 68x + 34y \leq 476 \quad \text{(or } y \leq 14 - 2x) \quad (4) \]
\[ \text{maximize } z = 30x + 40y \quad (5) \]

- Inequalities (1)–(4): Admissible solutions
  (They form a convex set in \( \mathbb{R}^2 \).)
- Line (5): Objective function
**Linear Programming**

Example (ctd., visualization)

\[
x \geq 0, \quad y \geq 0 \\
y \leq 6 - \frac{1}{3}x \\
y \leq 8 - x \\
y \leq 14 - 2x \\
\text{max } z = 30x + 40y
\]

\[
x = 3, \quad y = 5 \implies \text{optimal solution at } (3, 5)
\]
Linear Programming

Definition (Linear program)

A linear program (LP) in standard from consists of

- $n$ real-valued variables $x_i$;  $n$ coefficients $b_i$;
- $m$ constants $c_j$;  $n \cdot m$ coefficients $a_{ij}$;
- $m$ constraints of the form

$$c_j \leq \sum_{i=1}^{n} a_{ij} x_i,$$

- and an objective function to be minimized ($x_i \geq 0$)

$$\sum_{i=1}^{n} b_i x_i.$$
Linear Programming

Solution of an LP: assignment of values to the \( x_i \) satisfying the constraints and minimizing the objective function.

Remarks:

- Maximization instead of minimization: easy, just change the signs of all the \( b_i \)’s, \( i = 1, \ldots, n \).
- Equalities instead of inequalities: \( x + y \leq c \) if and only if there is a \( z \geq 0 \) such that \( x + y + z = c \) (\( z \) is called a slack variable).
Linear Programming

Solution algorithms:

- Usually, one uses the **simplex algorithm** (which is worst-case exponential!).
- There are also polynomial-time algorithms such as interior-point or ellipsoid algorithms.

Tools and libraries:

- `lp_solve`
- CLP
- GLPK
- CPLEX
- gurobi
3 Zero-Sum Games
We start with finite zero-sum games for two reasons:

- They are easier to solve than general finite two-player games.
- Understanding how to solve finite zero-sum games facilitates understanding how to solve general finite two-player games.
Motivation
Linear Programming
Zero-Sum Games
General Finite Two-Player Games
Summary

In the following, we will exploit the zero-sum property of a game $G$ when searching for mixed-strategy Nash equilibria. For that, we need the following result.

**Proposition**

Let $G$ be a finite zero-sum game. Then the mixed extension of $G$ is also a zero-sum game.

**Proof.**

Homework.
Let $G$ be a finite zero-sum game with mixed extension $G'$. Then we know the following:

1. **Previous proposition implies:** $G'$ is also a zero-sum game.
2. **Nash’s theorem implies:** $G'$ has a Nash equilibrium.
3. **Maximinimizer theorem + (1) + (2) implies:** Nash equilibria and pairs of maximinimizers in $G'$ are the same.
Consequence:
When looking for mixed-strategy Nash equilibria in $G$, it is sufficient to look for pairs of maximinimizers in $G'$.

Method: Linear Programming
Approach:

- Let $G = \langle N, (A_i)_{i \in N}, (u_i)_{i \in N} \rangle$ be a finite zero-sum game:
  - $N = \{1, 2\}$.
  - $A_1$ and $A_2$ are finite.
  - $U_1(\alpha, \beta) = -U_2(\alpha, \beta)$ for all $\alpha \in \Delta(A_1), \beta \in \Delta(A_2)$.

- Player 1 looks for a maximimimizer mixed strategy $\alpha$.

- For each possible $\alpha$ of player 1:
  - Determine expected utility against best response of pl. 2.
    (Only need to consider finitely many pure candidates for best responses because of Support Lemma).
  - Maximize expected utility over all possible $\alpha$. 
Linear Program Encoding

**Result:** maximinimizer $\alpha$ for player 1 in $G'$

(= Nash equilibrium strategy for player 1)

**Analogously:** obtain maximinimizer $\beta$ for player 2 in $G'$

(= Nash equilibrium strategy for player 2)

**With maximinimizer theorem:** we can combine $\alpha$ and $\beta$
into a Nash equilibrium.
"For each possible $\alpha$ of player 1, determine expected utility against best response of player 2, and maximize."

translates to the following LP:

\[ \alpha(a) \geq 0 \quad \text{for all } a \in A_1 \]
\[ \sum_{a \in A_1} \alpha(a) = 1 \]
\[ U_1(\alpha, b) = \sum_{a \in A_1} \alpha(a) \cdot u_1(a,b) \geq u \quad \text{for all } b \in A_2 \]

Maximize $u$. 
Linear Program Encoding

Example (Matching pennies)

<table>
<thead>
<tr>
<th></th>
<th>$H$</th>
<th>$T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H$</td>
<td>1, -1</td>
<td>-1, 1</td>
</tr>
<tr>
<td>$T$</td>
<td>-1, 1</td>
<td>1, -1</td>
</tr>
</tbody>
</table>

Linear program for player 1:
Maximize $u$ subject to the constraints

\[
\alpha(H) \geq 0, \quad \alpha(T) \geq 0, \quad \alpha(H) + \alpha(T) = 1,
\]

\[
\alpha(H) \cdot u_1(H, H) + \alpha(T) \cdot u_1(T, H) = \alpha(H) - \alpha(T) \geq u,
\]

\[
\alpha(H) \cdot u_1(H, T) + \alpha(T) \cdot u_1(T, T) = -\alpha(H) + \alpha(T) \geq u.
\]

Solution: $\alpha(H) = \alpha(T) = 1/2, u = 0.$
Remark: There is an alternative encoding based on the observation that in zero-sum games that have a Nash equilibrium, maximinimization and minimaximization yield the same result.

Idea: Formulate linear program with inequalities

\[ U_1(a, \beta) \leq u \quad \text{for all } a \in A_1 \]

and minimize \( u \). Analogously for \( \beta \).
4 General Finite Two-Player Games
For general finite two-player games, the LP approach does not work.

Instead, use instances of the linear complementarity problem (LCP):

- Linear (in-)equalities as with LPs.
- Additional constraints of the form \(x_i \cdot y_i = 0\)
  (or equivalently \(x_i = 0 \lor y_i = 0\))
  for variables \(X = \{x_1, \ldots, x_k\}\) and \(Y = \{y_1, \ldots, y_k\}\), and
  \(i \in \{1, \ldots, k\}\).
- No objective function.

With LCPs, we can compute Nash equilibria for arbitrary finite two-player games.
General Finite Two-Player Games

Let $A_1$ and $A_2$ be finite and let $(\alpha, \beta)$ be a Nash equilibrium with payoff profile $(u, v)$. Then consider this LCP encoding:

\[ u - U_1(a, \beta) \geq 0 \quad \text{for all } a \in A_1 \quad (6) \]
\[ v - U_2(\alpha, b) \geq 0 \quad \text{for all } b \in A_2 \quad (7) \]
\[ \alpha(a) \cdot (u - U_1(a, \beta)) = 0 \quad \text{for all } a \in A_1 \quad (8) \]
\[ \beta(b) \cdot (v - U_2(\alpha, b)) = 0 \quad \text{for all } b \in A_2 \quad (9) \]
\[ \alpha(a) \geq 0 \quad \text{for all } a \in A_1 \quad (10) \]
\[ \beta(b) \geq 0 \quad \text{for all } b \in A_2 \quad (12) \]
\[ \alpha(a) \geq 0 \quad \text{for all } a \in A_1 \quad (10) \]
\[ \sum_{a \in A_1} \alpha(a) = 1 \quad (11) \]
\[ \sum_{b \in A_2} \beta(b) = 1 \quad (13) \]
Remarks about the encoding:

- In (8) and (9): for instance,
  \[ \alpha(a) \cdot (u - U_1(a, \beta)) = 0 \]
  if and only if
  \[ \alpha(a) = 0 \quad \text{or} \quad u - U_1(a, \beta) = 0. \]

  This holds in every Nash equilibrium, because:
  - if \( a \notin \text{supp}(\alpha) \), then \( \alpha(a) = 0 \), and
  - if \( a \in \text{supp}(\alpha) \), then \( a \in B_1(\beta) \), thus \( U_1(a, \beta) = u \).

- With additional variables, the above LCP formulation can be transformed into LCP normal form.
Theorem

A mixed strategy profile \((\alpha, \beta)\) with payoff profile \((u, v)\) is a Nash equilibrium if and only if it is a solution to the LCP encoding over \((\alpha, \beta)\) and \((u, v)\).

Proof.

- Nash equilibria are solutions to the LCP: Obvious because of the support lemma.

- Solutions to the LCP are Nash equilibria: Let \((\alpha, \beta, u, v)\) be a solution to the LCP. Because of (10)–(13), \(\alpha\) and \(\beta\) are mixed strategies.
Proof (ctd.)

Solutions to the LCP are Nash equilibria (ctd.): Because of (6), $u$ is at least the maximal payoff over all possible pure responses, and because of (8), $u$ is exactly the maximal payoff.

If $\alpha(a) > 0$, then, because of (8), the payoff for player 1 against $\beta$ is $u$.

The linearity of the expected utility implies that $\alpha$ is a best response to $\beta$.

Analogously, we can show that $\beta$ is a best response to $\alpha$ and hence $(\alpha, \beta)$ is a Nash equilibrium with payoff profile $(u, v)$.
Solution Algorithm for LCPs

Naïve algorithm:

Enumerate all $$(2^n - 1) \cdot (2^m - 1)$$ possible pairs of support sets.

For each such pair $$(\text{supp}(\alpha), \text{supp}(\beta))$$:

- Convert the LCP into an LP:
  - Linear (in-)equalities are preserved.
  - Constraints of the form $$\alpha(a) \cdot (u - U_1(a, \beta)) = 0$$ are replaced by a new linear equality:
    - $$u - U_1(a, \beta) = 0$$, if $$a \in \text{supp}(\alpha)$$, and
    - $$\alpha(a) = 0$$, otherwise,

  Analogously for $$\beta(b) \cdot (v - U_2(\alpha, b)) = 0$$.

- Objective function: maximize constant zero function.

- Apply solution algorithm for LPs to the transformed program.
Solution Algorithm for LCPs

- Runtime of the naïve algorithm: $O(p(n + m) \cdot 2^{n+m})$, where $p$ is some polynomial.
- Better in practice: Lemke-Howson algorithm.
- Complexity:
  - unknown whether $\text{LcpSolve} \in \mathsf{P}$.
  - $\text{LcpSolve} \in \mathsf{NP}$ is clear
    (naïve algorithm can be seen as a nondeterministic polynomial-time algorithm).
5 Summary
Computation of mixed-strategy Nash equilibria for finite zero-sum games using linear programs.
\[\Rightarrow\] polynomial-time computation

Computation of mixed-strategy Nash equilibria for general finite two player games using linear complementarity problem.
\[\Rightarrow\] computation in \( \text{NP} \).