

Algorithmic game theory

Martin Balko

3rd lecture

October 19th 2026



Proof of the Minimax Theorem

The Minimax Theorem

The Minimax Theorem (Theorem 2.21)

Every zero-sum game has worst-case optimal strategies, computable in polynomial time. There is a number v such that, for any worst-case optimal strategies x^* and y^* , the strategy profile (x^*, y^*) is a Nash equilibrium and $\beta(x^*) = (x^*)^\top M y^* = \alpha(y^*) = v$.



Figure: John von Neumann (1903–1957) and Oskar Morgenstern (1902–1977).

Sources: <https://en.wikiquote.org> and <https://austriainusa.org>

The Minimax Theorem

The Minimax Theorem (Theorem 2.21)

Every zero-sum game has worst-case optimal strategies, computable in polynomial time. There is a number v such that, for any worst-case optimal strategies x^* and y^* , the strategy profile (x^*, y^*) is a Nash equilibrium and $\beta(x^*) = (x^*)^\top M y^* = \alpha(y^*) = v$.

- Recall that $\beta(x) = \min_{y \in S_2} x^\top M y$ and $\alpha(y) = \max_{x \in S_1} x^\top M y$ are the best possible cost of player 2 to x and payoff of player 1 to y , respectively.
- Also, the worst-case optimal strategy $\bar{x}$ for player 1, satisfies

$$\beta(\bar{x}) = \max_{x \in S_1} \beta(x).$$

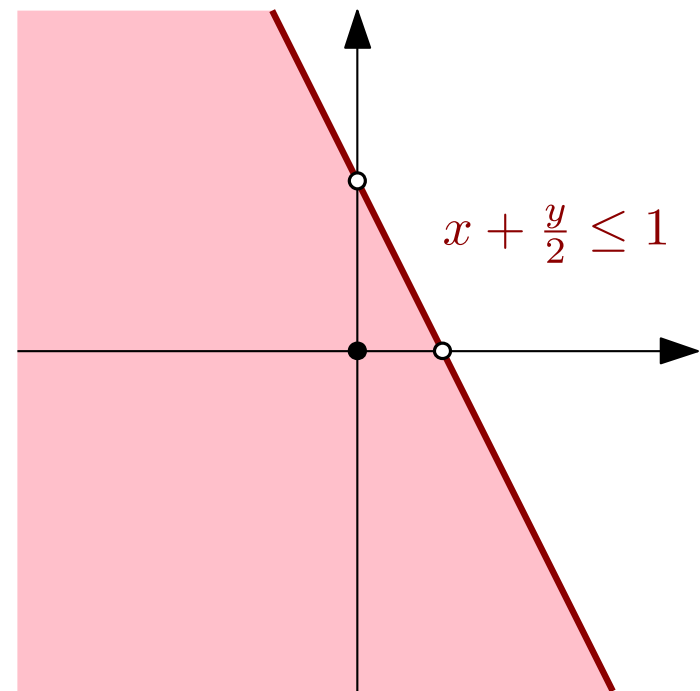
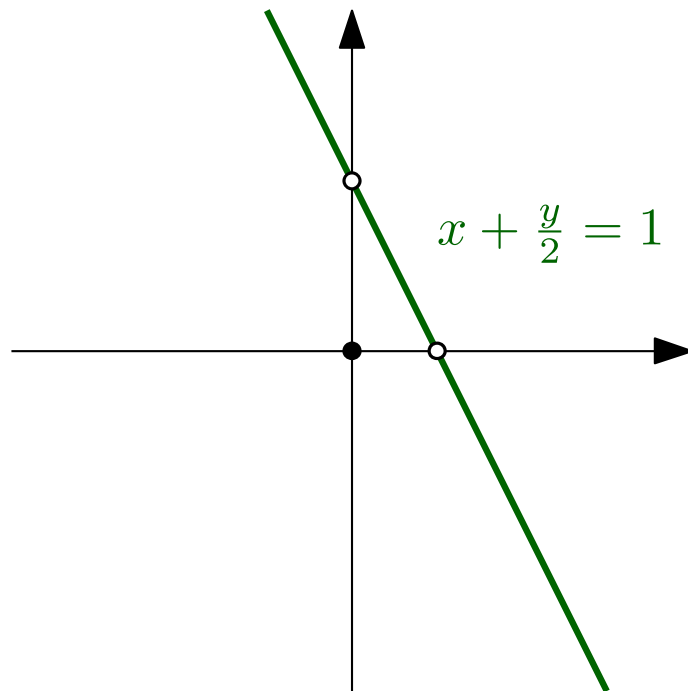
- The worst-case optimal strategy $\bar{y}$ for player 2, satisfies

$$\alpha(\bar{y}) = \min_{y \in S_2} \alpha(y).$$

- We prove the theorem using linear programming.

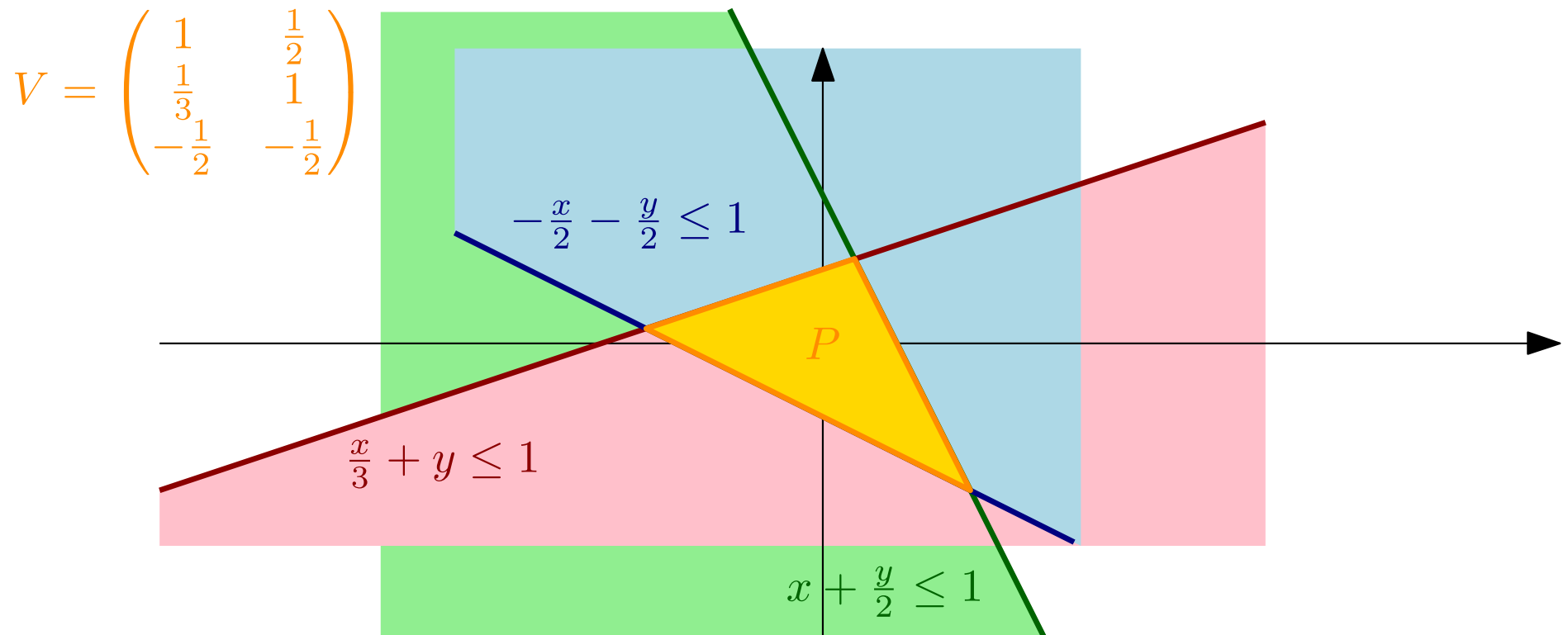
Preliminaries from geometry

- A **hyperplane** in $\mathbb{R}^d$ is a set $\{x \in \mathbb{R}^d : v^\top x = w\}$ for some nonzero $v \in \mathbb{R}^d$ and $w \in \mathbb{R}$.
- A **halfspace** in $\mathbb{R}^d$ is a set $\{x \in \mathbb{R}^d : v^\top x \leq w\}$.



Preliminaries from geometry

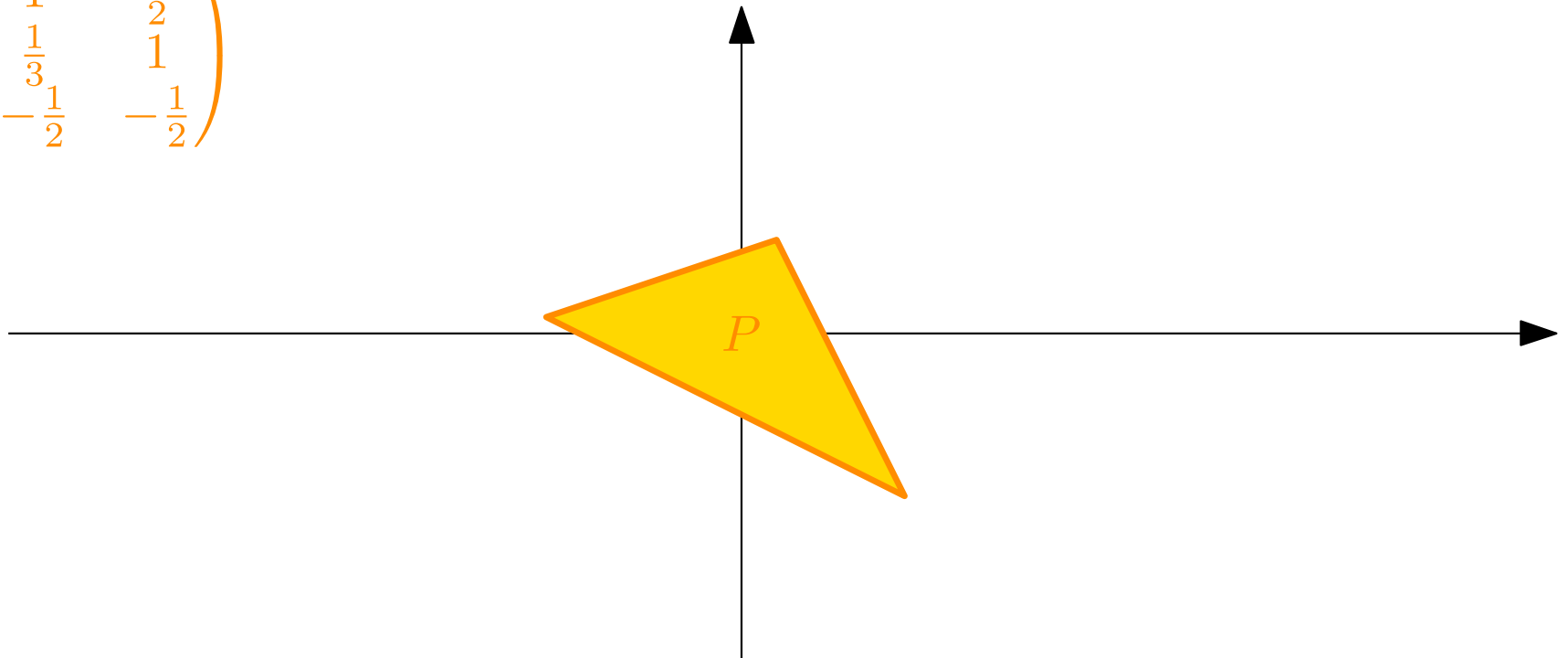
- A (convex) **polyhedron** P in $\mathbb{R}^d$ is an intersection of finitely many halfspaces in $\mathbb{R}^d$. That is, $P = \{x \in \mathbb{R}^d : Vx \leq u\}$ for some $V \in \mathbb{R}^{n \times d}$ and $u \in \mathbb{R}^n$, where n is the number of halfspaces determining P .
- A bounded polyhedron is called a **polytope**. A d -dimensional polytope is **simple** if no point is contained in more than d facets.



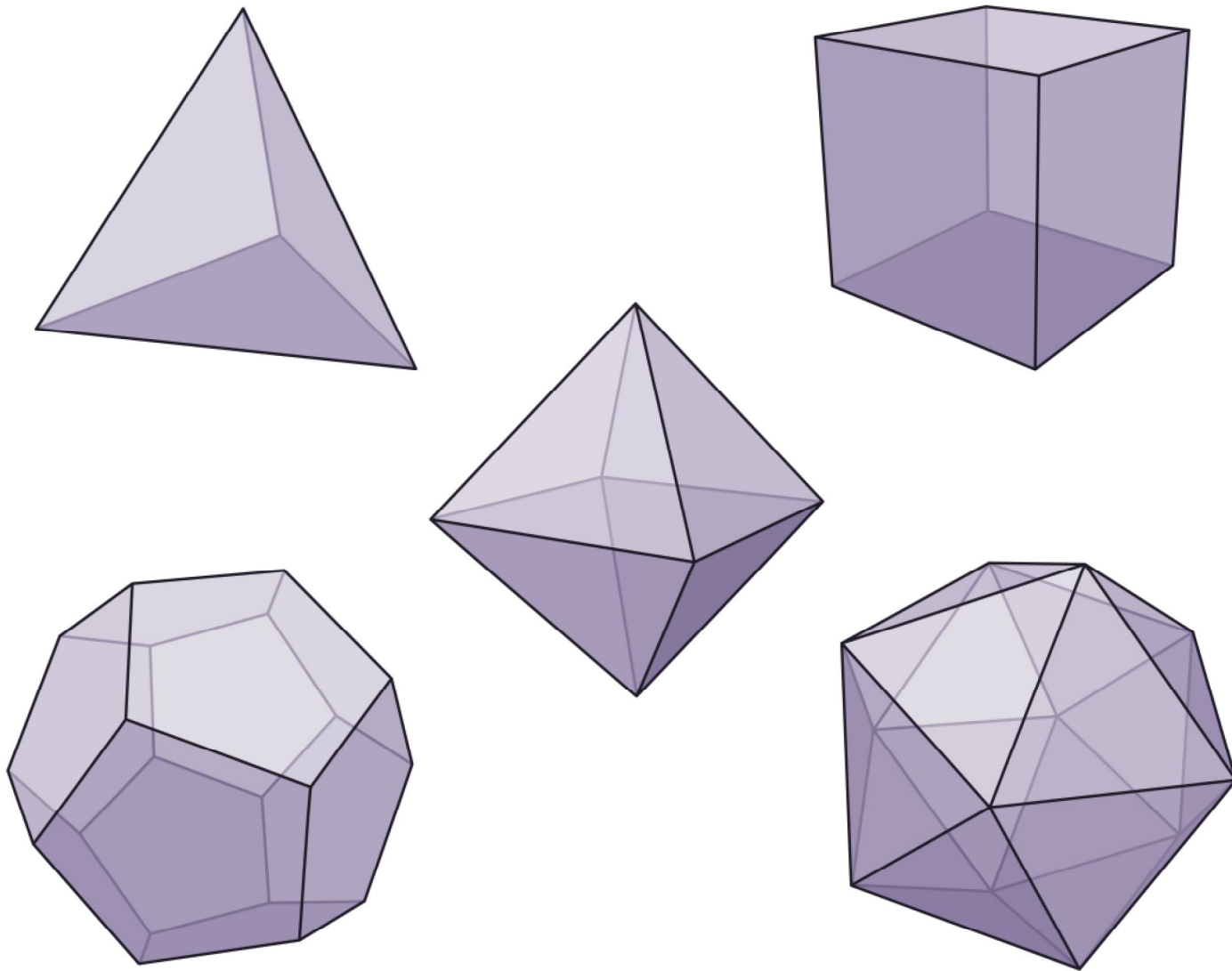
Preliminaries from geometry

- A (convex) **polyhedron** P in $\mathbb{R}^d$ is an intersection of finitely many halfspaces in $\mathbb{R}^d$. That is, $P = \{x \in \mathbb{R}^d : Vx \leq u\}$ for some $V \in \mathbb{R}^{n \times d}$ and $u \in \mathbb{R}^n$, where n is the number of halfspaces determining P .
- A bounded polyhedron is called a **polytope**. A d -dimensional polytope is **simple** if no point is contained in more than d facets.

$$V = \begin{pmatrix} 1 & \frac{1}{2} \\ \frac{1}{3} & 1 \\ -\frac{1}{2} & -\frac{1}{2} \end{pmatrix}$$

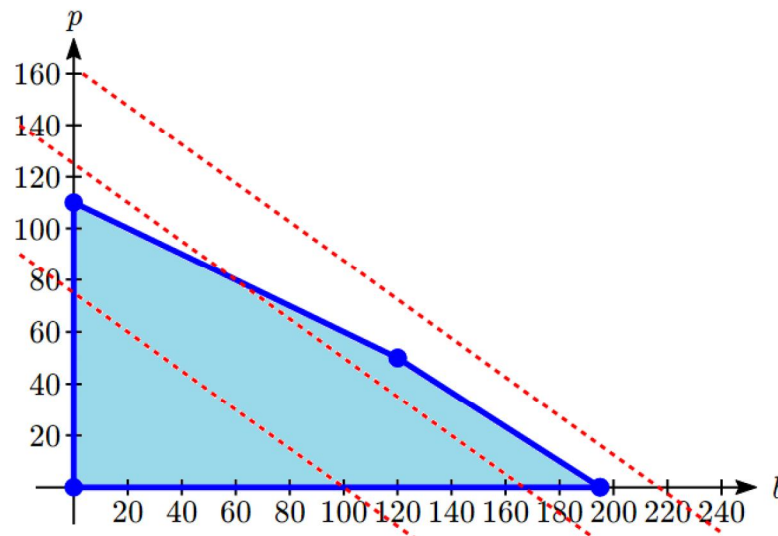


Examples of polytopes in $\mathbb{R}^3$



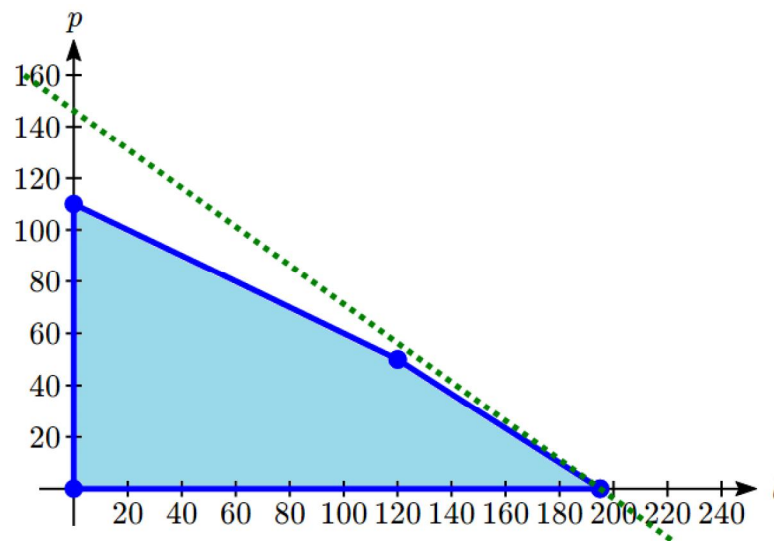
Linear programming

- A **linear program (LP)** is an optimization problem with a linear objective function and linear constraints.
- Every linear program P can be expressed in the **canonical form**: given $c \in \mathbb{R}^m$, $b \in \mathbb{R}^n$, and $A \in \mathbb{R}^{n \times m}$, we want to **maximize** $c^T x$ **subject to** the constraints $Ax \leq b$ and $x \geq 0$.
- Rational LPs can be solved in **polynomial time**. In practice, the **Simplex method** works, although it does not have a polynomial worst-case running time. The **Ellipsoid method** runs in polynomial time even in the worst-case.
- Solving linear programs graphically:



Linear programming

- A **linear program (LP)** is an optimization problem with a linear objective function and linear constraints.
- Every linear program P can be expressed in the **canonical form**: given $c \in \mathbb{R}^m$, $b \in \mathbb{R}^n$, and $A \in \mathbb{R}^{n \times m}$, we want to **maximize** $c^T x$ **subject to** the constraints $Ax \leq b$ and $x \geq 0$.
- Rational LPs can be solved in **polynomial time**. In practice, the **Simplex method** works, although it does not have a polynomial worst-case running time. The **Ellipsoid method** runs in polynomial time even in the worst-case.
- Solving linear programs graphically:



Source: https://en.wikipedia.org/wiki/Linear_programming#/media/File:Simplex_method_2D.png

Duality of linear programming

- The linear program P where we want to

maximize $c^\top x$ subject to the constraints $Ax \leq b$ and $x \geq 0$

is called the **primal linear program**.

- The associated **dual linear program** D is to

minimize $b^\top y$ subject to constraints $A^\top y \geq c$ and $y \geq 0$.

- “Solving a system of linear equalities from the rows-perspective instead of the columns-perspective”.

- The following **Duality Theorem** has several important consequences.

The Duality Theorem (Theorem 2.22)

If both linear programs P and D have feasible solutions, then they both have optimal solutions. Moreover, if x^* and y^* are optimal solutions of P and D , respectively, then $c^\top x^* = b^\top y^*$. That is, the maximum of P equals the minimum of D .

- Dual programs can be constructed for any linear program.

General recipe for duality

	Primal linear program	Dual linear program
Variables	$x_1, \dots, x_m$	$y_1, \dots, y_n$
Matrix	$A \in \mathbb{R}^{n \times m}$	$A^T \in \mathbb{R}^{m \times n}$
Right-hand side	$b \in \mathbb{R}^n$	$c \in \mathbb{R}^m$
Objective function	$\max c^T x$	$\min b^T y$
Constraints	i th constraint has $\leq$	$y_i \geq 0$
	$\geq$	$y_i \leq 0$
	$=$	$y_i \in \mathbb{R}$
	$x_j \geq 0$	j th constraint has $\geq$
	$x_j \leq 0$	$\leq$
	$x_j \in \mathbb{R}$	$=$

Table: A recipe for making dual programs.

Proof of the Minimax Theorem I

- We now proceed with the proof of the Minimax Theorem.

The Minimax Theorem (Theorem 2.21)

Every zero-sum game has worst-case optimal strategies, computable in polynomial time. There is a number v such that, for any worst-case optimal strategies x^* and y^* , the strategy profile (x^*, y^*) is a Nash equilibrium and $\beta(x^*) = (x^*)^\top M y^* = \alpha(y^*) = v$.

- We want to compute x^* such that $\beta(x^*) = \max_{x \in S_1} \beta(x)$ where $\beta(x) = \min_{y \in S_2} x^\top M y$ using LP. We first show how **not** to do it.
- Naive straightforward approach with variables $x_1, \dots, x_m$:

$$\text{maximize } \beta(x) \text{ subject to the constraints } \sum_{i=1}^m x_i = 1 \text{ and } x \geq 0.$$

- **In general, this is not LP!** (the objective function $\beta(x) = \min_{y \in S_2} x^\top M y$ is not linear in x) What can we compute with LP?

Proof of the Minimax Theorem II

- For fixed $x \in S_1$, we can compute a best response of 2 to x .
- We use the following linear program P with variables $y_1, \dots, y_n$:

$$(P) \quad \text{minimize } x^\top M y \text{ subject to } \sum_{j=1}^n y_j = 1 \text{ and } y \geq \mathbf{0}.$$

- Its dual is the following LP D with a single variable $x_0 \in \mathbb{R}$ (Exercise):

$$(D) \quad \text{maximize } x_0 \text{ subject to } \mathbf{1}x_0 \leq M^\top x.$$

- Since P and D are feasible, the Duality Theorem gives their common optimal value $\beta(x)$.
- Thus, if we treat $x_1, \dots, x_m$ as variables in D , we obtain the following linear program D' with variables $x_0, x_1, \dots, x_m$:

$$(D') \quad \text{maximize } x_0 \text{ subject to } \mathbf{1}x_0 - M^\top x \leq \mathbf{0}, \sum_{i=1}^m x_i = 1 \text{ and } x \geq \mathbf{0}.$$

- The optimum x^* of D' is a worst-case optimal strategy for 1!

Proof of the Minimax Theorem III

- Analogously, we can compute a **worst-case optimal strategy** y^* for 2 using this linear program P' with variables $y_0 \in \mathbb{R}$ and $y_1, \dots, y_n$:

$$(P') \quad \text{minimize } y_0 \text{ subject to } \mathbf{1}y_0 - My \geq \mathbf{0}, \sum_{j=1}^n y_j = 1 \text{ and } y \geq \mathbf{0}.$$

- Both LPs are feasible and bounded, proving existence. It remains to show that (x^*, y^*) is NE and $\beta(x^*) = (x^*)^\top My^* = \alpha(y^*) = v$.
- Using the **general recipe for duality**, we see that P' and D' are dual to each other! (**Exercise**)
- By the **Duality Theorem**, P' and D' have the same optimal value

$$\beta(x^*) = x_0^* = y_0^* = \alpha(y^*).$$

This value v is attained in any worst-case optimal strategy.

- By **part (c) of Lemma 2.20**, (x^*, y^*) is NE, that is, we have $\beta(x^*) = (x^*)^\top My^* = \alpha(y^*)$.



Nash equilibria in bimatrix games

Bimatrix games

- Since zero-sum games are solved now, we try to efficiently find Nash equilibria in **bimatrix games**, that is, games of 2 players (not necessarily zero-sum).
- Example: **Prisoner's dilemma**

	Testify	Remain silent
Testify	$(-2, -2)$	$(0, -3)$
Remain silent	$(-3, 0)$	$(-1, -1)$

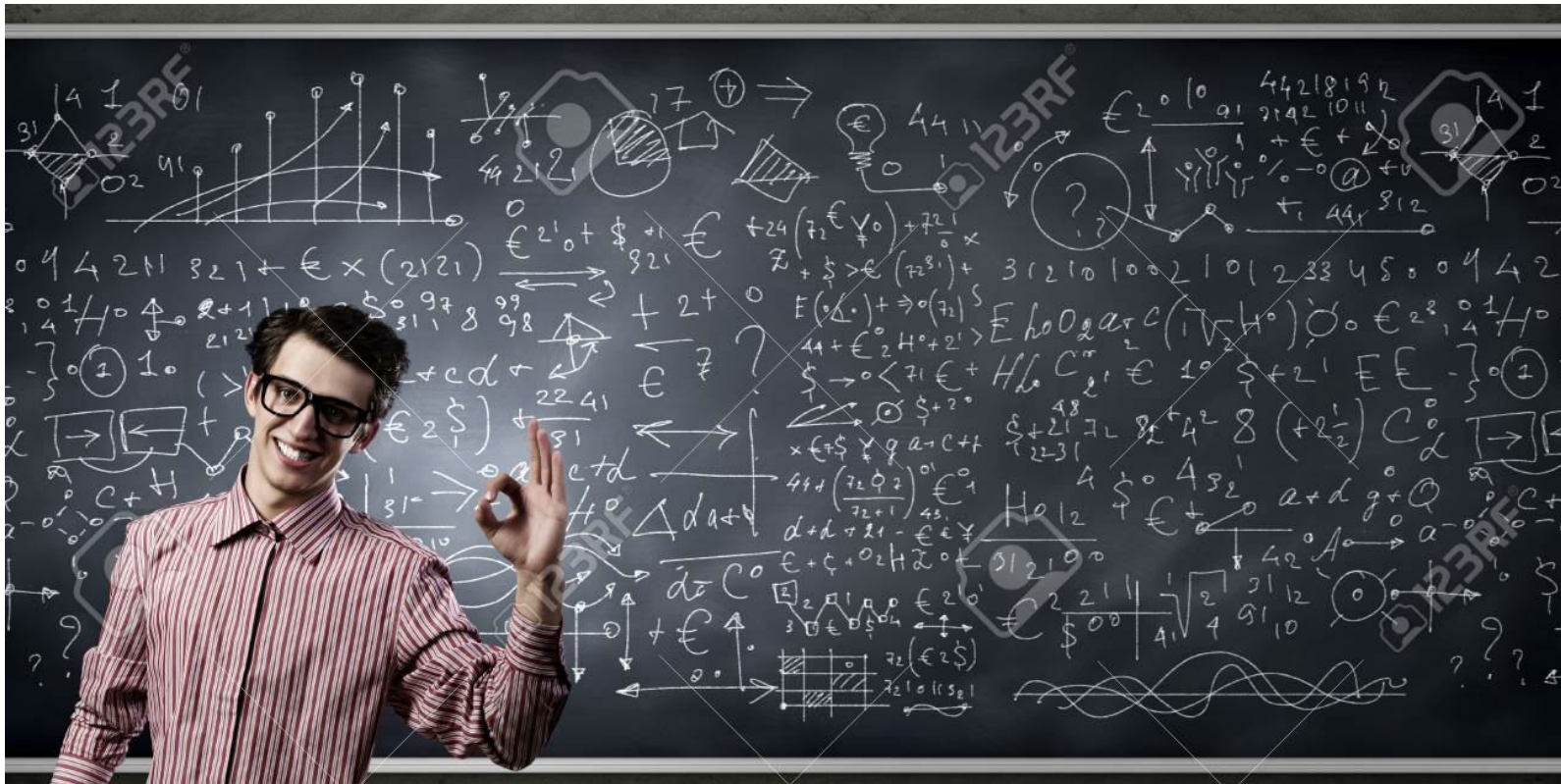


Bimatrix games examples: collaborative projects



Source: <https://filestage.io/>

Bimatrix games examples: education, knowledge sharing



Source: <https://www.123rf.com/>

Bimatrix games examples: the battle for Gotham's soul

	Cooperate	Detonate
Cooperate	$(0,0)$	$(0,1)$
Detonate	$(1,0)$	$(0,0)$

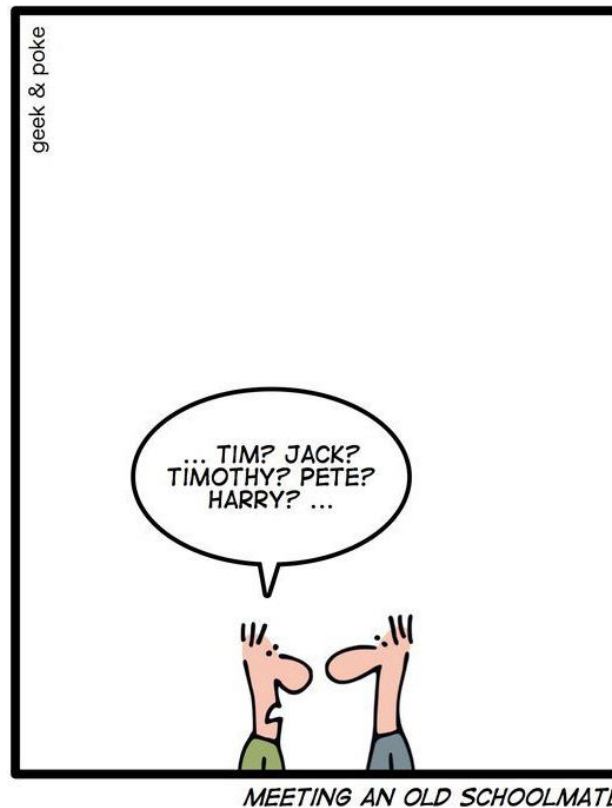


Sources: <https://www.cbr.com/>

Nash equilibria in bimatrix games by brute force

- We try to design an algorithm for finding Nash equilibria in games of two players (**bimatrix games**).
- We state some observations that yield a **brute-force algorithm**.

*SIMPLY EXPLAINED:
BRUTE FORCE ATTACK*



Source: <https://pinterest.com>

- Later, we show a **classical algorithm** for this problem.

Best response condition

- We first state the perhaps most useful observation in our course.
- The **support** of a mixed strategy is $Supp(s_i) = \{a_i \in A_i : s_i(a_i) > 0\}$.

Best response condition (Observation 2.23)

In a normal-form game $G = (P, A, u)$ of n players, for every player $i \in P$, a mixed strategy s_i is a best response to s_{-i} if and only if all pure strategies in the support of s_i are best responses to s_{-i} .

- In bimatrix games, we can **enumerate supports** and test each pair using LP.
- The heart of the problem is in finding the right supports.
- Once we have the right supports, the precise mixed strategies can be computed by solving a system of algebraic equations and inequalities (linear for two players).

Proof of the Best response condition

- **First**, assume every $a_i \in \text{Supp}(s_i)$ satisfies $u_i(a_i; s_{-i}) \geq u_i(s'_i; s_{-i})$ for every $s'_i \in S_i$. Then, for every $s'_i \in S_i$, the **linearity of u_i** implies

$$u_i(s) = \sum_{a_i \in \text{Supp}(s_i)} s_i(a_i) u_i(a_i; s_{-i}) \geq \sum_{a_i \in \text{Supp}(s_i)} s_i(a_i) u_i(s'_i; s_{-i}) = u_i(s'_i; s_{-i}).$$

- **Second**, assume s_i is a best response of i to s_{-i} . Suppose for contradiction there is $\bar{a}_i \in \text{Supp}(s_i)$ that is not a best response of i to s_{-i} . Since s_i is best response to s_{-i} , we have $u_i(\bar{a}_i; s_{-i}) < u_i(s_i; s_{-i})$. Since s_i is a best response to s_{-i} , we get $s_i(\bar{a}_i) < 1$. By the **linearity of u_i** , there is $\hat{a}_i \in \text{Supp}(s_i)$ with $u_i(\hat{a}_i; s_{-i}) > u_i(\bar{a}_i; s_{-i})$. We define a new mixed strategy $s_i^* \in S_i$ by setting $s_i^*(\bar{a}_i) = 0$, $s_i^*(\hat{a}_i) = s_i(\hat{a}_i) + s_i(\bar{a}_i)$ and keeping $s_i^*(a_i) = s_i(a_i)$ otherwise. Then, by the **linearity of u_i**

$$u_i(s_i^*; s_{-i}) = \sum_{a_i \in A_i} s_i^*(a_i) u_i(a_i; s_{-i}) > \sum_{a_i \in A_i} s_i(a_i) u_i(a_i; s_{-i}) = u_i(s),$$

a **contradiction**.



Best response condition in bimatrix games

- We can use this simple observation to design a **brute-force algorithm for finding NE** in bimatrix games.
- Let $G = (\{1, 2\}, A = A_1 \times A_2, u)$ be a bimatrix game.
- Let $A_1 = \{1, \dots, m\}$ and $A_2 = \{1, \dots, n\}$ (later considered disjoint).
- The payoffs u_1 and u_2 can be represented by matrices $M, N \in \mathbb{R}^{m \times n}$ as $M_{i,j} = u_1(i, j)$ and $N_{i,j} = u_2(i, j)$ for every $(i, j) \in A_1 \times A_2$.
- The **expected payoffs** of s with mixed strategy vectors x and y are then

$$u_1(s) = x^\top M y \quad \text{and} \quad u_2(s) = x^\top N y.$$

- By the **Best response condition**, x is a best response to y iff

$$\forall i \in A_1 : x_i > 0 \implies M_{i,*} y = \max\{M_{k,*} y : k \in A_1\}. \quad (1)$$

- Analogously, y is a best response to x iff

$$\forall j \in A_2 : y_j > 0 \implies N_{j,*}^\top x = \max\{N_{k,*}^\top x : k \in A_2\}. \quad (2)$$

NE by support enumeration I

- We consider only special bimatrix games (the reason will be clear later).
- A bimatrix game is **nondegenerate** if there are at most k pure best responses to every mixed strategy with support of size k .
 - “Most bimatrix games are nondegenerate” and there are perturbation methods to deal with degenerate games.
- Let $I \subseteq A_1$ and $J \subseteq A_2$ be supports in a nondegenerate game G .
- We define $|I| + |J|$ **variables** x_i for $i \in I$ and y_j for $j \in J$ that will represent non-zero values in mixed strategy vectors x and y .
- We define **equations** $\sum_{i \in I} x_i = 1$ and $\sum_{j \in J} y_j = 1$, and $|I| + |J|$ equations to ensure that the expected payoffs are equal and maximized at the support (but not necessarily over the entire A_1 or A_2):

$$\sum_{i \in I} N_{j,i}^T x_i = v \quad \text{and} \quad \sum_{j \in J} M_{i,j} y_j = u,$$

where u and v are two new variables. Note that they attain values $u = \max\{M_{i,*}y : i \in I\}$ and $v = \max\{N_{j,*}^T x : j \in J\}$.

NE by support enumeration II

- We have a system $S(I, J)$ of $|I| + |J| + 2$ variables $(x_i)_{i \in I}, (y_j)_{j \in J}, u, v$ and $|I| + |J| + 2$ linear equations.
- If the probabilities on the supports are positive and satisfy (1) and (2), then we have a NE by the Best response condition. If G is nondegenerate, then such a solution is unique if it exists (Exercise).
- It follows immediately from the Best response condition that supports of strategies in NE of a non-degenerate game have the same size.
- This suggests a simple algorithm for finding NE of G .
- Support enumeration: go through all possible supports $I \subseteq A_1$ and $J \subseteq A_2$ of size $k \in \{1, \dots, \min\{m, n\}\}$ and verify whether the supports I and J yield NE by solving the system $S(I, J)$ of linear equations.
- The running time is then about 4^n for $m = n$.

Example: Game of chicken

- We show the Support enumeration on the Game of chicken.

	Turn (1)	Go straight (2)
Turn (1)	(0, 0)	(-1, 1)
Go straight (2)	(1, -1)	(-10, -10)

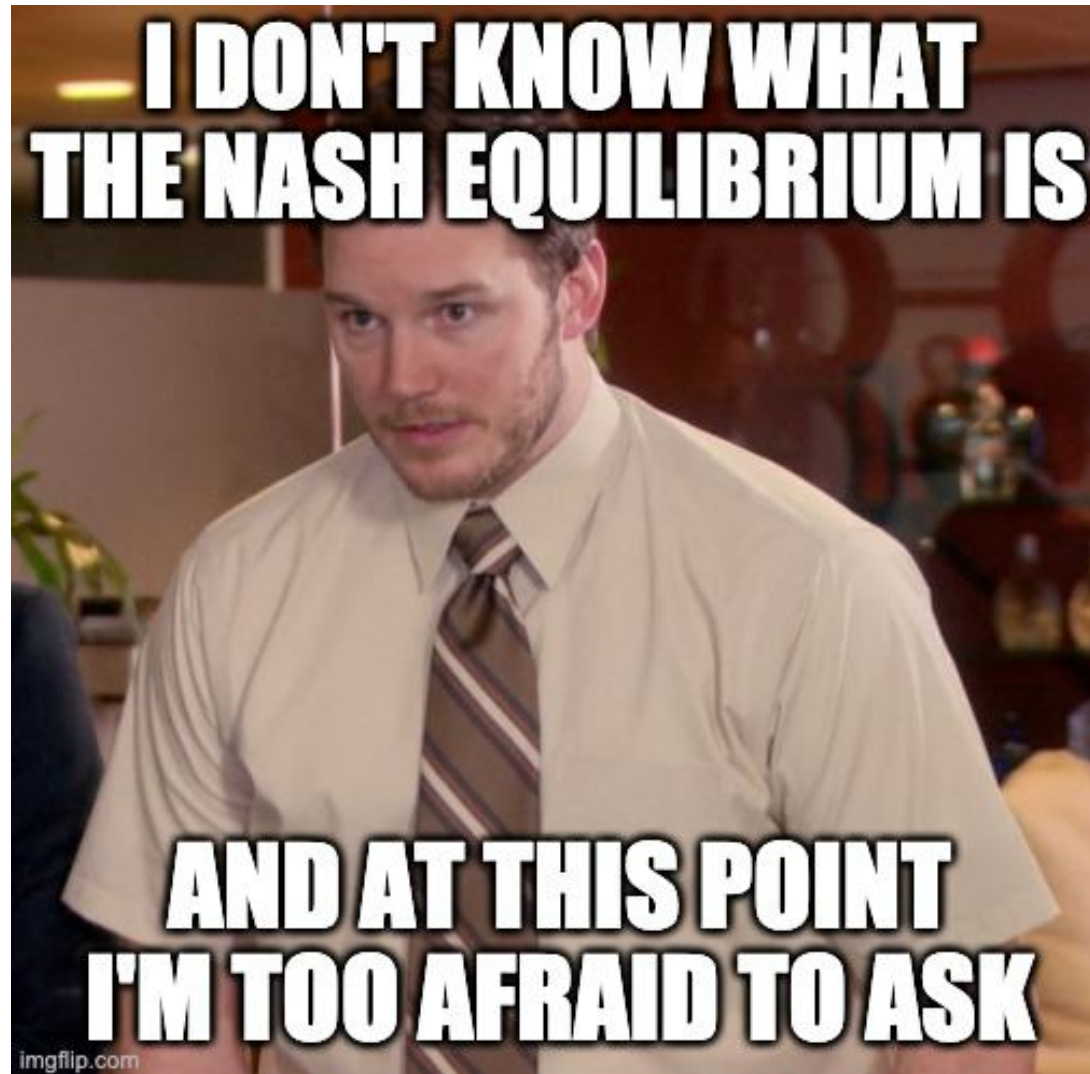
- That is, we have $M = \begin{pmatrix} 0 & -1 \\ 1 & -10 \end{pmatrix}$ and $N = \begin{pmatrix} 0 & 1 \\ -1 & -10 \end{pmatrix} = M^\top$.
- If $I = \{1, 2\}$ and $J = \{1, 2\}$, then we want to solve the following system of 6 equations with 6 variables x_1, x_2, y_1, y_2, u, v :

$$-x_2 = v, \quad x_1 - 10x_2 = v, \quad x_1 + x_2 = 1$$

$$-y_2 = u, \quad y_1 - 10y_2 = u, \quad y_1 + y_2 = 1$$

- This yields a unique solution $(x_1, x_2) = (\frac{9}{10}, \frac{1}{10})$ and $(y_1, y_2) = (\frac{9}{10}, \frac{1}{10})$. Since $x, y > 0$ and there is no better pure strategy, we have NE.

- Next lecture we learn the **Lemke–Howson algorithm**, a classical algorithm to find a Nash equilibrium in bimatrix games.



Thank you for your attention.