# Coins, Covers, and Strategies: A Journey Through Game Theory and Linear Programming

Optimal Decision Making — Spring 2025

This project is due on **May 23, 2025, at 23:59**. We provide an additional handout explaining some basic concepts of game theory and graph theory. You may form teams of up to three people. Each team should upload a **single** zip-file containing the pdf report and the code to Moodle. Make sure to clearly name all team members in your report.

## Overview

This project guides you through a journey of strategic thinking, optimization, and mathematical modeling — starting from intuition and ending in provable guarantees.

You will begin by playing a simple repeated game against a teaching assistant (TA), where your goal is to preserve as many coins as possible over 100 rounds. At first, you'll rely solely on intuition. As the project progresses, you will be introduced to powerful tools from game theory and linear programming to help you better analyze and improve your strategy.

Your final goal will be to understand and prove a fundamental result — **König's Theorem** — and to implement a linear programming solver that computes optimal strategies on real game boards.

**Important:** Throughout the project, you will encounter a series of labeled tasks (e.g., `Ph3-Task1`, `Ph4-Task3`, etc.). These are the only tasks that will be assessed.

Let the game begin.

## Phase 1: The Game Begins

*A mysterious 25 CHF has vanished from the TA budget. Was it a clerical error? A perfectly innocent coincidence? Or a daring act of strategic mischief by the students?*

The TAs, unwilling to chase students through paperwork and bureaucracy, have come up with a clever alternative to recover their lost money — through a game.

You are going to play a strategic game against the TA, repeated over 100 rounds. At the beginning of the game, each group of students will receive exactly 25 CHF in virtual coins — a full refund of their "alleged theft."

But here's the catch: the TA will try to take those coins back, one by one, through a repeated game described below.

### Game Description

The game board is a $6 \times 6$ grid. Certain cells are marked — these are the only places where students are allowed to hide their coin. The game board with marked cells is shown in Figure 1. At each round:

- **Students** choose one of the marked (red) cells and secretly hide a coin there.
- **The TA** picks a row or a column to check.
- If the TA's chosen row or column contains the hidden coin, the TA takes the coin. If not, the students keep their coin.
- The game will be repeated for a total of 100 rounds.

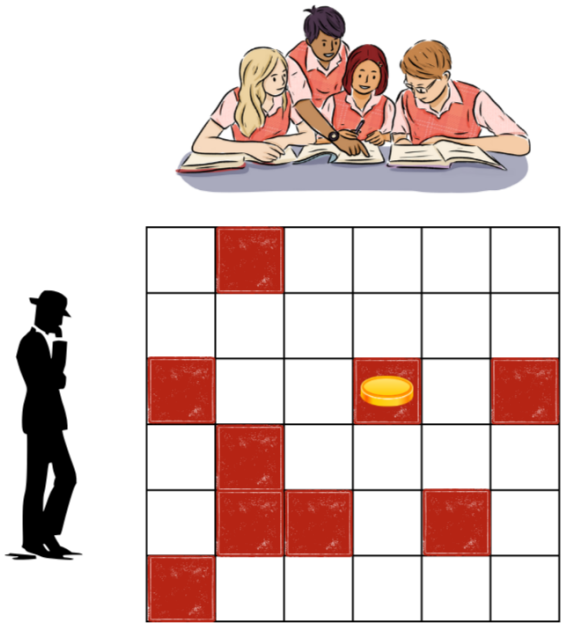

Figure 1: The game board showing the marked cells (in red) where coins can be hidden

### Strategic Rule

Before the game begins, **both sides must choose a fixed strategy**, and **they are not allowed to change it during the game**. That means:

- Students must decide in advance how they will hide the coin in each round. For example:
  - Always hide the coin in the same cell.
  - Choose uniformly at random among all valid cells.
  - Or any other strategy — deterministic or stochastic — but fixed throughout the 100 rounds.
- The TA will also select a fixed strategy beforehand — a deterministic or randomized rule to choose rows or columns — and will use it consistently over all 100 rounds.

### Your Challenge

If you lose more than 25 coins, you'll be in debt — and have to pay out of your pocket. Can you survive 100 rounds with a clever strategy? Can you make it out with coins to spare? Let the game begin.

## Phase 2: A Strategic Perspective on the Game

If you have started playing the game using your intuition, you might have already noticed something unsettling: you are losing coins — maybe even more than the 25 CHF you were given. That means you are paying from your own pocket!

But here is the surprising part: there is a clever way to play this game such that, no matter what the TA does, you are guaranteed to lose no more than 25 coins — and maybe even fewer. This is not about luck. It is about strategy. And behind that strategy lies a powerful mathematical framework called **game theory**. In

this phase, you will become familiar with foundational concepts from game theory that will help you analyze and optimize your strategy.

**Zero-sum games** in matrix form are popular models for analyzing competitive decision-making situations involving two players. Specifically, we distinguish player 1, who can choose one out of $m$ actions, and player 2, who can choose one out of $n$ actions. The possible payoffs of the player 1 are recorded in a matrix $A \in \mathbb{R}^{m \times n}$, that is, player 1 earns $A_{ij}$ if player 1 chooses action $i$ and player 2 chooses action $j$. Player 2 earns $-A_{ij}$ in the same scenario. Thus, the payoff of player 1 and player 2 always add up to 0; hence the name 'zero-sum game.'

The players need to select their actions simultaneously without knowledge of the adversary's strategy. In addition, the players are allowed to play **mixed strategies**, which randomize over the available actions. Formally, a mixed strategy is $p \in \Delta_d = \{p \in \mathbb{R}^d_+ : \sum_{i=1}^d p_i = 1\}$ selects action $i$ with probability $p_i$, with the number of the total available actions as $d$. Note that for player 1, we have $d = m$, whereas for player 2, we have $d = n$. In this exercise, we will use duality theory to formulate simple linear programs whose solutions correspond to the players' **optimal mixed strategies**, which maximize the expected payoffs. These optimal strategies form a **Nash equilibrium**, which exists thanks to the remarkable **minimax theorem**. These ideas will serve as your tools — and your shield — as you start building solid strategies to protect your coins.

Please refer to the accompanying document for a more detailed introduction to the related game theory concepts. Please read this material carefully before attempting to solve the questions.

## Phase 3: Bounding the Value of the Game

Now that you've seen the game in action and learned some foundational theory, it's time to analyze it more rigorously and compute the value of the game.

Let $p^\star$ denote the value of the game — that is, the expected amount students lose in one round when both players play optimally. We will now approach $p^\star$ from two directions:

1. We first show how the TA can guarantee a *minimum* return per round, which gives a **lower bound** on $p^\star$.
2. We then show how the students can guarantee a *maximum* loss per round, which gives an **upper bound** on $p^\star$.

### Step 1: A Lower Bound from a TA Strategy

Suppose the TA commits to a strategy that guarantees a return of at least $l$ coins per round, regardless of how the students choose to hide their coin. Then it must be that the value of the game satisfies $p^\star \geq l$.

More generally, consider a zero-sum game with a payoff matrix $A \in \mathbb{R}^{m \times n}$, and let $p^\star$ denote the value of the game:

$$\max_{x \in \Delta_m} \min_{y \in \Delta_n} x^\top A y = \min_{y \in \Delta_n} \max_{x \in \Delta_m} x^\top A y = p^\star,$$

where $\Delta_m$ and $\Delta_n$ denote the sets of mixed strategies (i.e., probability distributions) for the first and second players, respectively.

**Ph3-Task1 (3 points).** Construct the payoff matrix $A \in \mathbb{R}^{m \times n}$ corresponding to the specific game board shown in Figure 1.

**Ph3-Task2 (5 points).** Show that if the player 1 (TA) can find a strategy $\hat{x} \in \Delta_m$ such that

$$\forall y \in \Delta_n, \quad \hat{x}^\top A y \geq l,$$

then it follows that $p^\star \geq l$.

We now apply this principle to the strategy used by the TA in the game.

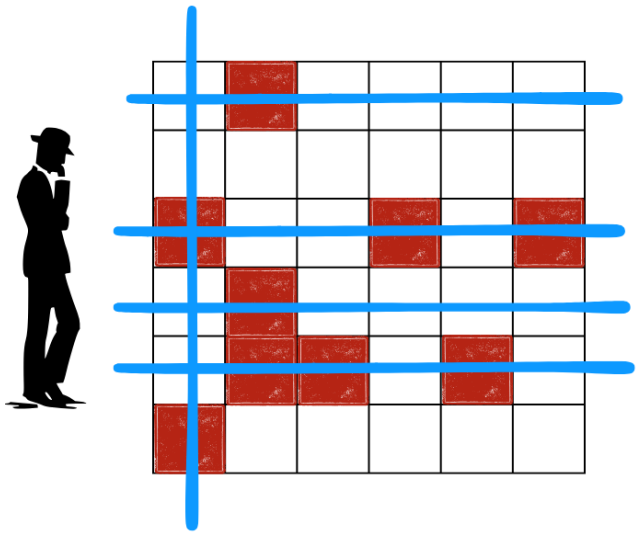

Figure 2: A line-cover that the TA can use for his strategy

- A **line-cover** is a collection of rows and columns that together cover all the marked (red) cells on the board. An example of a line-cover chosen by the TA is illustrated in Figure 2.
- The TA can choose a line **uniformly at random** from a line-cover.

**Ph3-Task3 (5 points).** Let $C$ be a line-cover of the board. Show that choosing uniformly from $C$ guarantees the TA a return of $1/|C|$ per round. Conclude that $p^\star \geq 1/|C|$.

### Step 2: An Upper Bound from a Students' Strategy

Now, suppose that the students commit to a strategy that ensures their coin is caught with probability at most $u$, no matter what the TA does.

**Ph3-Task4 (5 points).** Prove that if the player 2 (students) can find a strategy $\hat{y} \in \Delta_n$ such that

$$\forall x \in \Delta_m, \quad x^\top A\hat{y} \leq u,$$

then it follows that $p^\star \leq u$.

We now apply this principle to a strategy the students can use in the game:

- A **pairing** is a collection of marked (red) cells such that no two lie in the same row or column. This is called a "pairing" because it pairs specific rows with specific columns without overlap. An example of a pairing is illustrated in Figure 3.
- The students can choose a cell **uniformly at random** from a pairing.

**Ph3-Task5 (5 points).** Let $M$ be a pairing on the board. Show that if the students hide their coin uniformly at random over the cells in $M$, then the **expected number of coins lost per round** does not exceed $1/|M|$, no matter what strategy the TA uses. Conclude that $p^\star \leq 1/|M|$.

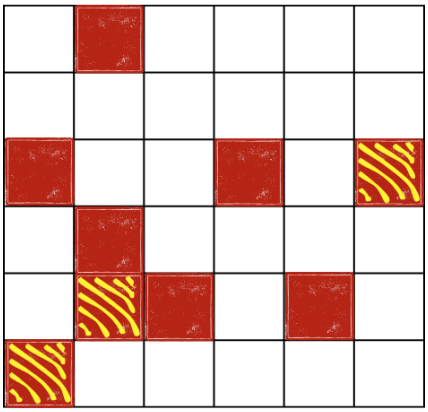

Figure 3: A pairing that students can use for their strategy

### Step 3: Putting It All Together

At this point, you have established that for any line-cover $C$ and any pairing $M$, the value of the game satisfies the bounds:

$$\frac{1}{|C|} \leq p^\star \leq \frac{1}{|M|}.$$

To tighten these bounds, you now need to identify the smallest possible line-cover and the largest possible pairing on the game board. These extremal values will allow you to pin down the exact value of the game.

**Ph3-Task6 (6 points).** Find a minimum-size line-cover and a maximum-size pairing. Then use these to obtain the exact value of the game $p^\star$. Finally, compute the total expected loss over 100 rounds using this value. Does this number match the amount of money you were given at the beginning of the game? What does that suggest about the fairness of the setup?

In the next phase, you will explore why this surprising equivalence between the minimum line-cover and the maximum pairing is not a coincidence — and how it relates to one of the most elegant results in this course: **duality theory**.

## Phase 4: Proving König's Theorem

In the previous phase, you observed that the size of a maximum pairing on your $6 \times 6$ game board equals the size of a minimum line-cover. In this phase, you will prove that this is not just a coincidence, but a general fact about a broad class of graphs.

This result is known as **König's Theorem**, and you will prove it using tools from linear programming. (You can refer to the accompanying handout for terminology on graphs, matchings, and vertex covers.)

**König's Theorem:** In any bipartite graph, the size of a maximum matching equals the size of a minimum vertex cover.

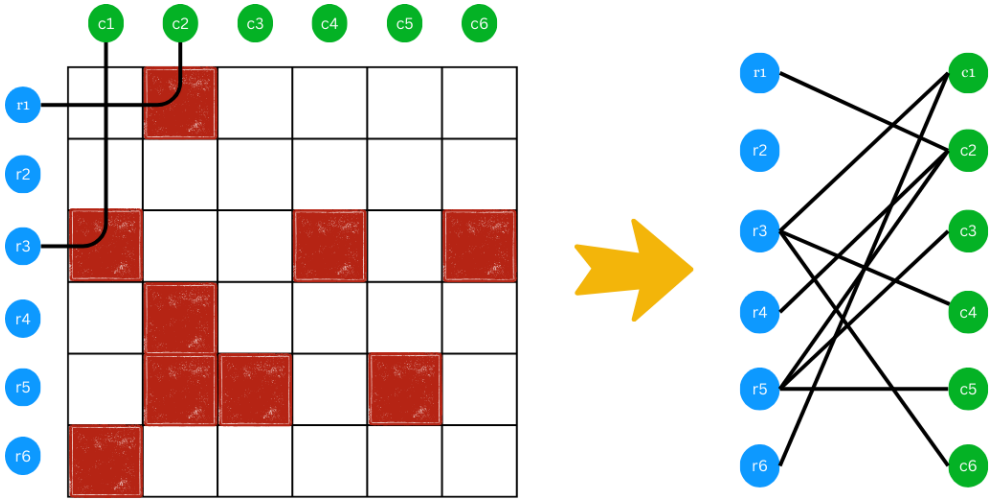


Figure 4: Transforming our game board to the corresponding bipartite graph

### From Grid to Graph: Connecting the Game to a Bipartite Structure

To translate the board to a graph:

- Assign one vertex for each row and one for each column.
- Place an edge between a row vertex $r_i$ and a column vertex $c_j$ whenever the cell at position $(i, j)$ is marked.

This construction yields a bipartite graph (Figure 4): row and column vertices form the two parts, and edges connect only between them. The graph precisely encodes the structure of your game board. A **matching** in this graph corresponds to a selection of non-conflicting red cells — that is, no two in the same row or column. A **vertex cover** corresponds to a set of rows and columns that together cover all red cells — that is, a line-cover.

### Step 1: Modeling Matching and Vertex Cover via Integer Programming

Let $G = (X \cup Y, E)$ be a bipartite graph with $m$ vertices and $n$ edges, where $X$ and $Y$ are the two parts of the bipartition. Label the vertices $v_1, \ldots, v_m$ and the edges $e_1, \ldots, e_n$. Let $A \in \mathbb{R}^{m \times n}$ be the **incidence matrix** of $G$ so that each row corresponds to a vertex and each column to an edge.

**Ph4-Task1 (6 points).** Define the maximum matching problem as an integer program. Introduce binary variables $x_j \in \{0, 1\}$ indicating whether edge $e_j$ is included in the matching. For each vertex $v_i$, write a constraint that ensures it is matched to at most one edge. Explain how these constraints enforce the definition of a matching — a set of edges with no shared vertices. Then, observe why the integrality condition $x_j \in \{0, 1\}$ can be relaxed to $x_j \in \mathbb{Z}$ without changing the solution.

**Ph4-Task2 (6 points).** Define the minimum vertex cover problem as an integer program. Introduce binary variables $y_i \in \{0, 1\}$ indicating whether vertex $v_i$ is included in the cover. For each edge $e_j$, write a constraint that ensures at least one of its endpoints is included. Explain how these constraints enforce the definition of a vertex cover — a set of vertices touching every edge. Then, observe why the integrality condition $y_i \in \{0, 1\}$ can be relaxed to $y_i \in \mathbb{Z}$ without changing the solution.

### Step 2: Relaxation and Duality

**Ph4-Task3 (5 points).** Drop the integrality constraints from both integer programs. Show that the resulting linear programs are dual to each other. By strong duality, conclude that their optimal values are equal.

### Step 3: Total Unimodularity and Integrality

We now aim to show that dropping the integrality constraints from the two integer programs does not change their optimal values. We will prove this using the concept of **total unimodularity**.

**Definition (Total Unimodularity).** A matrix $A$ is called **totally unimodular** if every square submatrix of $A$ (obtained by deleting some rows and columns) has determinant in $\{0, \pm 1\}$. In particular, all entries of $A$ must be in $\{0, \pm 1\}$.

Such matrices are of special interest because an integer program with a totally unimodular constraint matrix and an integral right-hand side can be solved by simply solving its LP relaxation. You will now establish this in a sequence of tasks.

**Ph4-Task4 (5 points).** Let $A$ be a totally unimodular matrix, and let $\bar{A}$ be the matrix obtained by appending a unit vector $e_i$ as a new last column. Prove that $\bar{A}$ is also totally unimodular.

**Ph4-Task5 (9 points).** Consider the following linear program with $n$ variables and $m$ inequalities:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{maximize}} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned}$$

where $\mathbf{b} \in \mathbb{Z}^m$, and $A \in \mathbb{R}^{m \times n}$ is totally unimodular. Prove that if this LP has an optimal solution, then it also has an optimal solution $\mathbf{x}^* \in \mathbb{Z}^n$. *Hint:* Any linear program (if solvable) admits an optimal **basic feasible solution**, which satisfies a subset of the constraints as equalities. These are found by solving a linear system defined by a square submatrix of $A$. If $A$ is totally unimodular and $\mathbf{b} \in \mathbb{Z}^m$, then such systems have integer solutions since the inverse of the submatrix has integer entries — by Cramer's Rule.

**Ph4-Task6 (Bonus – 6 points).** *Optional — complete for bonus credit.* Let $G = (X \cup Y, E)$ be a bipartite graph. Prove that its incidence matrix $A$ is totally unimodular. *Hint:* Use induction and consider how columns interact in any square submatrix.

**Ph4-Task7 (5 points).** Use the previous results to prove König's Theorem by showing that both LP relaxations have integral optimal solutions and equal values by strong duality.

### Real-World Parallel: Monitoring Power Grids

To see how these ideas apply beyond a classroom game, consider the problem of monitoring power flows in an electric grid. Operators must install specialized sensors, called Phasor Measurement Units (PMUs), to observe electricity flow between different regions (generation zones and load centers). These devices provide high-precision measurements of voltage and current phasors, which makes them extremely crucial for detecting faults or disturbances, and uncovering anomalies that might indicate cyberattacks. Due to cost, PMUs cannot be installed everywhere — so the challenge is to place the minimum number of PMUs while still ensuring that all inter-area power flows are observable. This corresponds exactly to finding a minimum vertex cover on a bipartite graph. One set of nodes represents boundary buses in a generation region — these are the points where power exits the generation area and enters the broader network. The other set represents boundary buses in the transmission network — the points where power enters from the generation region. The edges between them represent physical transmission lines carrying electricity between the region and the transmission network.

Now, imagine a malicious attacker trying to disrupt the grid without being detected. The attacker's goal is to find the maximum set of transmission lines they can exploit such that no endpoint is being monitored. This corresponds to a maximum matching in the unmonitored graph — the same concept that defined optimal

student strategies in the coin game. In both cases, the game is the same: one side covers, and the other matches. König's Theorem tells us that the smallest number of PMUs needed to monitor all flows is equal to the largest number of flows that could be manipulated if no PMUs were placed.

## Phase 5: Implementing a Simplex-Based Solver

You've developed strategies and proved key results — now it's time to implement them. In this phase, you'll write a Python LP solver to find a maximum matching and minimum vertex cover in a bipartite graph like the one from the game.

**Important constraint:** You must implement the **Simplex algorithm** from scratch. Use of any LP solvers or optimization libraries (e.g., `scipy.optimize`, `cvxpy`) is not allowed.

**Ph5-Task (35 points).**

- Implement a simplex solver in Python that solves a standard form linear program (LP). The solver should use the two-phase simplex algorithm, consisting of Phase I and Phase II (see the lecture notes for details). However, you are only required to implement Phase II. The Phase I implementation is already provided in the code skeleton `simplex_solver.py`, and your task is to complete the missing parts for Phase II in the skeleton. In Phase II, use the final basis and tableau obtained from Phase I as the initial basis and tableau. Then apply the tableau method to find an optimal solution. The implementation should also handle the case when the problem is unbounded (see lecture notes for details).

  For pivoting in the tableau method—that is, selecting which non-basic variable should enter the basis and which basic variable should leave—the simplex solver should allow the user to choose between Dantzig's rule and Bland's rule.

  *Remark:* For the pivot column, Dantzig's rule selects the column corresponding to the most negative reduced cost component, while Bland's rule selects the first column corresponding to a negative reduced cost component. For the pivot row, if multiple rows attain the minimum ratio, Dantzig's rule selects an arbitrary row among them, whereas Bland's rule selects the first such row. While Dantzig's rule may result in cycling if a degenerate basic feasible solution exists, Bland's rule guarantees termination of the algorithm in finite time.

- Formulate the maximum matching and minimum vertex cover problems as primal and dual LPs, respectively. You may refer back to the LP formulations obtained in Phase 4. Then, using your simplex solver, solve both the maximum matching and the minimum vertex cover problems (corresponding to the maximum pairing and minimum line-cover problems, respectively) for the incidence matrix of the board in Figure 1, as well as for a larger $100 \times 100$ board defined in the code skeleton `run_coin_game.py`.

  Assuming you have correctly implemented the simplex method in `simplex_solver.py`, you only need to run the code in `run_coin_game.py` to solve the problems and plot the results.

  Your report should include:

  (i) A plot of the board corresponding to the optimal solution;

  (ii) The corresponding optimal value,

  for each of the two boards.

  Try also changing the pivoting method from Dantzig's rule to Bland's rule. Is the solution different? Is the optimal value different?

*Good luck — and may your strategy be sharper than your opponent's!*