Algorithmic game theory (NDMI098)
Lecture notes

Martin Balko

October 20, 2024

Contents

1 Introduction

Contents of this chapter

1.1 Preface

These are notes to a lecture Algorithmic game theory (NDMI098) taught by M. Balko at Charles University in Prague. This text should serve as an introductory text on the game theory with emphasis on algorithmic perspective on this field. The notes cover the material presented in the course relatively concisely, although some parts might go beyond the usual scope of the course. The current version of the notes are still under construction, as the course is being held for the first time in the winter term 2018/2019, so it is possible that some parts might still change significantly.

Literature. There are several great sources on algorithmic game theory available and the current version of the notes is mostly based on these.

Perhaps the most comprehensive source on algorithmic theory is the following book edited by Nisan, Roughgarden, Tardos, and Vazirani [NRTV07]. This great book is a result of the effort of more than 40 of the top researchers in the area who have written chapters that go from the foundations to the state of the art, covering game-theory basics, algorithm mechanism design, quantifying the inefficiency of equilibria, and applications of game theory.

- Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani, editors. *Algorithmic game theory*. Cambridge University Press, Cambridge, 2007

A large portion of this text, especially parts about mechanism design, is based on the following book by Roughgarden [Rou16]. This book grew out of the Stanford University course on algorithmic game theory taught by Tim Roughgarden, and it gives a very nice and accessible introduction to many of the most important concepts in the field.

- Tim Roughgarden. *Twenty lectures on algorithmic game theory*. Cambridge University Press, Cambridge, 2016

A very good introductory text on game theory is a beautiful thin book by Leyton-Brown and Shoham [LBS08].

- Kevin Leyton-Brown and Yoav Shoham. *Essentials of game theory*, volume 3 of *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool Publishers, Williston, VT, 2008. A concise, multidisciplinary introduction

Some parts about the Minimax theorem are taken from an excellent book by Matoušek and Gärtner [MG06] about linear programming. This book can also serve as a very nice introduction to linear programming.

- Jiří Matoušek and Bernd Gärtner. *Understanding and Using Linear Programming*. Springer-Verlag New York, Inc., 2006

On exercises. At the end of most of the sections, the reader can find a collection of exercises. Solving them might help the reader in understanding the material covered. We classify the exercises according to difficulty. This classification is based on points listed next to the problem statements. Generally, the higher the number of points is, the harder the exercise. Finding a solution for one-point exercises should be quite easy, solving an exercise worth three points and more might require some bright idea or a bit more work.

A note for Czech students. Since the Czech terminology is sometimes slightly different than word-for-word translation, we use footnotes throughout these notes to illustrate Czech translation of some selected terms. We try to list more possibilities whenever the translation is not uniquely determined.

Finally, if you spot any mistake in these lecture notes or if you have suggestions of how to improve the exposition, please, feel free to send any comments to `balko@kam.mff.cuni.cz`. I would really appreciate it.

Acknowledgement. I would like to thank Ondřej Mička, Anton Dandarov, Jan Kuchařík, David Klement, Vladimír Drechsler, Viktoria Patapeika, Jan Kaifer, Jakub Komárek, Jakub Kubík, Jan Plot, Patrik Zavoral, and Tomáš Čížek for carefully reading the lecture notes and for their valuable comments.

1.2   Algorithmic game theory

Game theory is the study of formal models of strategic environments and strategic interaction between rational decision-makers. The start of modern game theory was the proof of the existence of mixed-strategy equilibria in two-person zero-sum games by John von Neumann. Game theory developed extensively in the 1950s by many scholars and found many applications in diverse areas such as economics and biology. Game theory has been recognized as an important tool in various fields, for example, more than ten game theorists have won the Nobel Prize in economics.

In these notes we focus on *algorithmic game theory*, a relatively new field that lies in the intersection of game theory and computer science. The main objective of algorithmic game theory is to design effective algorithms in strategic environments.

In the first part of these notes, we focus on algorithmic theory's earliest research goals— algorithms for computing equilibria (Nash, correlated, and several others). Informally, an equilibrium is a "steady state" of a system where no participant, assuming everything else stays the same, has anything to gain by changing only his own strategy. We also consider online learning algorithms and show that we can use them to efficiently approximate certain equilibria.

The second part of these notes is devoted to *mechanism design*, a subarea of game theory that deals with designing games toward desired objectives. The players have unknown and private utilities and the goal is to design rules so that strategic behavior by participants leads to a desirable outcome. This area, which is sometimes called reverse game theory, has broad applications ranging from economics and politics (markets, auctions, voting procedures) to networked-systems (internet interdomain routing, sponsored search auctions).

2 Finding Nash equilibria

Contents of this chapter

In this chapter, we introduce basic concepts in game theory. We state the most fundamental definitions and illustrate them on specific examples of games. We introduce some basic solution concepts such as *Nash equilibrium* and *Pareto optimality* and prove the famous *Nash's theorem* about the existence of Nash equilibria in every game with a finite number of players, each having a finite number of actions.

In the rest of this chapter, we deal with the problem of finding Nash equilibria efficiently. We first prove the *Minimax theorem*, showing that it is possible to do so in so-called *zero-sum games*, which are games of two players, where the gain of one player equals the loss of the other one. Then we focus on bimatrix games, present the *Lemke–Howson algorithm* for finding a Nash equilibrium in a bimatrix game, introduce the complexity class *PPAD*, and argue that this is the right class for studying the complexity of finding Nash equilibria in bimatrix games.

Since, as we will see, there is good evidence suggesting that finding Nash equilibria is a computationally difficult task, we relax the definition of a Nash equilibrium and consider some of its variants. Namely, we focus on *approximate Nash equilibria* and *correlated equilibria* and show that both these solution concepts can be found efficiently.

Afterward, we consider online learning algorithms and study simple learning algorithms that minimize so-called *external regret*. We show that such algorithms can be used to prove the Minimax theorem and also can be used in a procedure called *no-regret dynamics* that can find so-called *coarse correlated equilibria* that form a superset of correlated equilibria and are even easier to compute. Similarly, we present a version of this procedure for so-called *swap regret* that can be used to approximate correlated equilibria.

Finally, we conclude the chapter with a short description of *games in extensive form*, that is, games represented by trees and we discuss how to compute Nash equilibria there.

2.1 Games in normal form

The *normal form* of a game, also known as *strategic* or *matrix form*, is a representation of the game using a (high-dimensional) matrix. It is the most familiar representation of games and arguably the most fundamental, as most other representations of games can be reduced to it. A game written in this way includes all perceptible and conceivable strategies, and their corresponding payoffs, for each player.

Definition 2.1 (Normal-form game). *A (finite, n-player) normal-form game[1] is a triple (P, A, u), where*

- *P is a finite set of n players,*

- *$A = A_1 \times \cdots \times A_n$ is a set of action profiles, where A_i is a set of actions available to player i,*

- *and $u = (u_1, \ldots, u_n)$ is an n-tuple, where each $u_i \colon A \to \mathbb{R}$ is the utility function (or payoff function[2]) for player i.*

That is, each normal-form game $G = (P, A, u)$ can be represented by a real n-dimensional matrix $M = (M_a)_{a \in A}$, where $M_a = u(a)$. Knowing the utility function, all players simultaneously choose an action from the set of their available actions. The resulting *action profile* $a = (a_1, \ldots, a_n)$ is then evaluated using the utility function. The ith coordinate $u_i(a)$ of $u(a)$ denotes the gain (or loss) of player i on the action profile a. This value can be interpreted as a measure of the player's i level of happiness at state a.

For example, the normal form of the well-known game Rock-Paper-Scissors can be captured by the following two-dimensional matrix.

Example 2.2 (Rock-Paper-Scissors). *We have two players, that is, $P = \{1, 2\}$. Both players have the same set of actions $A_1 = \{Rock, Paper, Sccissors\} = A_2$ and the utility function assigns a value from $\{-1, 0, 1\}$ to each player, depending on the action chosen. For example, we have $u(Paper, Rock) = (1, -1)$, because player 1 wins by choosing Paper if player 2 chooses Rock. A normal form of this game is depicted in Table 2.1.*

	Rock	Paper	Scissors
Rock	(0,0)	(-1,1)	(1,-1)
Paper	(1,-1)	(0,0)	(-1,1)
Scissors	(-1,1)	(1,-1)	(0,0)

Table 2.1: A normal form of the game Rock-paper-scissors.

Example 2.3 (Chess). *To illustrate that the normal-form games are very general, we sketch how to represent chess as a normal-form game. Each action of player $i \in \{black, white\}$ is a list of all possible situations that can happen on the board together with the move player i would make in that situation. This is possible as the game of chess is finite. Now, if each player i chooses his action a_i, then we can simulate the whole game of chess the players would play and check what is the result $(u_{black}(a_{black}, a_{white}), u_{white}(a_{black}, a_{white}))$ of that game. Of course, the resulting matrix A is astronomically large and not practical to use.*

---

[1] Hra v normální formě.
[2] Výplatní funkce či užitková funkce.

Each player i in a normal-form game $G = (P, A, u)$ of n players can follow a certain *strategy*. A strategy is a prescription for how he chooses an action from A_i that he plays. One possible strategy is to always select a single action and play it. This is so-called *pure strategy*.

Definition 2.4 (Pure strategy)**.** *The set of* pure strategies[3] *of player i is the set A_i of available actions for i. A* pure-strategy profile[4] *is an n-tuple $(s_1, \ldots, s_n)$, where $s_i \in A_i$ for each player i.*

In other words, a pure-strategy profile is a choice of pure strategy for each player. We note that sometimes authors do not distinguish between pure strategies and actions in the literature.

Another type of strategy, developed largely by Von Neumann, is a *mixed strategy* where player i selects an action from A_i according to some probability distribution on A_i.

Definition 2.5 (Mixed strategy)**.** *The set S_i of* mixed strategies[5] *of player i is the set $\Pi(A_i)$, where $\Pi(X)$ is a set of all probability distributions on a set X. Similarly as in the case of pure strategies, a* mixed-strategy profile[6] *is an n-tuple $(s_1, \ldots, s_n)$, where $s_i \in S_i$ for each player i.*

That is, the set S_i contains infinitely many mixed strategies for player i. Note that a pure strategy is a degenerate case of a mixed strategy in which the action to be played is assigned probability 1. For a mixed strategy s_i of player i and action $a_i \in A_i$, we use $s_i(a_i)$ to denote the probability that the action a_i is played by player i under mixed strategy s_i.

The set $\{a_i \in A_i : s_i(a_i) > 0\}$ is called the *support*[7] of s_i. A mixed strategy s_i is *fully mixed* if the support of s_i is the entire set A_i, that is, if s_i assigns every action from A_i a positive probability. A *fully-mixed-strategy profile* is a strategy profile, where the strategy of every player is fully mixed.

With the notion of mixed strategies, the goal of each player in G is to maximize his *expected payoff*. The expected payoff of player i is defined as the expected value of u_i under the product distribution $\prod_{j=1}^{n} s_j$.

Definition 2.6 (Expected payoff of a mixed strategy)**.** *In a normal-form game $G = (P, A, u)$ of n players, the* expected payoff *(or* expected utility*)[8] for player i of the mixed-strategy profile $s = (s_1, \ldots, s_n)$ is*

$$u_i(s) = \sum_{a=(a_1, \ldots, a_n) \in A} u_i(a) \prod_{j=1}^{n} s_j(a_j).$$

That is, given the strategy profile s, we calculate the average of the payoffs of all outcomes $a \in A$, weighted by the probabilities of each outcome under s. The expected payoff is the unique extension of the payoff function u_i that is linear in the mixed strategy of each player. To capture this formally, we use a bit unfortunate but established notation s_{-i} to denote the strategy profile $s = (s_1, \ldots, s_n)$ without the strategy of player i. That is, $s_{-i} = (s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n)$. For a strategy $s_i' \in S_i$ of player i, we use $u_i(s_i'; s_{-i})$ to denote the number $u_i(s_1, \ldots, s_{i-1}, s_i', s_{i+1}, \ldots, s_n)$. Thus, in particular, $u_i(s_i; s_{-i}) = u_i(s)$. The linearity of the expected payoff then means

$$u_i(s) = \sum_{a_i \in A_i} s_i(a_i) u_i(a_i; s_{-i}) \tag{2.1}$$

for every player $i \in P$ and every mixed-strategy profile $s = (s_1, \ldots, s_n)$. We leave the proof of (2.1) to the reader; see Exercise 2.1.

Example 2.7 (Payoffs in the Rock-Paper-Scissors game)**.** *To illustrate these definitions, consider the pure strategy profile $s = (Paper, Scissors)$ in the Rock-Paper-Scissors game. This profile corresponds to the situation when the first player always chooses Paper, while the second player*

---

[3] Čistá strategie či ryzí strategie.
[4] Profil čistých strategií či čistý strategický profil.
[5] Smíšená strategie.
[6] Profil smíšených strategií či smíšený strategický profil.
[7] Doména smíšené strategie.
[8] Střední hodnota výplatní (či užitkové) funkce.

always selects Scissors. Obviously, player 1 always loses in this profile and thus $u_1(s) = -1$ and $u_2(s) = 1$.

Now, for $i \in \{1, 2\}$, let $s_i \in S_i = \Pi(A_i)$ be the mixed strategy, where player i chooses each action from A_i independently at random with probability $1/3$. Then each player wins with probability $1/3$ in the mixed-strategy profile (s_1, s_2) and neither player can increase his expected payoff, which is 0, via a unilateral deviation, as, for example,

$$u_1(s) = 1 \cdot \left(\frac{1}{3} \cdot \frac{1}{3} + \frac{1}{3} \cdot \frac{1}{3} + \frac{1}{3} \cdot \frac{1}{3} \right) + (-1) \cdot \left(\frac{1}{3} \cdot \frac{1}{3} + \frac{1}{3} \cdot \frac{1}{3} + \frac{1}{3} \cdot \frac{1}{3} \right)$$
$$+ 0 \cdot \left(\frac{1}{3} \cdot \frac{1}{3} + \frac{1}{3} \cdot \frac{1}{3} + \frac{1}{3} \cdot \frac{1}{3} \right) = 0.$$

2.1.1 Examples of normal-form games

Here, we provide more examples of normal-form games and list some possible types of such games. Several of these examples are used later to help understand further definitions and concepts in subsequent sections. We focus here only on normal-form games with two players, that is, we will have $P = \{1, 2\}$. These games are called *bimatrix games*[9], as each such game can be represented with a two-dimensional matrix, that is, with a table. In the examples below, player 1 is the "row player" while player 2 corresponds to the "column player".

The following game called *Prisoner's dilemma* and illustrated in Example 2.8, is a standard example in game theory, which shows that two rational individuals might not cooperate, even if it appears that it is in their best interests to do so.

Example 2.8 (Prisoner's dilemma). *Two prisoners, 1 and 2, are being held in solitary confinement and cannot communicate with each other. Each one of them is given an opportunity to either betray the other prisoner by testifying that the other committed the crime, or to cooperate with the other by remaining silent. If they both testify, each of them serves two years in prison. If 1 testifies but 2 remains silent, 1 will be set free and 2 will serve three years in prison (and vice versa). If they both remain silent, both of them will only serve 1 year in prison. The matrix of this game is illustrated in Table 2.2.*

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

Table 2.2: A normal form of the game Prisoner's dilemma.

Even if it appears that the best interest of both prisoners is to cooperate together and remain silent, the only stable solution of this game is, quite paradoxically, when both prisoners testify. Otherwise, one of them can change his action to Testify and improve his payoff. In the Prisoner's dilemma example, the precise payoff numbers are not so important, the example works also in the more general setting illustrated in Table 2.3 as long the condition $a < b < c < d$ holds.

	Testify	Remain silent
Testify	(b,b)	(d,a)
Remain silent	(a,d)	(c,c)

Table 2.3: A normal form of the generalized game Prisoner's dilemma. Here, $a < b < c < d$.

---

[9] Maticové hry.

Definition 2.9 (Zero-sum game). *A bimatrix game $G = (P, A, u)$ is a* zero-sum game[10] *if for every action profile $a \in A$ we have $u_1(a) + u_2(a) = 0$.*

In other words, in each state of the game, the gain of the first player equals the loss of the second player and vice versa. Rock-Paper-Scissors is a classical example of a zero-sum game. An even simpler example of a zero-sum game is the game *Matching pennies*. In this game, both players have only two possible actions; see Example 2.10.

Example 2.10 (Matching pennies). *Each of the two players has a penny and chooses either Heads of Tails. Then both players simultaneously reveal the chosen side of their coin and compare them. If the pennies match, then player 1 wins and keeps both pennies. Otherwise, player 2 keeps both pennies.*

	Heads	Tails
Heads	(1,-1)	(-1,1)
Tails	(-1,1)	(1,-1)

Table 2.4: A normal form of the game Matching pennies.

We use this game later to illustrate the concepts of mixed strategies and Nash equilibrium.

Another classical example is the game *Battle of sexes*; see Example 2.11. This game displays elements of both coordination and competition.

Example 2.11 (Battle of sexes[11]). *A husband and wife wish to spend an evening together rather than separately, but cannot decide which event to attend. The husband wishes to go to a football match while the wife wants to go to the opera (or the other way around, if you prefer). If they cannot communicate, where should they go? The matrix form of this game is depicted in Table 2.5.*

	Football	Opera
Football	(2,1)	(0,0)
Opera	(0,0)	(1,2)

Table 2.5: A normal form of the game Battle of Sexes.

Our last example is the *Game of chicken*, also known as the *Hawk-dove game* or *Snowdrift game*.

Example 2.12 (Game of chicken[12]). *Drivers 1 and 2 drive towards each other on a collision course: one must swerve, or both die in the crash. However, if one driver swerves and the other does not, the one who swerves will be called a "chicken"; see Table 2.6 for the matrix form of this game.*

2.2 Basic solution concepts

In this section, we discuss how to reason in normal-form games. We already know what strategies are, now the objective is to maximize the player's expected payoff using appropriate strategies. In case of games with at least two players, the situation becomes more complicated, as the best strategy depends on the choices of the other players. Therefore, in game theory, we typically identify and study rules for predicting how a game will be played, called *solution*

---

[10] Hra s nulovým součtem.
[11] Souboj pohlaví či Manželský spor.
[12] Hra na kuře.

	Swerve	Straight
Swerve	(0,0)	(-1,1)
Straight	(1,-1)	(-10,-10)

Table 2.6: A normal form of the Game of chicken.

concepts[13]. Formally, a solution concept is a mapping from the set of all normal-form games that maps each game G to a set of strategy profiles of G. In this section, we describe the most commonly used solutions concepts: Nash's equilibrium and Pareto optimality.

2.2.1 Nash equilibrium

In a normal-form game $G = (P, A, u)$ of n players, assume first that player i in G knows what actions all the other players chose. Under this assumption, it is easy for player i to choose an action that maximizes his payoff. Such an action is called the *best response* of i to the actions chosen by other players.

Definition 2.13 (Best response). *The* best response[14] *of player i to the strategy profile s_{-i} is a mixed strategy s_i^* such that $u_i(s_i^*; s_{-i}) \geq u_i(s_i'; s_{-i})$ for each strategy $s_i' \in S_i$ of i.*

Informally, if a player could "cheat" and look at what strategies the other players have, he would like to switch his strategy to best response. For example, in the Matching pennies game (Example 2.10), the best response of player 1 to the action Heads played by player 2 is the action Heads, after which player 1 can keep the pennies. The best response of prisoner 1 to the other prisoner being silent in the Prisoner's dilemma game is to testify, since then prisoner 1 is set free.

Although in both these examples the best response was determined uniquely, in general this is not the case. In fact, there is always either exactly one best response or the number of best responses is infinite. The reason for this fact is that if s_i^1 and s_i^2 are two best responses of player i, then the linearity of expectation implies that the mixed strategy $ts_i^1 + (1 - t)s_i^2$ is also a best response of i for every $t \in [0, 1]$.

In general, the situation for player i is more complicated, as he does not know what strategies the other players selected, since players choose strategies simultaneously. We can, however, use the notion of best response to define one of the most important solutions concepts in game theory with diverse applications.

Definition 2.14 (Nash equilibrium). *For a normal-form game $G = (P, A, u)$ of n players, a* Nash equilibrium[15] *in G is a strategy profile $(s_1, \ldots, s_n)$ such that s_i is a best response of player i to s_{-i} for every $i \in P$.*

That is, Nash equilibrium is a stable solution concept, meaning that no player would like to change his strategy if he knew the strategies of the other players. The notion of Nash equilibria for normal-form games was introduced by Nash [Nas50] in 1950 and it is the most influential concept in game theory to this date. Von Neumann and Morgenstern [vNMKR44] introduced the concept of Nash equilibria even sooner, but their analysis was restricted to zero-sum games.

Since every Nash equilibrium is a strategy profile, it can be *pure*, *mixed*, or *fully mixed*. For example, there are no pure Nash equilibria in the Matching pennies game, as there is no pure strategy profile (s_1, s_2) such that s_1 is a best response to s_2 and vice versa. There is a unique mixed Nash equilibrium in this game: both players choose Heads or Tails with equal probability.

To give another example, the Battle of sexes game (Example 2.11) has two pure Nash equilibria: (Football, Football), where the husband (player 1) and the wife (player 2) go to the football

---

[13] Koncept řešení.
[14] Nejlepší odpověď.
[15] Nashova rovnováha či Nashovo ekvilibrium.

match, and (Opera, Opera), where they both go to opera. Are these two the only Nash equilibria of this game?

No, we show that there is a third, fully mixed, Nash equilibrium. Assume that the husband's mixed strategy s_1 is to choose the action Football at random with probability $p = s_1(\text{Football})$ and the action Opera with probability $1 - p = s_1(\text{Opera})$. Similarly, assume that the wife's mixed strategy s_2 is to choose Opera with probability $q = s_2(\text{Opera})$ and Football with probability $1 - q = s_2(\text{Football})$.

Proposition 2.15. *In the game Battle of sexes, the strategy profile* (s_1, s_2) *is a Nash equilibrium if and only if* $(p, q) = (1, 0)$, $(p, q) = (0, 1)$, *or* $(p, q) = (2/3, 2/3)$.

Proof. For the husband, the expected payoff of going to the football match under the mixed strategy profile $s = (s_1, s_2)$ is

$$u_1(\text{Football}; s_{-1}) = q u_1(\text{Football}, \text{Opera}) + (1 - q) u_1(\text{Football}, \text{Football}) = q \cdot 0 + (1 - q) \cdot 2$$
$$= 2 - 2q$$

and the expected payoff of going to opera is

$$u_1(\text{Opera}; s_{-1}) = q u_1(\text{Opera}, \text{Opera}) + (1 - q) u_1(\text{Opera}, \text{Football}) = q \cdot 1 + (1 - q) \cdot 0 = q.$$

Thus, husband's expected payoff equals

$$u_1(s) = p \cdot u_1(\text{Football}; s_{-1}) + (1 - p) \cdot u_1(\text{Opera}; s_{-1}) = p(2 - 2q) + (1 - p)q = q - p(3q - 2).$$

The function $u_1(s)$ is strictly increasing in p if $u_1(\text{Football}; s_{-1}) > u_1(\text{Opera}; s_{-1})$. This occurs when $q < 2/3$ and in this case the husband's best response is $p = 1$ and the couple goes to football. If $q > 2/3$, then $u_1(s)$ is strictly decreasing in p and the husband's best response is $p = 0$, in which case the couple goes to opera. If $q = 2/3$, then $u_1(s)$ does not depend on p, and any $p \in [0, 1]$ is a best response for the husband. In this case, they both might go to either of the two events. By symmetry, $q = 0$ is a best response for the wife if $p > 2/3$ and $q = 1$ if $p < 2/3$. For $p = 2/3$, any $q \in [0, 1]$ is a best response for the wife.

Altogether, we see that the strategy profile (s_1, s_2) is a Nash equilibrium if and only if $(p, q) = (1, 0)$, $(p, q) = (0, 1)$, or $(p, q) = (2/3, 2/3)$. The first two strategy profiles are the pure Nash equilibria we have seen from the start. The third fully-mixed-strategy profile corresponds to the situation in which the players go to their preferred event more often than the other. $\square$

2.2.2 Nash's theorem

Now that we have introduced the notion of Nash equilibria and illustrated it on a few examples, a natural question is whether they always exist in every game. In 1950, Nash [Nas50] showed that, indeed, every normal-form game has a mixed Nash equilibrium.

Theorem 2.16 (Nash's theorem [Nas50, Nas51]). *Every normal-form game has a Nash equilibrium.*

This is perhaps the most influential result in game theory and it won Nash the Nobel Memorial Prize in Economic Sciences in 1994. Before Nash, Von Neumann and Morgenstern [vNMKR44] showed that a mixed-strategy Nash equilibrium exists for any zero-sum game with a finite set of actions. However, Nash's theorem guarantees the existence of at least one mixed Nash equilibrium in each normal-form game for any finite number of players. The notion of mixed strategies is crucial for this result, since, as we have seen in the game Matching pennies, some games might have no pure Nash equilibrium.

In the rest of this section, we show a proof of Theorem 2.16. The proof is essentially topological, as its main ingredient is a fixed-point theorem due to Brouwer [Bro11]. Nash's original proof [Nas50] uses Kakutani's Fixed Point Theorem [Kak41]. Here, we follow the proof from [Nas51]. The proof itself is rather short, however, we need to state some definitions first.

For a positive integer d, a subset X of $\mathbb{R}^d$ is *compact* if X is closed and bounded. We say that a subset Y of $\mathbb{R}^d$ is *convex* if every line segment containing two points from Y is fully contained in Y. That is, for any two points x, y from Y, $tx + (1-t)y \in Y$ for every $t \in [0, 1]$. For n affinely independent points $x_1, \ldots, x_n$ from $\mathbb{R}^d$, an $(n-1)$-*simplex* $\Delta_n = \Delta_n(x_1, \ldots, x_n)$ with the vertex set $\{x_1, \ldots, x_n\}$ is the set of *convex combinations* of the points $x_1, \ldots, x_n$. That is,

$$\Delta_n = \left\{ \sum_{i=1}^{n} t_i x_i \colon t_i \in [0, 1], \sum_{i=1}^{n} t_i = 1 \right\}.$$

It is not difficult to observe that each simplex is a compact convex set in $\mathbb{R}^d$.

The following theorem is a famous result, called *Brouwer's Fixed Point Theorem*, which has been used across numerous fields of mathematics. It is a fundamental result in the topology of Euclidean spaces and, as we will see, it also plays a central role in game theory. Here, we state its extended version for compact and convex sets.

Theorem 2.17 (Brouwer's Fixed Point Theorem). *For a positive integer d, let K be a non-empty compact convex set in $\mathbb{R}^d$. Let $f\colon K \to K$ be a continuous mapping. Then, there exists a* fixed point $x_0 \in K$ *for f, that is, $f(x_0) = x_0$.*

The proof of this result is far from being trivial and we do not state it here.

We also use the following simple lemma, which states that the Cartesian product of compact sets in $\mathbb{R}^d$ is compact and which is a special case of Tychonoff's theorem. Again, we do not state the proof here, although it is not difficult and we leave it as an exercise.

Lemma 2.18. *For positive integers n and $d_1, \ldots, d_n$, let $K_1, \ldots, K_n$ be compact sets, each K_i lying in $\mathbb{R}^{d_i}$. Then, the Cartesian product $K_1 \times \cdots \times K_n$ is a compact set in $\mathbb{R}^{d_1 + \cdots + d_n}$.*

Now, we are ready to present the proof of Nash's theorem about the existence of mixed Nash equilibria in every normal-form game.

Proof of Theorem 2.16. Let $G = (P, A, u)$ be a normal-form game of n players. For each player $i \in P$, let S_i be the of mixed strategies for player i. We will use Brouwer's Fixed Point theorem for a suitable continuous mapping f defined on the set $S = S_1 \times \cdots \times S_n$ of all mixed strategies for G so that the obtained fixed point for f in S is the Nash equilibrium for G. In order to do so, we first need to verify that S is compact and convex.

Without loss of generality, for each $i \in P$, we regard the actions from the set A_i as vertices of an $(|A_i| - 1)$-simplex in $\mathbb{R}^{|A_i|}$. Then each S_i is a simplex in $\mathbb{R}^{|A_i|}$ with the vertex set A_i of possible actions for i. In particular, each set S_i is a compact and convex set in $\mathbb{R}^{|A_i|}$. Thus, by Lemma 2.18, the set S is a compact set in $\mathbb{R}^m$, where $m = |A_1| + \cdots + |A_n|$. It is easy to show that S is also convex, as for any two mixed-strategy profiles $s = (s_1, \ldots, s_n), s' = (s'_1, \ldots, s'_n) \in S$ and $t \in [0, 1]$, the point $ts + (1-t)s' = (ts_1 + (1-t)s'_1, \ldots, ts_n + (1-t)s'_n)$ is also a mixed-strategy profile for G and thus lies in S.

It remains to define the appropriate continuous mapping $f\colon S \to S$ for which the fixed points are Nash equilibria. Let $s \in S$ be a given mixed strategy profile for G. For every player $i \in P$ and every action $a_i \in A_i$, we first define a mapping $\varphi_{i,a_i}\colon S \to \mathbb{R}$ by setting

$$\varphi_{i,a_i}(s) = \max\{0, u_i(a_i; s_{-i}) - u_i(s)\}.$$

Using the definition of u_i, one can show that this mapping is continuous. Its image is positive if and only if player i can improve his payoff by selecting the action a_i instead of using the strategy s_i in the strategy profile s. Using the functions φ_{i,a_i}, we define a new strategy profile $s' \in S$ by setting

$$s'_i(a_i) = \frac{s_i(a_i) + \varphi_{i,a_i}(s)}{\sum_{b_i \in A_i} (s_i(b_i) + \varphi_{i,b_i}(s))} = \frac{s_i(a_i) + \varphi_{i,a_i}(s)}{1 + \sum_{b_i \in A_i} \varphi_{i,b_i}(s)} \tag{2.2}$$

for all $i \in P$ and $a_i \in A_i$. Observe that each $s'_i(a_i)$ lies in $[0, 1]$ and $\sum_{a_i \in A_i} s'_i(a_i) = 1$. Thus, s' is indeed a strategy profile for G. Intuitively, s' is a strategy profile obtained from s by increasing

probabilities at actions that are better responses to s. Finally, we define the function f by setting $f(s) = s'$.

The function f is continuous, since the functions φ_{i,a_i} are continuous. Thus, Theorem 2.17 implies that there is a strategy profile in S that is a fixed point for f. To finish the proof, we show that a strategy profile $s \in S$ is a Nash equilibrium of G if and only if s is a fixed point for f.

First, if s is a Nash equilibrium for G, then all functions φ_{i,a_i} are constant zero functions, and thus $f(s) = s$, so s is a fixed point for f. For the other direction, assume that a strategy profile $s = (s_1, \ldots, s_n) \in S$ is a fixed point for f. Consider an arbitrary player $i \in P$. There is an action $a'_i \in A_i$ from the support of s_i such that $u_i(a'_i; s_{-i}) \leq u_i(s)$, as otherwise $u_i(s) < \sum_{a_i \in A_i} s_i(a_i)u_i(a_i; s_{-i})$, which is impossible by (2.1), that is, by the linearity of the expected payoff. Since $u_i(a'_i; s_{-i}) \leq u_i(s)$, we have $\varphi_{i,a'_i}(s) = 0$ from the definition of φ_{i,a'_i}. Plugging this into (2.2), we obtain

$$s'_i(a'_i) = \frac{s_i(a'_i)}{1 + \sum_{b_i \in A_i} \varphi_{i,b_i}(s)}.$$

Since s is a fixed point for f, we get $s'_i(a'_i) = s_i(a'_i)$ and, since a'_i is in the support of s_i, we have $s_i(a'_i) > 0$. Thus, the denominator in the above expression is 1. This means that $\varphi_{i,b_i}(s) = 0$ for every $b_i \in A_i$, and thus player i cannot improve his expected payoff by moving to a pure strategy. In other words, $u_i(s) \geq u_i(b_i; s_{-i})$ for every $b_i \in A_i$. Combining this estimate with (2.1), we obtain

$$u_i(s''_i; s_{-i}) = \sum_{b_i \in A_i} s''_i(b_i)u_i(b_i; s_{-i}) \leq \sum_{b_i \in A_i} s''_i(b_i)u_i(s) = u_i(s)$$

for every mixed-strategy $s''_i \in S_i$. Therefore s_i is a best response of player i to s_{-i} and s is a mixed Nash equilibrium of G. □

Note that the proof is not constructive, it just says that a Nash equilibrium exists, but does not tell us how to find one. The problem of finding Nash equilibria of a given game efficiently is discussed in the rest of this chapter.

Nash's theorem is stated for normal-form games, so only for games with a finite number of players, where each player has a finite number of actions. Both these assumptions are necessary, as games with an infinite number of players or with a finite number of players who have access to an infinite number of actions may not have Nash equilibria. A simple example of such a game is a two-player game, where each player selects a positive integer and a player who selected larger number gets payoff 1 while the other gets 0. In case of a tie, both players get 0. An example with infinite number of players, each with a finite number of actions, is given by the so-called *Banach-Mazur game*. In this game $G = (P, A, u)$, we have $P = \mathbb{N}$ and each player i has $A_i = \{0, 1\}$. The payoffs u are defined so that if an action profile a from A is eventually constant, then all players receive a payoff of 1. If a is not eventually constant, then all players receive a payoff of 0.

2.2.3 Pareto optimality

Another notion of optimality in normal-form games concerns the view of an outside observer of the game. Which outcomes of the game should the observer consider better than others? In general, it is difficult to answer this question, since we cannot say that the interests of some players are more important than the interests of other players; consider the game Battle of sexes from Example 2.11 as an illustration. One solution concept that is used to capture the notion of optimality from the view of an outside observer is *Pareto optimality* (or *Pareto efficiency*) in which a state is considered to be optimal if no player can be made better off without making any other player worse off.

A strategy profile s in a normal-form game $G = (P, A, u)$ *Pareto dominates*[16] strategy profile s', written $s' \prec s$, if, for every player $i \in P$, $u_i(s) \geq u_i(s')$, and there exists a player $j \in P$ such that $u_j(s) > u_j(s')$. This relation $\prec$ between strategy profiles is a partial ordering of the set

---

[16] Pareto dominuje či dominuje podle Pareta.

S of all strategy profiles of G. The outcomes of G that are considered best are the maximal elements of S in $\prec$.

Definition 2.19. *A strategy profile $s \in S$ is* Pareto optimal[17] *if there does not exist another strategy profile $s' \in S$ that Pareto dominates s.*

The concept is named after an Italian engineer and economist Vilfredo Pareto, who used it in his studies of economic efficiency and income distribution. Note that there is at least one Pareto optimal strategy profile in every normal-form game, but it might not be unique. For example, in zero-sum games, all strategy profiles are Pareto optimal. If there is more than one Pareto optimal profile, then such profiles are pairwise incomparable in $\prec$.

The example with zero-sum games shows that a Pareto optimal strategy profile might not be a Nash equilibrium. In the other direction, the Prisoner's dilemma game from Example 2.8 shows that Nash equilibria might not be Pareto optimal. We already mentioned that there is a unique Nash equilibrium in this game where both prisoners testify; the reader might try to prove this rigorously in Exercise 2.2. However, this strategy profile is not Pareto optimal, as each prisoner can remain silent instead, which does not make the other prisoner worse off. In fact, both prisoners can get a better payoff if they both remain silent. This example shows that sometimes a Nash equilibrium might not seem optimal to the outside observer of the game.

2.2.4   Exercises

Exercise 2.1. *Verify that the expected payoff of a mixed strategy in a normal-form game $G = (P, A, u)$ of n players is linear. That is, prove that $u_i(s) = \sum_{a_i \in A_i} s_i(a_i) u_i(a_i; s_{-i})$ for every player $i \in P$ and every mixed-strategy profile $s = (s_1, \ldots, s_n)$.* **[1]**

Exercise 2.2. *Compute mixed Nash equilibria in the following games:*

  (a) *Prisoner's dilemma (Example 2.8),* **[1]**

  (b) *Rock-Paper-Scissors (Example 2.2),* **[2]**

  (c) *Game of chicken (Example 2.12),* **[2]**

and formally show that no other Nash equilibria exist in these games.

Exercise 2.3. *Watch the scene from A Beautiful Mind where the John Nash character played by Russell Crowe explains what a Nash equilibrium is. Assume that the scenario described is modeled as a game with four players (the men), each with the same five actions (the women). Explain why the solution proposed in the movie is not a Nash equilibrium.* **[1]**

Exercise 2.4. *Consider the following game of $n \geq 2$ players. Every player selects, independently, a number from $\{1, \ldots, 1000\}$. The goal of each player is to have his number closest to half of the average of all the selected numbers.*

We define two variants of this game, depending on the tie-breaking rule. In the first rule, all players that are closest to the half of the average split evenly the payoff of 1. In the second tie-breaking rule, each player that is closest to the half of the average receives a payoff of 1.

How would you play each of these games? Find all pure Nash equilibria of the game under

  (a) *the first tie-breaking rule,* **[2]**

  (b) *the second tie-breaking rule.* **[2]**

Exercise 2.5. *There are $n \geq 2$ people on the street who all notice an injured man. Each one of them has two possible actions, either helping the injured man or not. If nobody helps the man, everybody gets a payoff 0. If somebody helps, all get a payoff 1, but anybody who offered help has to subtract c from his payoff, where $0 < c < 1$. Find a symmetric Nash equilibrium of this game, that is, a mixed equilibrium where all players use the same strategy. What is the probability the man is helped at all? Is it good for the injured man to have more witnesses around?* **[3]**

---

[17] Pareto optimální či optimální podle Pareta.

Exercise 2.6 (Iterated dominance equilibrium)**.** *Let $G = (P, A, u)$ be a normal-form game of n players. For player i, we say that a strategy $s_i \in S_i$ is* strictly dominated *by a strategy $s_i' \in S_i$ if, for every $s_{-i} \in S_{-i}$, we have $u_i(s_i; s_{-i}) < u_i(s_i'; s_{-i})$. Consider the following iterated process that will help us find Nash equilibria in some games.*

Set $A_i^0 = A_i$ and $S_i^0 = S_i$ for every player $i \in P$. For $t \geq 1$ and $i \in P$, let A_i^t be the set of pure strategies from A_i^{t-1} that are not strictly dominated by a strategy from S_i^{t-1} and let S_i^t be the set of mixed strategies with support contained in A_i^t. Let T be the first step when the sets A_i^T and S_i^T are no longer shrinking for any $i \in P$. If each player $i \in P$ is left with one strategy $a_i \in A_i^T$, we call $a_1 \times \cdots \times a_n$ an iterated dominance equilibrium *of G.*

(a) *Show that every iterated dominance equilibrium is a Nash equilibrium.* **[2]**

(b) *Find an example of a two-person game in normal form game with a pure Nash equilibrium that is not iterated dominance equilibrium.* **[1]**

Exercise 2.7. *Use iterated elimination of strictly dominated strategies (introduced in Exercise 2.6) to find the unique Nash equilibrium in the following normal-form game of 2 players (see Table 2.7) by first reducing the game to a 2×2 game.*

	c_1	c_2	c_3	c_4
r_1	(5, 2)	(22, 4)	(4, 9)	(7, 6)
r_2	(16, 4)	(18, 5)	(1, 10)	(10, 2)
r_3	(15, 12)	(16, 9)	(18, 10)	(11, 3)
r_4	(9, 15)	(23, 9)	(11, 5)	(5, 13)

Table 2.7: The game from Exercise 2.7.

Exercise 2.8 (Dominant strategy equilibrium)**.** *Let $G = (P, A, u)$ be a normal-form game of n players. An action profile $a^* \in A$ is* dominant strategy equilibrium *in G if, for every player $i \in P$, we have $u_i(a_i^*; a_{-i}) \geq u_i(a_i; a_{-i})$ for every $a_i \in A_i$ and $a_{-i} \in A_{-i}$.*

(a) *Is every dominant strategy equilibrium in G a Nash equilibrium in G?* **[1]**

(b) *Does every dominant strategy equilibrium in G Pareto dominate all other strategy profiles?* **[1]**

(c) *Let s be a strategy profile that Pareto dominates all other strategy profiles. Does that mean that s is a dominant strategy equilibrium in G?* **[1]**

Exercise 2.9. *Consider the game in which two players choose non-negative integers of size at most 1000. Player 1 chooses an even integer, while player 2 chooses an odd integer. When the players announce their number, the player who chose the lower number wins the number he announced in dollars. Find all pure Nash equilibria of this game.* **[3]**

2.3 Nash equilibria in zero-sum games

By Nash's theorem (Theorem 2.16), we know that Nash equilibria exist in every normal-form game. This is a very reassuring fact since any game can thus reach a state in which no player has an incentive to change his strategy. This gives us a prediction about the rational strategic behavior of the players. Unfortunately, as we have already mentioned, the proof of Nash's theorem is not constructive and does not tell us how to find a Nash equilibrium. In the rest of Chapter 2, we deal with the problem of finding an efficient algorithm for finding a Nash equilibrium of a given game. We start by dealing with zero-sum games and show that for such games Nash equilibria can be found efficiently using linear programming and that, in fact, such an equilibrium "solves" the game.

Let $G = (P, A, u)$ be a zero-sum game. We recall that this means that there are two players 1 and 2 and every action profile $a \in A$ satisfies $u_1(a) + u_2(a) = 0$. Thus, whatever one player gains, the other one loses. Examples of such games include the Rock-Paper-Scissors game (Example 2.2) and the Matching pennies game (Example 2.10).

Let $A_1 = \{1, \ldots, m\}$ and $A_2 = \{1, \ldots, n\}$ be actions available for player 1 and 2, respectively. Since $u_1(a) = -u_2(a)$, the game G can be represented with an $m \times n$ payoff matrix $M = (m_{i,j})$ such that $m_{i,j} = u_1(i,j) = -u_2(i,j)$. For a mixed-strategy profile (s_1, s_2), we use x_i and y_j to denote the probabilities $s_1(i)$ and $s_2(j)$, respectively, for every $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$. That is, we represent the mixed strategies s_1 and s_2 with vectors $x = (x_1, \ldots, x_m)$ and $y = (y_1, \ldots, y_n)$, respectively, that have non-negative coordinates and satisfy $\sum_{i=1}^{m} x_i = 1$ and $\sum_{j=1}^{n} y_j = 1$. We call the vectors x and y the *mixed strategy vectors*[18] for s and we sometimes refer to x and y as strategies. So we might, for example, use (x, y) to denote the mixed strategy profile $s = (s_1, s_2)$. With this notation, the sets S_1 and S_2 of mixed strategies of players 1 and 2, respectively, are the simplices $\Delta(e_1, \ldots, e_m)$ and $\Delta(f_1, \ldots, f_n)$, respectively, where $e_i \in \mathbb{R}^m$ and $f_i \in \mathbb{R}^n$ are vectors with the ith coordinate 1 and all others zero.

Using this representation of G, the expected payoff of player 1 equals

$$u_1(s) = \sum_{a=(i,j)\in A} u_1(a)s_1(i)s_2(j) = \sum_{i=1}^{m}\sum_{j=1}^{n} m_{i,j}x_i y_j = x^\top M y.$$

For player 2, we obviously have $u_2(s) = -u_1(s)$. Thus, player's 2 best response to a strategy x of 1, is a vector $y \in S_2$ that minimizes $x^\top M y$. Similarly, player's 1 best response to a strategy y of 2 is $x \in S_1$ that maximizes $x^\top M y$. The best expected payoff of 2 that player 2 can achieve against player's 1 strategy x equals $\beta(x) = \min_{y \in S_2} x^\top M y$. Analogously, for player 1, the best expected payoff to player's 2 strategy y is $\alpha(y) = \max_{x \in S_1} x^\top M y$. Note that we used the fact that the simplices S_1 and S_2 are compact and thus the values $\beta(x)$ and $\alpha(y)$ are well-defined. With this notion, a strategy profile (x, y) is a Nash equilibrium if and only if it satisfies $\beta(x) = x^\top M y = \alpha(y)$.

Assume player 1 expects player 2 to select a best response to every strategy x he can come up with. Player 1 then chooses a mixed strategy $\overline{x}$ from S_1 that maximizes his expected payoff under this, rather pessimistic, assumption. This strategy, which is called *worst-case optimal* for 1, satisfies $\beta(\overline{x}) = \max_{x \in S_1} \beta(x)$. Analogously, assuming player 2 is similarly pessimistic, the worst-case optimal strategy for 2 is a mixed strategy $\overline{y} \in S_2$ that satisfies $\alpha(\overline{y}) = \min_{y \in S_2} \alpha(y)$.

The following result shows that in order to achieve a Nash equilibrium in a zero-sum game, both players must select their worst-case optimal strategies.

Lemma 2.20. (a) *For all mixed strategies $x \in S_1$ and $y \in S_2$, we have $\beta(x) \leq x^\top M y \leq \alpha(y)$.*

 (b) *If a strategy profile (x^*, y^*) is a Nash equilibrium, then both strategies x^* and y^* are worst-case optimal for players 1 and 2, respectively.*

 (c) *Any mixed strategies $x^* \in S_1$ and $y^* \in S_2$ satisfying $\beta(x^*) = \alpha(y^*)$ form a Nash equilibrium (x^*, y^*).*

Proof. (a) This part follows immediately from the definitions of the functions β and α.

 (b) Part (a) implies that $\beta(x) \leq \alpha(y^*)$ for every $x \in S_1$. Since (x^*, y^*) is a Nash equilibrium, we have $\beta(x^*) = \alpha(y^*)$. This together with the previous fact implies $\beta(x) \leq \beta(x^*)$ for every $x \in S_1$. In other words, x^* is a worst-case optimal strategy for player 1. An analogous argument shows that y^* is a worst-case optimal strategy for player 2.

 (c) If $\beta(x^*) = \alpha(y^*)$, then part (a) implies $\beta(x^*) = (x^*)^\top M y^* = \alpha(y^*)$. Thus, (x^*, y^*) is a Nash equilibrium.

$\square$

---

[18] Vektory smíšených strategií.

2.3.1 The Minimax Theorem

The following result, called the *Minimax theorem*, was proved by Von Neumann in 1928 [vN28]. This theorem was a starting point of game theory Von Neumann is even quoted as saying "As far as I can see, there could be no theory of games ... without that theorem ... I thought there was nothing worth publishing until the Minimax Theorem was proved.".

Theorem 2.21 (The Minimax Theorem [vN28]). *For every zero-sum game, worst-case optimal strategies for both players exist and can be efficiently computed using linear programming. Moreover, there is a number v such that, for any worst-case optimal strategies x^* and y^* of players 1 and 2, respectively, the strategy profile (x^*, y^*) is a Nash equilibrium and $\beta(x^*) = (x^*)^\top M y^* = \alpha(y^*) = v$.*

In a certain sense, the Minimax theorem tells us everything about zero-sum games. In particular, it implies that every zero-sum game G has a Nash equilibrium. This also follows from Nash's theorem. However, the Minimax theorem historically predates Nash's result and also tells us much more: Nash equilibria can be found efficiently in zero-sum games. On top of that, there is a unique value $v = (x^*)^\top M(y^*)$ of the payoff that is attained in any Nash equilibrium (x^*, y^*). This value v is called the *value of the game G*.

Informally, the Minimax theorem tells us that there are no secrets involved in zero-sum games. Both players declaring their mixed strategies in advance changes nothing. Each player i can choose a worst-case optimal strategy and his expected payoff is always at least v, no matter which strategy the other player chooses. On the other hand, if the other player chooses his worst-case optimal strategy, then player i cannot hope for an expected payoff larger than v.

To explain the name "Minimax Theorem", note that if we write out the definitions of the functions β and α and the definitions of the worst-case optimal strategies x^* and y^*, then the equality $\beta(x^*) = v = \alpha(y^*)$ becomes

$$\max_{x \in S_1} \min_{y \in S_2} x^\top M y = v = \min_{y \in S_2} \max_{x \in S_1} x^\top M y.$$

Before we proceed with the proof of the Minimax theorem we mention some basic definitions and results from the theory of linear programming. In particular, we state the Duality theorem, which is crucial in the proof. We barely touch the surface of the theory about linear programming and an interested reader might find more in the literature; a nice introduction to linear programming is a book by Matoušek and Gärtner [MG06].

A *linear program* is an optimization problem with a linear objective function and linear constraints. Every linear program can be expressed in the following *canonical form P*: given vectors $c \in \mathbb{R}^m$ and $b \in \mathbb{R}^n$ and an $n \times m$ real matrix A, we want to maximize $c^\top x$ subject to the constraints $Ax \le b$ and $x \ge \mathbf{0}$. The problem of solving a linear program can be solved efficiently, that is, in polynomial time with respect to the size of the problem. In practice, the so-called *simplex algorithm* is quite efficient and can be guaranteed to find the global optimum if certain precautions against cycling are taken. It has been proved to solve "random" instances in a cubic number of steps, which is similar to its behavior on practical problems. Unfortunately, the worst-case behavior of the simplex algorithm is rather poor, as there are families of linear programming problems for which the simplex method takes an exponential number of steps. However, there are algorithms for solving linear programs even in the worst case, the first such algorithm found was so-called *ellipsoid algorithm*.

In the context of duality, the linear program P is called the *primal linear program*. When solving P, we are trying to find a linear combination of the n inequalities of the system $Ax \le b$ with some coefficients $y_1, \ldots, y_n \ge 0$ so that the resulting inequality has the jth coefficient at least c_j for every $j \in \{1, \ldots, m\}$ and the right hand is as small as possible. This leads to so-called *dual linear program D*: given vectors $c \in \mathbb{R}^m$ and $b \in \mathbb{R}^n$ and an $n \times m$ real matrix A, we want to minimize $b^\top y$ subject to constraints $A^\top y \ge c$ and $y \ge \mathbf{0}$.

Theorem 2.22 (The Duality Theorem). *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.*

The Duality Theorem is arguably the most important result in the theory of linear programs. There are several ways how to prove the Duality Theorem, for example, it is possible to derive it from the correctness of the simplex method or use the so-called *Farkas lemma*. We do not show the proof here, instead, we refer the reader to [MG06], where both variants of the proof are presented.

The Duality theorem holds for general linear programs, where some of the constraints contain equalities or opposite inequalities, not only for those in the canonical form. However, we need to construct the dual program in an appropriate way. Here, we state a general recipe, taken from [MG06], how to construct a dual program from a general linear program; see Table 2.8. Of course, it is possible to rewrite a given linear program to its canonical form by introducing new variables and then apply the Duality theorem (Theorem 2.22), so the recipe from Table 2.8 might be considered as a simpler shortcut. For minimization linear programs, the dual is defined as the reverse operation (from the right column to the left). In particular, the dual of the dual is the original primal.

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

Table 2.8: A recipe for making dual programs [MG06].

Now, having the Duality Theorem in hand, we can finally proceed with the proof of the Minimax Theorem.

Proof of the Minimax theorem (Theorem 2.21). Let $x = (x_1, \ldots, x_m)$ be a *fixed* mixed strategy vector of player 1. The best response of player 2 to x is the optimal solution of the following linear program P with variables $y_1, \ldots, y_n$: minimize $x^\top M y$ subject to constraints $\sum_{j=1}^n y_j = 1$ and $y_1, \ldots, y_n \geq 0$. Thus, for a given fixed x we can compute the value $\beta(x)$ using linear programming.

However, in order to find his worst-case optimal strategy, and thus to maximize $\beta(x)$ over $x \in S_1$, player 1 cannot use linear programming directly, since $\beta(x) = \min_{y \in S_2} x^\top M y$ is not a linear function. First, we use the recipe for constructing dual programs from Table 2.8 (from right to left) to derive the dual D of the program P. The dual linear program D then has only a single variable x_0 and looks as follows: maximize x_0 subject to constraints $\mathbf{1}x_0 \leq M^\top x$. That is, the coordinates $x_1, \ldots, x_m$ of x are treated as fixed numbers, the only variable is x_0. By the duality Theorem (Theorem 2.22), the programs P and D have the same optimal value $\beta(x)$.

Now, consider the following linear program D' with variables $x_0, x_1, \ldots, x_m$: maximize x_0 subject to constraints $\mathbf{1}x_0 - M^\top x \leq \mathbf{0}$, $\sum_{i=1}^m x_i = 1$, and $x \geq \mathbf{0}$. This is a linear program, since constraints in D are linear in $x_1, \ldots, x_m$. Assume $(x_0^*, x_1^*, \ldots, x_m^*)$ is an optimal solution for D' and let $x^* = (x_1^*, \ldots, x_m^*)$. Then $x_0^* = \beta(x^*) = \max_{x \in S_1} \beta(x)$, since the coordinates $x_1, \ldots, x_m$ of x are considered as variables in D'. So x^* is a worst-case optimal strategy for player 1 and also together with x_0^* provides an optimal solution for D'.

By symmetry, applying the same reasoning for player 2, we obtain the following linear program P' with with variables $y_0, y_1, \ldots, y_n$: minimize y_0 subject to constraints $\mathbf{1}y_0 - My \geq \mathbf{0}$, $\sum_{j=1}^{n} y_j = 1$, and $y \geq \mathbf{0}$. Here, $y = (y_1, \ldots, y_n)$. Again, if $(y_0^*, y_1^*, \ldots, y_n^*)$ is an optimal solution of P' and $y^* = (y_1^*, \ldots, y_n^*)$, then $y_0^* = \alpha(y^*) = \min_{y \in S_2} \alpha(y)$. Similarly as before, y^* is a worst-case optimal strategy for player 2 and also together with y_0^* provides an optimal solution for P'.

As the final step, we note that the programs D' and P' are dual to each other, as the recipe from Table 2.8 shows. Thus, the programs D' and P' have the same optimal value $\beta(x^*) = x_0^* = y_0^* = \alpha(y^*)$ and we have proved that both players can find their worst-case optimal strategies x^* and y^* using linear programming. By part (c) of Lemma 2.20, (x^*, y^*) is a Nash equilibrium. $\qquad\square$

Since the problem of finding a solution of a given linear program can be solved in polynomial time, the Minimax Theorem implies that finding Nash's equilibria in zero-sum games can also be computed in polynomial time. The Minimax Theorem was originally proved by Von Neumann [vN28] in the 1920s using Brouwer's Fixed Point Theorem. Later, in the 1940s, Von Neumann found a different proof that was based on the duality of linear programming [vNMKR44]. Nowadays, there are proofs of the Minimax theorem that are based on so-called regret-minimization algorithms and that avoid even the use of linear programming. One such proof is presented in Subsection 2.6.3.

2.4 Nash equilibria in bimatrix games

Now we know that Nash equilibria can be efficiently computed in zero-sum games of two players. This statement is captured by the Minimax Theorem (Theorem 2.21), whose proof was based on linear programming. Here, we focus on more general case of *bimatrix games*, that is, normal-form games of two players, and we show that, unlike the case of zero-sum games, one cannot hope for significantly more positive results.

We restrict ourselves to games with only two players, as such games can be represented with matrices and studied using polyhedra. We discuss the representation of bimatrix games with polyhedra in Subsection 2.4.4. Then, in Subsection 2.4.5, we use such representations and we present *Lemke–Howson algorithm*, the best known combinatorial algorithm for finding equilibria in games of two players. This algorithm, in particular, yields an alternative proof of Nash's Theorem (Theorem 2.16). We then enclose this section by showing in Subsection 2.4.6 that the problem of finding Nash's equilibria is, in some sense, difficult. Similarly as in the theory of NP-completeness, this is done by introducing a suitable complexity class. Here, this happens to be the class *PPAD* ("Polynomial Parity Arguments on Directed graphs") that contains several natural and well-known problems and we show that the problem of finding Nash's equilibria is PPAD-complete, meaning that this problem is at least as difficult as any problem in PPAD. The complexity class PPAD is a class of problems that are believed to be hard, but obtaining PPAD-completeness is weaker evidence of intractability than obtaining NP-completeness, as PPAD is contained in NP.

2.4.1 Best response condition

Before considering two-player games, we prove the following very useful observation, which holds for an arbitrary normal-form game.

Observation 2.23 (Best response condition). *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}.*

Proof. First, assume every pure strategy a_i in the support $\mathrm{Supp}(s_i)$ of s_i is a best response of player i to s_{-i}, that is, $u_i(a_i; s_{-i}) \geq u_i(s_i'; s_{-i})$ for every $s_i' \in S_i$. Then, for every $s_i' \in S_i$, the linearity of u_i implies

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

where the last equality follows from $\sum_{a_i \in \mathrm{Supp}(s_i)} s_i(a_i) = 1$. Thus, s_i is a best response of i to s_{-i}.

For the second part, assume s_i is a best response of i to s_{-i}. Suppose for contradiction there is $\overline{a}_i \in \mathrm{Supp}(s_i)$ that is not a best response of i to s_{-i}. Then $s_i(\overline{a}_i) > 0$ and there exists $s_i' \in S_i$ with $u_i(\overline{a}_i; s_{-i}) < u_i(s_i'; s_{-i})$. Since s_i is a best response of i to s_{-i}, we obtain $s_i(\overline{a}_i) < 1$ and, by the linearity of u_i, there is $\hat{a}_i \in \mathrm{Supp}(s_i)$ with $u_i(\hat{a}_i; s_{-i}) > u_i(\overline{a}_i; s_{-i})$. We define a new mixed strategy s_i^* of i that is obtained from s_i by setting $s_i^*(\overline{a}_i)$ to zero, $s_i^*(\hat{a}_i)$ to $s_i(\hat{a}_i) + s_i(\overline{a}_i)$ and keeping $s_i^*(a_i) = s_i(a_i)$ otherwise. Clearly, $s_i^* \in S_i$. Then, by the linearity of u_i and using $s_i(\hat{a}_i) > 0$, we obtain

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

which contradicts the assumption that s_i is a best response of i to s_{-i}. $\qquad \square$

Thus, although it might seem at first glance that the problem of finding a Nash equilibrium in a normal-form game is a problem in continuous mathematics, Observation 2.23 reveals that the task is essentially combinatorial. As we will see later in Subsection 2.4.2, the hearth of the problem of finding Nash equilibria is in finding the right supports. Once we have the right supports for all players, the precise mixed strategies can be computed by solving a system of algebraic equations (which are linear in the case of two players).

2.4.2 Finding Nash equilibria by support enumeration

Let $G = (\{1, 2\}, A = A_1 \times A_2, u)$ be a bimatrix game and let A_1 and A_2 be the sets of actions of players 1 and 2, respectively. Without loss of generality, we assume that $A_1 = \{1, \ldots, m\}$ and $A_2 = \{1, \ldots, n\}$, as we will use the actions as indices of rows and columns in matrices. Later we assume that the sets A_1 and A_2 are disjoint, which can be arranged if we imagine that, say, $A_2 = \{m + 1, \ldots, m + n\}$.

Since there are only two players in G, the payoff functions u_1 and u_2 can be represented by $m \times n$ real matrices M and N, respectively. That is, $(M)_{i,j} = u_1(i, j)$ and $(N)_{i,j} = u_2(i, j)$ for every pair $(i, j) \in A = A_1 \times A_2$ (the rows and columns of the matrices are indexed by A_1 and A_2, respectively). The expected payoffs of a mixed strategy profile s with mixed strategy vectors x and y are then given by

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

Also, note that in the case of only two players, the expected payoff of each player is a linear function in the probabilities of pure strategies of the other player.

We recall the Best response condition (Observation 2.23), which, in the case of two-player games, can be restated as follows. If x and y are mixed strategy vectors of players 1 and 2, respectively, then x is a best response to y if and only if for all $i \in A_1$,

$$x_i > 0 \implies (M)_i y = \max\{(M)_k y \colon k \in A_1\}. \tag{2.3}$$

Analogously, y is the best response to x if and only if for all $j \in A_2$,

$$y_j > 0 \implies (N^\top)_j x = \max\{(N^\top)_k x \colon k \in A_2\}. \tag{2.4}$$

In what follows, we will focus only on so-called nondegenerate games. The reason for this restriction will be made clear later. In some sense, most games are nondegenerate, as degeneracy involves a special relationship among the payoffs. Also, there is a standard method, so-called *perturbation*, how to deal with degenerate games.

Definition 2.24 (Nondegenerate bimatrix game). *A bimatrix game is* nondegenerate *if there are at most k pure best responses to every mixed strategy with support of size k.*

First, consider the following simple idea how to find all Nash equilibria in a nondegenerate game G. The Best response condition (Observation 2.23) says that for every player $i \in \{1, 2\}$, 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}. Thus, finding a Nash equilibrium is about finding the right supports. Once we identify the support for each player, the mixed strategies can be computed by solving a system of linear equations. If sets $I \subseteq A_1$ and $J \subseteq A_2$ are identified as supports, then we have $|I| + |J|$ variables x_i, $i \in I$, and y_j, $j \in J$, where $x_i = s_1(i)$ and $y_j = s_2(j)$. Let $x \in \mathbb{R}^m$ be a vector with coordinates x_i with $i \in I$ and $x_i = 0$ otherwise. Analogously, define the vector $y \in \mathbb{R}^n$ for J. The equations then say that variables for each player sum up to 1, that is, $\sum_{i \in I} x_i = 1$ and $\sum_{j \in J} y_j = 1$, and that the expected payoffs are equal and maximized at the support. That is,

$$(N^\top)_j x = \sum_{i \in I} (N^\top)_{j,i} x_i = v \quad \text{and} \quad (M)_i y = \sum_{j \in J} (M)_{i,j} y_j = u,$$

where $u = \max\{(M)_i y \colon i \in I\}$ and $v = \max\{(N^\top)_j x \colon j \in J\}$. Thus, we have a system of $|I| + |J| + 2$ variables $x_1, \ldots, x_{|I|}, y_1, \ldots, y_{|J|}, u, v$ and $|I| + |J| + 2$ linear equations, whose solution yields $|I| + |J| + 2$ real numbers. If the numbers in the solution are all non-negative and conditions (2.3) and (2.4) hold, then we have a Nash equilibrium. If such a solution exists, it is unique, since G is nondegenerate (otherwise, we can reduce the support); see Exercise 2.13. If our game is degenerate, then we can use the above equalities and capture the last two conditions with linear inequalities, obtaining a linear program for each pair of supports (that no longer need to be of the same size).

Brute-force algorithm for finding Nash equilibria: It follows immediately from the Best response condition (Observation 2.23) that if (s_1, s_2) is a Nash equilibrium in a nondegenerate bimatrix game, then the strategies s_1 and s_2 have supports of equal size. Thus, it suffices to go through all possible supports $I \subseteq A_1$ and $J \subseteq A_2$ of size k for $k \in \{1, \ldots, \min\{m, n\}\}$ and then verify whether the supports I and J yield a Nash equilibrium by solving the corresponding system of linear equations. Unfortunately, the running time of this simple procedure is typically quite large, that is, exponential in m or n. For $m = n$, the algorithm takes about 4^n steps. On the bright side, this algorithm finds all Nash equilibria of the game. An analogous algorithm can be used to find Nash equilibria in games of more than two players, but the resulting equations are no longer linear.

Algorithm 2.4.1: SUPPORT ENUMERATION(G)

$Input$: A non-degenerate bimatrix game G.
$Output$: All Nash equilibria of G.
for each $k \in \{1, \ldots, \min\{m, n\}\}$ and a pair of supports (I, J), each of size k

$\begin{cases} \text{solve the system of equations } \sum_{i \in I} (N^\top)_{j,i} x_i = v, \sum_{j \in J} (M)_{i,j} y_j = u \\ \text{for all } i \in I, j \in J \text{ and } \sum_{i \in I} x_i = 1, \sum_{j \in J} y_j = 1 \\ \text{if } x, y \geq \mathbf{0} \text{ and } u = \max\{(M)_i y \colon i \in A_1\}, v = \max\{(N^\top)_j x \colon j \in A_2\}, \\ \text{then return } (x, y) \text{ as a Nash equilibrium} \end{cases}$

We show later that the game G can be represented with labeled polyhedra that capture the underlying structure of the game. With this representation in hand, we show, in Subsection 2.4.5, a more sophisticated algorithm that finds a Nash equilibrium of G and is efficient in practice. To give this geometric description of G, we first recall basic definitions from geometry.

2.4.3 Preliminaries from geometry

For positive integers k and d, let $p_1, \ldots, p_k$ be points in $\mathbb{R}^d$. An *affine combination*[19] of the points $p_1, \ldots, p_k$ is an expression of the form $\sum_{i=1}^k \alpha_i p_i$ with $\alpha_i \in \mathbb{R}$ and $\sum_{i=1}^k \alpha_i = 1$. An

---

[19] Afinní kombinace.

affine combination $\sum_{i=1}^{k} \alpha_i = 1$ for which $\alpha_i \geq 0$ for each $i \in \{1, \ldots, k\}$ is called a *convex combination*[20] of $p_1, \ldots, p_k$. For example, the set of affine combinations of two points p_1 and p_2 in $\mathbb{R}^2$ is the line containing p_1 and p_2 while the set of convex combinations of p_1 and p_2 is the line segment with endpoints p_1 and p_2. The set of all convex combinations of a point set P is called the *convex hull*[21] of P and is denoted by $\operatorname{conv}(P)$. The points $p_1, \ldots, p_k$ are *affinely* or *convexly independent*[22] if no point p_i is affine or convex combination, respectively, of points from $\{p_1, \ldots, p_k\} \setminus \{p_i\}$. A subset P of $\mathbb{R}^d$ is *convex* if every convex combination of points from P lies in P. Observe that the intersection of convex sets is also convex. The *dimension* of P is the maximum number of affinely independent points from P minus 1.

A (closed) *halfspace*[23] in $\mathbb{R}^d$ is a set $\{x \in \mathbb{R}^d : v^\top x \leq w\}$ for some $v \in \mathbb{R}^d$ and $w \in \mathbb{R}$. Similarly, a *hyperplane*[24] in $\mathbb{R}^d$ is a set $\{x \in \mathbb{R}^d : v^\top x = w\}$. In particular, each hyperplane H in $\mathbb{R}^d$ divides the space $\mathbb{R}^d$ into two halfspaces that share H and are otherwise disjoint.

A (convex) *polyhedron*[25] P in $\mathbb{R}^d$ is an intersection of finitely many halfspaces in $\mathbb{R}^d$. In other words, $P = \{x \in \mathbb{R}^d : Vx \leq u\}$ for some real matrix V with d columns and $u = (u_1, \ldots, u_n) \in \mathbb{R}^n$, where n is the number of halfspaces determining P. Since halfspaces are convex sets, it follows that each polyhedron is convex. A bounded polyhedron is called *polytope*[26]; see Figure 2.1 for an illustration. A *face*[27] of a polyhedron P is either P itself or a subset of P of the form $P \cap H$, where H is a hyperplane such that P is contained in one of the closed halfspaces determined by H. The faces of P with dimension 0, 1, and $d-1$ are called *vertices, edges,* and *facets* of P, respectively. Using $(V)_i$ to denote the ith row of the matrix V, it can be shown that each face F of P can be expressed as $F = \cap_{i \in I} \{x \in P : (V)_i x = u_i\}$ for some set I of indices of the rows of V. The equalities $(V)_i x = u_i$ are then called the *binding* equalities for F. A polyhedron is *simple*[28] if none of its points belongs to more than d facets.
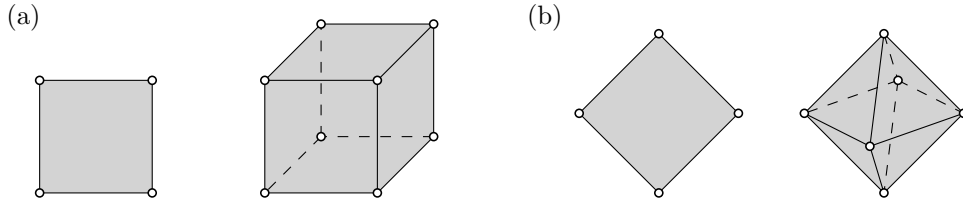
Figure 2.1: Examples of polytopes. (a) 2- and 3-dimensional cube (b) 2- and 3-dimensional crosspolytope.

2.4.4   Best response polytopes

We first define these labeled polyhedrons for general bimatrix games. Later we show that under certain assumptions on G, which can be easily achieved, the game G can be represented even with simple polytopes.

Definition 2.25 (Best response polyhedron[29].)**.** *The* best response polyhedron *for player* 1 *in* G *is a polyhedron* $\overline{P}$ *is defined as*

$$\overline{P} = \{(x, v) \in \mathbb{R}^m \times \mathbb{R} : x \geq \mathbf{0}, \mathbf{1}^\top x = 1, N^\top x \leq \mathbf{1}v\}.$$

Similarly, we define the best response polyhedron for player 2 *in* G *as*

$$\overline{Q} = \{(y, u) \in \mathbb{R}^n \times \mathbb{R} : y \geq \mathbf{0}, \mathbf{1}^\top y = 1, My \leq \mathbf{1}u\}.$$

[20]   Konvexní kombinace.
[21]   Konvexní obal.
[22]   Afinně či konvexně nezávislé.
[23]   Poloprostor.
[24]   Nadrovina.
[25]   Polyedr.
[26]   Polytop.
[27]   Stěna.
[28]   Jednoduchý mnohostěn.
[29]   Polyedr nejlepších odpovědí.
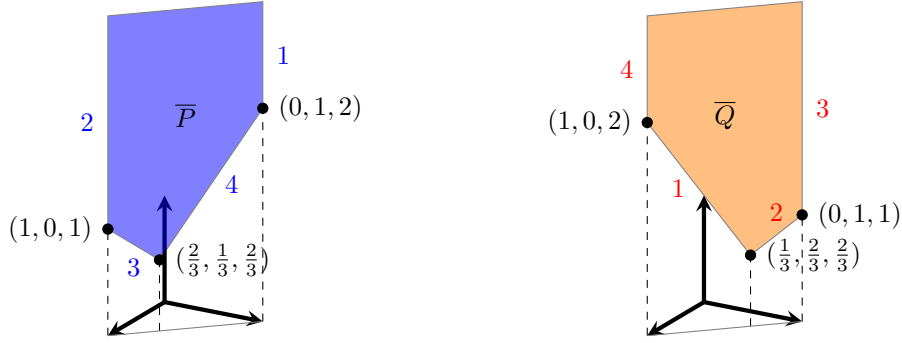
Figure 2.2: The best response polyhedra $\overline{P}$ and $\overline{Q}$ for the game Battle of Sexes. The labels of points in the facets are denoted by blue and red, respectively, numbers.

Let (x, v) and (y, u) be points from $\overline{P}$ and $\overline{Q}$, respectively. Let $s = (s_1, s_2)$ be a mixed-strategy profile and let x and y be mixed strategy vectors for s. Then the condition $My \le \mathbf{1}u$ says that the expected payoff $u_1(s)$ is at most u. This is because no matter which mixed strategy vector x player 1 selects, his payoff is always

$$u_1(s) = x^\top M y \le x^\top \mathbf{1} u = u \sum_{i \in A_1} x_i = u.$$

Analogously, the condition $N^\top x \le \mathbf{1}v$ says that the expected payoff $u_2(s)$ is at most v. Thus, for $i \in \{1, 2\}$, the points of the best response polyhedron for player i are the mixed strategies of player i together with the "upper envelope" of expected payoffs of the other player $3 - i$.

Now, we define the labels of points from $\overline{P}$ and from $\overline{Q}$. We say that a point (x, v) of $\overline{P}$ has *label* $i \in A_1 \cup A_2$ if either $i \in A_1$ and $x_i = 0$ or if $i \in A_2$ and $(N^\top)_i x = v$, where $(N^\top)_i$ is the ith row of $N^\top$. That is, if the corresponding inequality in the definition of $\overline{P}$ is binding. Similarly, a point (y, u) has a label $i \in A_1 \cup A_2$ if either $i \in A_1$ and $(M)_i y = u$, where $(M)_i$ is the ith row of M, or if $i \in A_2$ and $y_i = 0$. Note that each point from $\overline{P}$ or $\overline{Q}$ may have more labels.

Example 2.26. *Consider the game Battle of Sexes from Example 2.11. We recall that this is a game of two players* 1 *and* 2, *each having two actions, that is,* $m = 2 = n$. *We let* $A_1 = \{1, 2\}$, *and* $A_2 = \{3, 4\}$ *be the sets of the actions. Then the entries of the* 2×2 *payoff matrices* M *and* N *are* $(M)_{1,3} = 2$, $(M)_{2,4} = 1$, $(N)_{1,3} = 1$, $(N)_{2,4} = 2$, *and zero otherwise.*

The best response polyhedra for this game are depicted in Figure 2.2 and are given by

$$\overline{P} = \{(x_1, x_2, v) \in \mathbb{R}^2 \times \mathbb{R} \colon x_1, x_2 \ge 0,$$
$$x_1 + x_2 = 1,$$
$$x_1 \le v, 2x_2 \le v\}$$

and

$$\overline{Q} = \{(y_3, y_4, u) \in \mathbb{R}^2 \times \mathbb{R} \colon y_3, y_4 \ge 0,$$
$$y_3 + y_4 = 1,$$
$$2y_3 \le u, y_4 \le u\}.$$

Let s be a strategy profile for G and let $x \in \mathbb{R}^m$ and $y \in \mathbb{R}^n$ be its mixed strategy vectors. We use $u = u_1(s)$ and $v = u_2(s)$ to denote the corresponding expected payoffs for s.

Proposition 2.27. *The strategy profile* s *is a Nash equilibrium of* G *if and only if the pair* $((x, v), (y, u)) \in \overline{P} \times \overline{Q}$ *is* completely labeled, *that is, every label* $i \in A_1 \cup A_2$ *appears as a label of either* (x, v) *or* (y, u).

Proof. We recall the Best response condition (Observation 2.23), which says that 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}. A missing label i would mean that a pure strategy i from the support of player 1 ($x_i > 0$) was not a best response ($(M)_i y < u$) or that an analogous situation happened for player 2 and his pure strategy j.

On the other hand, if every label appears, then s_1 and s_2 are mutually best responses, as each pure strategy is a best response or it is not in the support. $\qquad\square$

We show that the definition of $\overline{P}$ and $\overline{Q}$ can be modified and simplified under certain mild assumptions. In particular, we can get rid of the expected payoffs u and v from the definitions if they are always positive. Thus, from now on, we assume that the matrices M and $N^\top$ are non-negative and have no zero column. Note that this assumption can be achieved simply by adding a sufficiently large constant to the payoffs, as this does not change the structure of the Nash equilibria.

Then, we can normalize the expected payoffs to 1. By dividing each inequality $(N^\top)_i x \leq v$ in the definition of $\overline{P}$ with v, treating x_i/v as a new variable, and by doing the same for $\overline{Q}$, we obtain the following definition.

Definition 2.28 (Normalized best response polytope[30])**.** *Assume that the matrices M and $N^\top$ are non-negative and have no zero column. The* best response polytope *for player* 1 *in G is a polytope P defined as*

$$P = \{x \in \mathbb{R}^m \colon x \geq \mathbf{0}, N^\top x \leq \mathbf{1}\}.$$

Similarly, the best response polytope for player 2 *in G is a polytope Q defined as*

$$Q = \{y \in \mathbb{R}^n \colon y \geq \mathbf{0}, My \leq \mathbf{1}\}.$$

Note that, in this definition, we do not restrict to vectors x and y whose coordinates sum up to one. For a nonzero non-negative vector x, we can achieve this by *normalizing x to $x/(\mathbf{1}^\top x)$*. For a mixed strategy s_1 of player 1 with the mixed strategy vector $x/(\mathbf{1}^\top x)$, the inequalities in the definition of the normalized best response polytope P have the following meaning. If $x_i \geq 0$ is binding, then $i \in A_1$ is not in the support of s_1. If $(N^\top)_j x \leq 1$ is binding, then $j \in A_2$ is a best response to s_1. Analogous statements are true for the polytope Q.

Example 2.29. *Again, consider the game Battle of Sexes and its best response polyhedra from Example 2.26. After normalizing, we obtain the following two normalized best response polytopes for this game:*

$$P = \{(x_1, x_2) \in \mathbb{R}^2 \colon x_1, x_2 \geq 0,$$
$$x_1 \leq 1, 2x_2 \leq 1\}$$

and

$$Q = \{(y_3, y_4) \in \mathbb{R}^2 \colon y_3, y_4 \geq 0,$$
$$2y_3 \leq 1, y_4 \leq 1\}.$$

That is, P and Q are labeled rectangles $[0, 1] \times [0, 1/2]$ and $[0, 1/2] \times [0, 1]$, respectively; see Figure 2.3.

Since the matrices M and $N^\top$ are non-negative and have no zero column, the polytopes P and Q are bounded (and thus are indeed polytopes) and have dimensions m and n, respectively. The polyhedron $\overline{P}$ is in a one-to-one correspondence with $P \setminus \{\mathbf{0}\}$ under the projective transformation $(x, v) \mapsto x/v$. Analogously, the projective transformation $(y, u) \mapsto y/u$ is a one-to-one correspondence between $\overline{Q}$ and $Q \setminus \{\mathbf{0}\}$. Since projective transformations preserve incidences of faces, the points of P and Q have the same labels as their preimages. In particular, we have the following variant of Proposition 2.27 for P and Q.

[30] Normalizovaný polytop nejlepších odpovědí.

Figure 2.3: The normalized best response polytopes P and Q for the game Battle of Sexes. The labels of points in the facets are denoted by blue and red, respectively, numbers.

Corollary 2.30. *Let $s = (s_1, s_2)$ be a strategy profile for G and let x and y be mixed strategy vectors of s_1 and s_2, respectively. Then s is a Nash equilibrium of G if and only if the point $(x/u_2(s), y/u_1(s)) \in P \times Q \setminus \{(\mathbf{0}, \mathbf{0})\}$ is completely labeled.*

Now, we apply the assumption that G is nondegenerate. This assumption implies that each point of P has at most m labels and each point of Q has at most n labels. This is because if x corresponds to a mixed strategy of player 1 with support of size k, then x has at most $m - k$ labels in A_1 and so if x had more than m labels, then x would have more than k best responses in A_2. An analogous statement is true for Q. Thus, P and Q are both simple, as a point of P or Q contained in more than m or n, respectively, facets has more than m or n, respectively, labels. Moreover, since our assumptions about M and N give that the simple polytopes P and Q have dimension m and n, respectively, only vertices of P can have m labels and only vertices of Q can have n labels. Thus, by Corollary 2.30, only vertices of P and Q can be Nash's equilibria.

Nash's equilibria by vertex enumeration: The last fact suggests another simple algorithm to find all Nash's equilibria in a nondegenerate bimatrix game. It suffices to consider all vertices $x \in P \setminus \{\mathbf{0}\}$ and all vertices $y \in Q \setminus \{\mathbf{0}\}$ and check whether the pair (x, y) is completely labeled. If it is, then we have found a Nash equilibrium with mixed strategy vectors $x/(\mathbf{1}^\top x)$ and $y/(\mathbf{1}^\top y)$ and expected payoffs $\mathbf{1}^\top x$ and $\mathbf{1}^\top y$.

Algorithm 2.4.2: Vertex enumeration(G)

$Input$: A non-degenerate bimatrix game G.
$Output$: All Nash equilibria of G.
for each pair (x, y) of vertices from $(P \setminus \{\mathbf{0}\}) \times (Q \setminus \{\mathbf{0}\})$
$\begin{cases} \text{if } (x, y) \text{ is completely labeled,} \\ \text{then return } (x/(\mathbf{1}^\top x), y/(\mathbf{1}^\top y)) \text{ as a Nash equilibrium} \end{cases}$

The problem of finding all vertices of a simple polytope in $\mathbb{R}^d$ is a well-studied problem in computational geometry. It can be solved efficiently; Avis and Fukuda [AF92] found an algorithm that finds all vertices of a simple polytope in $\mathbb{R}^d$ with v vertices and N defining inequalities in time $O(dNv)$. Unfortunately, even with this result in hand, the simple procedure based on going through all vertices can still take exponentially many steps. For example, in the case $m = n$, the best response polytopes can have c^n vertices for some constant $c > 1$ (although the number of vertices never exceeds 2.9^n by the Upper Bound Theorem, a famous result from combinatorial geometry). Note that it also gives an upper bound on the number of Nash equilibria a nondegenerate bimatrix game can have.

2.4.5 Lemke–Howson algorithm

We now present an algorithm by Lemke and Howson [LH64] for finding Nash equilibrium in a bimatrix game $G = (\{1, 2\}, A, u)$. The algorithm resembles the simplex method from linear programming and uses the normalized best response polytopes P and Q introduced in

Subsection 2.4.4. The analysis of this algorithm yields an alternative and elementary proof of Nash's Theorem (Theorem 2.16).

Recall that Corollary 2.30 says that Nash equilibria in G correspond to completely labeled pairs of vertices from $P \times Q \setminus \{(\mathbf{0}, \mathbf{0})\}$, so our task is to find such a pair. We also recall that we are assuming, without loss of generality, that all entries in the matrices M and N are non-negative and there is no zero column in M nor in $N^\top$. We also assume that G is nondegenerate. Thus, in particular, the best response polytopes P and Q are simple, which means that each vertex of P is incident to exactly m facets and exactly m edges. Similarly, each vertex of Q is incident to exactly n facets and exactly n edges. It thus follows from the definition of labels that every vertex of P has exactly m labels and every vertex of Q has exactly n labels. Similarly, each edge of P has exactly $m - 1$ labels and each edge of Q has exactly $n - 1$ labels.

Dropping a label $l \in A_1 \cup A_2$ in a vertex x of P means traversing the unique edge of P that is incident to x and has all m labels that x has besides l. The other endpoint of this edge has the same labels as x, only the label l is replaced with a new label, which is said to be *picked up*. Dropping and picking up a label in vertices of Q is defined analogously.

The algorithm alternately follows edges of P and Q, keeping the vertex in the other polytope fixed. It starts at the $(\mathbf{0}, \mathbf{0}) \in \mathbb{R}^m \times \mathbb{R}^n$. This is an artificial equilibrium, which is completely labeled, as all the m inequalities $\boldsymbol{x} \geq 0$ in P and all the n inequalities $\boldsymbol{y} \geq 0$ in Q are binding for $(x, y) = (\mathbf{0}, \mathbf{0})$. It is not a pair of mixed strategy vectors of a Nash equilibrium of G, since we cannot rescale them so that the coordinates sum up to 1. As a first step, the algorithm chooses an arbitrary label k from $A_1 \cup A_2$ and drops it. At the endpoint of the corresponding edge (of P if $k \in A_1$ and of Q if $k \in A_2$) a new label l is picked up. This label l has a *duplicate* in the other polytope, that is, l is present among the labels of the current vertex of the other polytope. We drop the duplicate of l in the other polytope in the next step, which leads to picking up a new label l'. If l' has a duplicate, we drop it and continue. Otherwise, l' does not have a duplicate, which means that the pair (x, y) of currently visited vertices of P and Q is completely labeled. This is when the algorithm stops and outputs (x, y).

Algorithm 2.4.3: LEMKE–HOWSON(G)

$Input$: A nondegenerate bimatrix game G.
$Output$: One Nash equilibrium of G.
$(x, y) \leftarrow (\mathbf{0}, \mathbf{0}) \in \mathbb{R}^m \times \mathbb{R}^n$,
$k \leftarrow$ arbitrary label from $A_1 \cup A_2$,
$l \leftarrow k$,
while $(true)$

do $\begin{cases} \text{In } P, \text{ drop the label } l \text{ from } x \text{ and redefine } x \text{ as the new vertex,} \\ \quad \text{redefine } l \text{ as the newly picked up label. Switch to } Q. \\ \text{If } l = k, \text{ stop looping.} \\ \\ \text{In } Q, \text{ drop the label } l \text{ from } y \text{ and redefine } y \text{ as the new vertex,} \\ \quad \text{redefine } l \text{ as the newly picked up label. Switch to } P. \\ \text{If } l = k, \text{ stop looping.} \end{cases}$

Output $(x/(\mathbf{1}^\top x), y/(\mathbf{1}^\top y))$.

In the first step of the Lemke–Howson algorithm, dropping a label means giving the corresponding pure strategy a positive probability. Thus, the first two steps consist of taking a pure strategy k and considering a best response l to k. In a general step of the algorithm, a duplicate label is either a new best response, which gets a positive probability in the next step, or it is a pure strategy whose probability has just become zero and thus it no longer needs to be maintained in the other polytope and the path moves away from the best response facet.

We now verify the correctness of the Lemke–Howson algorithm.

Proposition 2.31. *The Lemke–Howson algorithm stops after a finite number of steps and outputs a pair of mixed strategy vectors of a Nash equilibrium of G.*

Proof. Let k be the label chosen in the first step of the Lemke–Howson algorithm. We define a *configuration graph* with the vertex set formed by those pairs (x, y) of vertices from $P \times Q$ that are k-*almost completely labeled*, meaning that every label from $A_1 \cup A_2 \setminus \{k\}$ appears as a label of x or of y. Clearly, the configuration graph is finite as the polytopes P and Q have finitely many vertices. Two vertices (x, y) and (x', y') of the configuration graph are connected by an edge if either $x = x'$ and yy' is an edge of Q or if xx' is an edge of P and $y = y'$.

We show that the configuration graph has degrees only 1 or 2, so it is a disjoint union of paths and cycles. There are two types of vertices in the configuration graph. If (x, y) has all labels from $A_1 \cup A_2$, then (x, y) is connected to exactly one other vertex, as exactly one of the vertices x and y has the label k and we can drop k only from this one vertex. Otherwise (x, y) has all labels from $A_1 \cup A_2 \setminus \{k\}$ and there is a unique label shared by x and y. Then (x, y) is adjacent to two vertices, as we can drop the duplicate label from x in P or from y in Q.

The Lemke–Howson algorithm starts at the leaf $(\mathbf{0}, \mathbf{0})$ of a path in the configuration graph. The algorithm then walks along this path and does not visit any vertex of the configuration graph twice, as the next vertex pair on the path is always unique (we move to the other endpoint of the unique edge that contains current vertex and corresponds to the dropped label) and visiting the same vertex twice would give a vertex of degree larger than 2 in the configuration graph since we started at a leaf. Thus, the algorithm terminates after a finite number of steps in the other leaf (x^*, y^*) of the path. Since (x^*, y^*) is a leaf in the configuration graph, it is completely labeled. We leave as an exercise to show that this endpoint is not of the form $(x, \mathbf{0})$ or $(\mathbf{0}, y)$; see Exercise 2.16. By Corollary 2.30, the completely labeled pair (x^*, y^*) on the output corresponds to a Nash equilibrium of G after rescaling the coordinates so that they sum up to 1. $\qquad\square$

The degree sum formula implies that the number of vertices of degree 1 is even in the configuration graph. The set of such vertices consists of $(\mathbf{0}, \mathbf{0})$ and of pairs of mixed strategy vectors of Nash equilibria of G. Thus, we obtain the following result, which yields an alternative proof of Nash's Theorem (Theorem 2.16).

Corollary 2.32. *A nondegenerate bimatrix game has an odd number of Nash equilibria.*

The Lemke–Howson algorithm can start at any Nash equilibrium of G, not just at the artificial equilibrium $(\mathbf{0}, \mathbf{0})$. However, there might be a Nash equilibrium that cannot be found by the Lemke–Howson algorithm; see Exercise 2.17. We also remark that degenerate games can have an infinite number of Nash equilibria [NRTV07].

Although the Lemke–Howson algorithm can take an exponential number of iterations [AH02, Chapter 45], it performs well in practice and is considered among the best algorithms for finding Nash equilibria in bimatrix games. It has been experimentally shown that, on uniformly random games, the Lemke–Howson algorithm runs in polynomial time and some of its heuristic modifications run even in linear time [CDRP08].

We describe an efficient implementation of the Lemke–Howson algorithm. This method is called *complementary pivoting* and it resembles the simplex algorithm for solving a linear program.

First, we introduce new variables s and r, called *slack variables*, so that the polytopes P and Q are represented by a system of linear equations. The slack variables $s \in \mathbb{R}^n$ and $r \in \mathbb{R}^m$ are non-negative vectors such that $x \in P$ and $y \in Q$ if and only if

$$N^\top x + s = \mathbf{1} \quad \text{and} \quad r + My = \mathbf{1} \tag{2.5}$$

and

$$x \geq \mathbf{0}, \quad s \geq \mathbf{0}, \quad r \geq \mathbf{0}, \quad y \geq \mathbf{0}. \tag{2.6}$$

The ith inequality $(N^\top)_i x \leq 1$ is binding (that is, $(N^\top)_i x = 1$) if and only if $s_i = 0$. Similarly, the ith inequality of $My \leq \mathbf{1}$ is binding if and only if $r_i = 0$. Thus, slack variables correspond to binding inequalities and represent facets of the polytopes. The pair (x, y) is completely

labeled if and only if $x_i r_i = 0$ for every $i \in A_1$ and $y_j s_j = 0$ for every $j \in A_2$. By (2.6), this is equivalent with the orthogonality conditions $x^\top r = 0$ and $y^\top s = 0$.

A *basic solution* to (2.5) is given by n *basic* (linearly independent) columns of $N^\top x + s = \mathbf{1}$ and m *basic* columns of $r + My = \mathbf{1}$, where the *nonbasic* variables that correspond to the m and n, respectively, other columns are set to zero, so that the basic variables are uniquely determined. A basic *feasible* solution to (2.5) also satisfies (2.6). Thus, every basic feasible solution determines a vertex $x \in P$ and $y \in Q$. The labels of these vertices are given by the respective nonbasic variables. Note that basic variables and sets of labels have opposite meanings, as labels imply a binding inequality. Thus, "nonbasic variable enters the basis" means the same as "label is dropped" and "basic variable leaves the basis" is the same as "label is added".

Pivoting is a change of the basis where a nonbasic variable enters and a basic variable leaves the set of basic variables. For a given entering variable z_j, the leaving variable is chosen to preserve feasibility of the basis, that is, the inequalities (2.6). This is done by performing the so-called *minimum ratio test*. In nondegenerate games, the leaving variable is determined uniquely by this test. If we write the system given by (2.5) and (2.6) as $Cz = q$ for some real matrix $C \in \mathbb{R}^{(m+n) \times 2(m+n)}$ and vectors $z \in \mathbb{R}^{2(m+n)}, q \in \mathbb{R}^{m+n}$, then the variable z_i that leaves a basis $\{C_l : l \in \beta\}$ of the space spanned by the columns C_l of C is determined as follows. Set $C_\beta^{-1} q = (\bar{q}_1, \ldots, \overline{q_{m+n}})$ and $(C_\beta^{-1} C_j)_i = \bar{c}_{i,j}$ for $i \in \beta$, where C_β is a square matrix formed by columns with indices from β. We choose $i \in \beta$ such that $\bar{c}_{i,j} > 0$ and $\bar{q}_i / \bar{c}_{i,j}$ is minimized. Note that $z_\beta = (z_j)_{j \in \beta}$ is the vector of basic solutions and satisfies $C_\beta z_\beta = q$.

The choice of the entering variable depends on the solution that one wants to find. In the Simplex method for linear programming, one chooses this variable in order to improve the value of the objective function. Here, we look for a complementary solution where $x^\top r = 0$ and $y^\top s = 0$. Since nonbasic variables represent facets, each basis defines a vertex which is labeled with the indices of the nonbasic variables. The variables thus form *complementary pairs* (x_i, r_i) for $i \in A_1$ and (y_j, s_j) for $j \in A_2$. Thus, a k-almost completely labeled vertex is a basis that has exactly one basic variable from each complementary pair, except for a pair of variables (x_k, r_k) if $k \in A_1$ or of (y_k, s_k) if $k \in A_2$ that are both basic. Correspondingly, there is another pair of complementary variables that are both nonbasic, representing the duplicate label. One of them is chosen as the entering variable, depending on the direction of the computed path.

The pivoting algorithm starts by choosing an arbitrary label $k \in A_1 \cup A_2$ and by letting the first entering variable be x_k if $k \in A_1$ and y_k if $k \in A_2$. Then we pivot using the minimum ratio test while maintaining feasibility. We stop when the variable that left the basis has index k.

Example 2.33. *We use the Lemke–Howson algorithm and compute a Nash equilibrium of the following bimatrix game*

$$M = \begin{pmatrix} 0 & 6 \\ 2 & 5 \\ 3 & 3 \end{pmatrix} \quad \textit{and} \quad N = \begin{pmatrix} 1 & 0 \\ 0 & 2 \\ 4 & 3 \end{pmatrix}.$$

The rows are indexed by $A_1 = \{1, 2, 3\}$ and the columns by $A_2 = \{4, 5\}$. Let $k = 2$ be the missing label that we choose. First, we introduce the slack variables r_1, r_2, r_3, s_4, s_5 and write the corresponding system of equations in the form $z_\beta = C_\beta^{-1} q - \sum_{j \notin \beta} C_\beta^{-1} C_j z_j$ as

$$
\begin{aligned}
r_1 &= 1 && - 6y_5 \\
r_2 &= 1 - 2y_4 - 5y_5 \\
r_3 &= 1 - 3y_4 - 3y_5 \\
s_4 &= 1 && - x_1 && - 4x_3 \\
s_5 &= 1 && - 2x_2 - 3x_3.
\end{aligned}
$$

Since the missing label is 2, the first entering variable is x_2. By the minimum ratio test, we look at the column that contains x_2 and search for the minimum ratio among all rows. There is only a single option and thus s_5 leaves the basis. The fifth equation is rewritten as $x_2 = \frac{1}{2} - \frac{3}{2}x_3 - \frac{1}{2}s_5$.

The complement of s_5 is y_5, thus y_5 enters the basis. Now, the minimum ratio in the column that contains y_5 is $1/6$ and so r_1 leaves the basis. We get

$$
\begin{aligned}
y_5 &= \frac{1}{6} & & -\frac{1}{6}r_1 \\
r_2 &= \frac{1}{6} - 2y_4 & & +\frac{5}{6}r_1 \\
r_3 &= \frac{1}{2} - 3y_4 & & +\frac{1}{2}r_1 \\
s_4 &= 1 & & -x_1 - 4x_3 \\
x_2 &= \frac{1}{2} & & -\frac{3}{2}x_3 - \frac{1}{2}s_5.
\end{aligned}
$$

The complement of r_1 is x_1 and it enters the basis. By the minimum ratio test, s_4 leaves the basis (again, there is only one option) and we obtain

$$
\begin{aligned}
y_5 &= \frac{1}{6} & & -\frac{1}{6}r_1 \\
r_2 &= \frac{1}{6} - 2y_4 & & +\frac{5}{6}r_1 \\
r_3 &= \frac{1}{2} - 3y_4 & & +\frac{1}{2}r_1 \\
x_1 &= 1 & & -4x_3 - s_4 \\
x_2 &= \frac{1}{2} & & -\frac{3}{2}x_3 & & -\frac{1}{2}s_5.
\end{aligned}
$$

The complement of s_4 is y_4 and it enters the basis. The minimum ratio is $1/12$ and thus r_2 leaves the basis and we have

$$
\begin{aligned}
y_5 &= \frac{1}{6} - \frac{1}{6}r_1 \\
y_4 &= \frac{1}{12} + \frac{5}{12}r_1 - \frac{1}{2}r_2 \\
r_3 &= \frac{1}{4} - \frac{3}{4}r_1 + \frac{3}{2}r_2 \\
x_1 &= 1 & & -4x_3 - s_4 \\
x_2 &= \frac{1}{2} & & -\frac{3}{2}x_3 & & -\frac{1}{2}s_5.
\end{aligned}
$$

Then the algorithm terminates since the variable r_2, with the missing label 2 as index, has become nonbasic. Setting the nonbasic variables to zero (that is, the variables on the right side), we obtain a solution, which, after re-normalizing, gives a pair of mixed strategy vectors $(x/(\mathbf{1}^\top x), y/(\mathbf{1}^\top y)) = ((\frac{2}{3}, \frac{1}{3}, 0)^\top, (\frac{1}{3}, \frac{2}{3})^\top)$. This gives a Nash equilibrium of this game. The other two Nash equilibria are determined by pairs of vectors $((0, \frac{1}{3}, \frac{2}{3})^\top, (\frac{2}{3}, \frac{1}{3})^\top)$ and $((0, 0, 1)^\top, (1, 0)^\top)$.

Finally, we briefly mention how to compute Nash equilibria in degenerate games. The problem with degenerate games that arises in the Lemke–Howson algorithm is that the variable that leaves the basis might not be determined uniquely by the minimum ratio test. Thus, we have to find a good "tie-breaking rule" that our algorithm chooses. Unfortunately, the algorithm can enter a loop and run forever if we choose a bad rule. A good tie-breaking rule, running in a polynomial time, can be obtained by "lexicographic perturbation". We do not explain this method here, instead, we refer an interested reader to [Ste02].

2.4.6 The class PPAD

We have shown a few algorithms how to compute a Nash equilibrium of a bimatrix game. Unfortunately, all the presented algorithms, including the Lemke–Howson algorithm, can take

an exponential number of iterations. In fact, despite many efforts, there is still no known polynomial-time algorithm for computing a Nash equilibrium of a bimatrix game. In such cases, one usually applies tools from the complexity theory and finds a reduction to a problem for which there is an evidence suggesting it is computationally intractable. For example, finding a reduction to an NP-hard problem. Here, we develop the relevant complexity theory for arguing that computing Nash's equilibrium of a bimatrix game may be intractable. The goal is to prove that the problem is complete for a suitable complexity class.

Let *NASH* be the problem of finding a Nash equilibrium of a bimatrix game. The problem NASH is not a good candidate for an NP-complete problem, since, by Nash's Theorem (Theorem 2.16), Nash equilibria always exist in normal-form games. Thus, if we treat NASH as a decision problem, the answer is always 'yes'. Instead, we work with the closely related class *FNP* ("functional NP"). The input of an algorithm for an FNP problem is an instance of a problem from NP. The algorithm outputs a solution, provided one exists. If there is no solution, the algorithm outputs 'no'. Informally, FNP problems are just like NP problems except that we demand a solution to be produced for 'yes' instances. For example, an FNP algorithm for SAT receives a CNF formula φ and outputs an interpretation that satisfies φ, if it exists, and 'no' otherwise. The functional version of SAT is also an example of an FNP-complete problem.

The problem NASH belongs to the class FNP, as checking whether a strategy profile is a Nash equilibrium can be done efficiently, using the Best Response Condition (Observation 2.23), which says that it suffices to check only pure strategies. To be rigorous, we also note that there are Nash equilibria of 2-player games whose description in bits takes polynomial size with respect to the size of the input, since, as we already know, computing the mixed strategy vectors of a Nash equilibrium with given supports requires to solve a system of linear equations.

Is NASH FNP-complete? If so, we would have strong evidence of the intractability of NASH. Unfortunately, the following result suggests that it is unlikely, as it is widely believed that $NP \neq coNP$.

Theorem 2.34 ([MP91])**.** *If the problem NASH is FNP-complete, then NP = coNP.*

Proof. Suppose there is a polynomial-time reduction from the functional SAT problem to NASH. That is, there are two polynomial-time algorithms A and B that satisfy the following conditions. The algorithm A maps every SAT formula φ to a bimatrix game $A(\varphi)$. The algorithm B maps every Nash equilibrium (s_1, s_2) of a bimatrix game $A(\varphi)$ to a satisfying assignment $B(s_1, s_2)$ of φ, if one exists, and to the string 'no' otherwise.

Now, to show that NP = coNP, we show that the NP-complete problem SAT is in coNP. Then we have $NP \subseteq coNP$, which implies that the set of *complements* of the problems in NP (problems where yes and no answers are switched) is a subset of the set of complements in coNP. Formally, we have $X \in NP \Leftrightarrow \overline{X} \in coNP$ by the definition of coNP, where $\overline{X}$ is the complement of X. Since $NP \subseteq coNP$, we get $X \in coNP \Leftrightarrow \overline{X} \in NP \Rightarrow \overline{X} \in coNP \Leftrightarrow X \in NP$. That is, we have coNP $\subseteq$ NP as well and thus NP = coNP.

It suffices to prove that, for an arbitrary 'no' instance φ of SAT, an arbitrary Nash equilibrium (s_1, s_2) of $A(\varphi)$ is a short and efficiently verifiable proof of the unsatisfiability of φ, as then SAT is in coNP. We recall that at least one Nash equilibrium of $A(\varphi)$ is guaranteed to exist by Nash's Theorem (Theorem 2.16). To show that a given Nash equilibrium (s_1, s_2) of $A(\varphi)$ is an efficiently verifiable proof of unsatisfiability of φ, it suffices to perform two checks. First, compute the game $A(\varphi)$ using A and verify that (s_1, s_2) is a Nash equilibrium of $A(\varphi)$. Second, use B to verify that $B(s_1, s_2)$ is the 'no' string. If (s_1, s_2) passes both of these tests, then we have verified in polynomial time that φ is unsatisfiable. □

Note that the proof of Theorem 2.34 uses the mismatch between the functional version of SAT, which does not have to have a solution, and between NASH, where the solution is always guaranteed to exist.

Altogether, we see that we need to consider a more specialized complexity class than FNP. Inspecting the proof of Nash's Theorem (Theorem 2.16), we see that the existence of a Nash equilibrium in a bimatrix game is established using a reduction of the problem to Brouwer's Fixed Point Theorem. Unfortunately, this is not very helpful, since the problem *BROUWER*

of finding Brouwer's fixed point is known to be a hard problem [HPV89, Pap94]. It turns out that the reduction holds also in the opposite direction and thus, in a sense, NASH is at least as difficult as BROUWER.

Consider the proof of the correctness of the Lemke–Howson algorithm (proof of Proposition 2.31). In the proof we have constructed a so-called configuration graph, which is a graph where every degree is either one or two and there is a specified *source vertex* of degree one, which corresponds to the "artificial equilibrium" $(\mathbf{0}, \mathbf{0})$. So the graph is a disjoint union of cycles and paths. It is possible to direct edges of this graph so that every vertex has in-degree at most one and out-degree at most one and the source has out-degree one. This representation eventually gives us a Nash equilibrium, since the other endpoints correspond to Nash equilibria.

We have thus arrived at the following abstract structure of the problem NASH (for nondegenerate games, but something similar holds in the general case as well), which we call the *END-OF-THE-LINE* problem: for a directed graph G with every vertex having at most one predecessor and one successor, given a vertex s of G with no predecessor, find a vertex $t \neq s$ with no predecessor or no successor. This problem might seem trivial, however, there is a subtlety involved, as the graph G is not actually given to us on the input, but it is specified by some polynomial-time computable function $f(v)$ that returns the predecessor and successor (if they exist) of a vertex v (or, alternatively, G is generated from a boolean circuit). Thus, G can be exponentially large with respect to the size of the input.

There is a simple algorithm to solve END-OF-THE-LINE. It suffices to start from s and follow the path until we arrive at the other endpoint of the path. Unfortunately, this algorithm is not efficient, as G can be exponentially large. What makes the problem END-OF-THE-LINE interesting is that several other problems admit this structure, for example the problem of finding an approximate Brouwer's fixed point of a function or the problem *HAM SANDWICH*: given n sets of $2n$ points in $\mathbb{R}^n$, find a hyperplane H that contains exactly n points from each of the sets in each open halfspace determined by H. More examples of such problems can be found in [Pap94]. There is no known polynomial-time algorithm for any of these problems.

One way of thinking about the END-OF-THE-LINE problem is that it is an abstract representation of some given problem (NASH, for example), where the vertices of the large directed graph correspond to feasible solutions and there is an edge from u to v is the feasible solution can be improved to a solution v. Of course, if we then insist on solving the given problem using these local search steps, then finding the solution might require exponentially many steps, as the directed graph can be large. However, it is not clear whether there is some more "clever" algorithm that forgets about local improvements and can "zoom out" around the search space and ignore the structure of the directed graph. This is what makes determining the complexity of this problem hard.

Let *PPAD* ("Polynomial Parity Arguments on Directed graphs") be a complexity class consisting of problems that admit a polynomial-time reduction to END-OF-THE-LINE. This class was introduced by Papadimitriou [Pap94] in 1994 and PPAD-complete problems are believed to be hard, although obtaining PPAD-completeness is a weaker evidence of intractability than NP-completeness. Even though it seems unlikely, it could still be the case that PPAD = P while P $\neq$ NP.

The proof of the correctness of the Lemke–Howson algorithm shows that NASH for nondegenerate games belongs to PPAD. The following result, proved by Chen, Deng, and Teng [CDT09] and Daskalakis, Goldberg, and Papadimitriou [DGP09], shows that the problem NASH is PPAD-hard and it is one the main breakthroughs in algorithmic game theory.

Theorem 2.35 ([CDT09, DGP09])**.** *The problem NASH is PPAD-complete.*

We do not present the proof here, as it is quite technical. However, an interested reader might check overviews of the proof written by Papadimitriou [Pap07] or by Roughgarden [Rou10].

So far we have considered only bimatrix games, that is, games with two players. One may use so-called *Sperner's lemma* (see Exercise 2.19) to show that the problem of finding approximate Nash equilibria in games with at least three players lies in PPAD. However, the problem of computing an exact Nash equilibrium of a game of at least three players appears

to be strictly harder than PPAD [EY10]. One of the reasons why the problem of computing exact Nash equilibria of games with at least three players is harder is that there might not be a rational Nash equilibrium [Nas51]. On the other hand, in games of two players with rational payoffs there are always rational Nash equilibria and thus they can be computed exactly.

Finally, let us mention that if we modify the problem NASH in some way so that the existence is not always guaranteed, then, quite often, it is the case that the resulting problem becomes NP-complete. For example, given a bimatrix game G, the following decision problems are NP-complete [GZ89]. Decide whether G has

- at least two Nash equilibria.

- a Nash equilibrium whose support contains a strategy s.

- a Nash equilibrium whose support does not contain a strategy s.

- a Nash equilibrium in which both players have the total utility at least a given amount.

2.4.7 Exercises

Exercise 2.10. *Use the Support enumeration algorithm to find a Nash equilibrium of the Game of chicken with supports of size 2.* **[1]**

Exercise 2.11. *Decide whether the Game for Gotham's soul is degenerate and find all Nash equilibria of this game. How is the set of equilibria different from previously computed examples?* **[2]**

	Cooperate (1)	Detonate (2)
Cooperate (1)	$(0,0)$	$(0,1)$
Detonate (2)	$(1,0)$	$(0,0)$

Table 2.9: The Game for Gotham's soul.

Exercise 2.12. *Decide which of these two payoff matrices determines a degenerate game.* **[2]**

(a) $M = \begin{pmatrix} 0 & 4 & 1 \\ 2 & 2 & 4 \\ 3 & 2 & 2 \end{pmatrix}$ *and* $N = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 3 & 4 \\ 0 & 1 & 1 \end{pmatrix}$.

(b) $M = \begin{pmatrix} 0 & 4 & 1 \\ 2 & 2 & 4 \\ 3 & 2 & 2 \end{pmatrix}$ *and* $N = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 3 & 1 \\ 3 & 4 & 1 \end{pmatrix}$.

Exercise 2.13. *Prove that if a bimatrix game is non-degenerate, then the system of equations in the Support enumeration algorithm has a unique solution with $\mathbf{x}, \mathbf{y} > \mathbf{0}$, $u = \max\{(M)_i y \colon i \in A_1\}$ and $v = \max\{(N^\top)_j x \colon j \in A_2\}$.* **[4]**
Hint: Prove that if there are more solutions, then we can reduce the support.

Exercise 2.14. *Show that the following linear programs from the proof of the Minimax Theorem (Theorem 2.21) are dual to each other.*

(a) *For a matrix $M \in \mathbb{R}^{m \times n}$,* **[2]**

(b) *For a matrix $M \in \mathbb{R}^{m \times n}$,* **[2]**

You may use the recipe for making dual programs from Table 2.8.

	Program P	Program D
Variables	$y_1, \ldots, y_n$	x_0
Objective function	$\min x^\top M y$	$\max x_0$
Constraints	$\sum_{j=1}^n y_j = 1,$	$\mathbf{1}x_0 \le M^\top x.$
	$y_1, \ldots, y_n \ge 0.$	

	Program P'	Program D'
Variables	$y_0, y_1, \ldots, y_n$	$x_0, x_1, \ldots, x_m$
Objective function	$\min y_0$	$\max x_0$
Constraints	$\mathbf{1}y_0 - My \ge \mathbf{0},$	$\mathbf{1}x_0 - M^\top x \le \mathbf{0},$
	$\sum_{j=1}^n y_j = 1,$	$\sum_{i=1}^m x_i = 1,$
	$y_1, \ldots, y_n \ge 0.$	$x_1, \ldots, x_m \ge 0.$

Exercise 2.15. *Use the Lemke–Howson algorithm and compute a Nash equilibrium of the following bimatrix game* **[3]**

$$M = \begin{pmatrix} 1 & 3 & 0 \\ 0 & 0 & 2 \\ 2 & 1 & 1 \end{pmatrix} \quad and \quad N = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 0 & 3. \end{pmatrix}$$

Start the computation by choosing the label 1.

Exercise 2.16. *Show that the Lemke–Howson algorithm does not terminate in a vertex of the form $(x, \mathbf{0})$ or $(\mathbf{0}, y)$ in the configuration graph.* **[2]**

Exercise 2.17. *Show that the following symmetric 3×3 game has a Nash equilibrium that cannot be found by the Lemke–Howson algorithm.* **[3]**

$$M = \begin{pmatrix} 3 & 3 & 0 \\ 4 & 0 & 1 \\ 0 & 4 & 5 \end{pmatrix} = N^\top$$

Exercise 2.18. *Prove that if (s_1, s_2) and (s_1', s_2') are mixed Nash equilibria of a two-player zero-sum game, then so are (s_1, s_2') and (s_1', s_2).* **[2]**

Exercise 2.19. *[Sperner's Lemma] Let S be a given subdivision of a triangle T in the plane. A legal coloring the vertices of S assigns one of three colors (red, blue, and green) to each vertex of S such that all the three colors are used on the vertices of T. Moreover, a vertex of S lying on an edge of T must have one of the two colors of the endpoints of this edge.*

Prove that, in every legal coloring of S, there is is a triangular face of S whose vertices are colored with all three colors.

Hint: Use a reduction to the END-OF-THE-LINE problem. **[3]**

2.5 Other notions of equilibria

As we have seen, finding Nash equilibria is a difficult computational task and possibly intractable unless PPAD $\subseteq$ FP. This fact justifies introducing other solution concepts that possess some of the qualities of Nash equilibria and yet are easier to compute and work with. After all, how can we expect the players to find a Nash equilibrium, if our computers cannot?

In this section we introduce two such concepts: approximate and correlated Nash equilibria.

2.5.1 ε-Nash equilibria

The following solution concept captures the idea that players do not care about changing their strategies when the change results in a very small gain.

Definition 2.36 (ε-Nash equilibrium)**.** *Let $G = (P, A, u)$ be a normal-form game of n players and let $\varepsilon > 0$. A strategy profile $s = (s_1, \ldots, s_n)$ is an ε-Nash equilibrium[31] if, for every player $i \in P$ and for every strategy $s'_i \in S_i$, we have $u_i(s_i; s_{-i}) \geq u_i(s'_i; s_{-i}) - \varepsilon$.*

In other words, an ε-Nash equilibrium is a mixed strategy profile such that no other strategy can improve the payoff by more than an additive ε. We remark that computing an approximate Nash equilibrium in two-player games where ε acts multiplicatively is PPAD-complete, even for constant values of the approximation [Das13].

It follows from Nash's Theorem (Theorem 2.16) that ε-Nash equilibria exist in every normal-form game. Indeed, every Nash equilibrium is surrounded by a region of ε-Nash equilibria for every $\varepsilon > 0$. The concept is also computationally useful since the probabilities involved in an exact Nash equilibrium need not be rational numbers. However, computers usually work with real numbers using a floating-point approximation and thus only ε-Nash equilibria, where ε is roughly the "machine precision".

On the other hand, there are also some disadvantages to this solution concept. For example, although every Nash equilibrium is surrounded by a region of ε-Nash equilibria, it might be the case that a given ε-Nash equilibrium is not close to any Nash equilibrium; see Exercise 2.20. Thus, we cannot think of ε-Nash equilibria as approximations of the true Nash equilibria.

Obviously, an ε-Nash equilibrium is not very compelling solution concept unless ε is very small. Thus, we would ideally want to have an efficient scheme for approximating ε-Nash equilibria. An optimization problem P with input of size n and with a parameter $\varepsilon > 0$ has a *PTAS* (*polynomial-time approximation scheme*) if there is an algorithm that computes an ε-approximate solution of P in time $O(n^{f(1/\varepsilon)})$ for some function f. The problem P is said to have *FPTAS* (*fully polynomial-time approximation scheme*) if there is such an algorithm that runs in time $O((1/\varepsilon)^c n^d)$ for some constants c and d.

Chen et al. [CDhT06] showed that FPTAS for the problem of computing ε-Nash equilibria is impossible unless PPAD $\subseteq$ FP. The existence of PTAS for this problem remains an interesting and challenging open problem.

In the following result due to Lipton, Markakis, and Mehta [LMM03], we show that there is an approximation scheme for computing ε-Nash equilibria in normal-form games of two players running in a quasi-polynomial time. This result can be extended to normal-form games of more players; see [LMM03, Theorem 3].

Theorem 2.37 ([LMM03])**.** *Let $G = (P, A, u)$ be a normal-form game of two players, each having m actions, such that the payoff matrices have entries between 0 and 1. For every $\varepsilon > 0$, there exists an algorithm for computing an ε-Nash equilibrium for G in time $m^{O(\log m / \varepsilon^2)}$.*

Before proceeding with the proof, we recall a standard tail inequality called a *Hoeffding bound*.

Theorem 2.38 (The Hoeffding bound [Hoe63])**.** *Let $Z = \frac{1}{s} \sum_{i=1}^{s} Z_i$ be a random variable, where $Z_1, \ldots, Z_s$ are independent random variables with $0 \leq Z_i \leq 1$. Let $\mu = \mathbb{E}[Z]$. Then, for every t with $0 < t < 1 - \mu$,*

$$\Pr[Z - \mu \geq t] \leq e^{-2st^2}.$$

Proof of Theorem 2.37. Let M and N be the $m \times m$ payoff matrices of players 1 and 2, respectively, where $m \geq 2$. We assume that all entries of these matrices are between 0 and 1, otherwise we have to rescale, which magnifies the ε by $\max\{\max\{M_{i,j}\} - \min\{M_{i,j}\}, \max\{N_{i,j}\} - \min\{N_{i,j}\}\}$. Since G is a bimatrix game, we see that a pair (x, y) of mixed strategy vectors forms an ε-Nash equilibrium for some $\varepsilon > 0$ if and only if for every mixed strategy vector x' of player 1 we have $(x')^\top M y \leq x^\top M y + \varepsilon$ and for every mixed strategy vector y' of player 2 we have $x^\top N y' \leq x^\top N y + \varepsilon$.

---

[31] ε-Nashova rovnováha či ε-Nashovovo ekvilibrium.

A mixed strategy is k-*uniform* if it is the uniform distribution on a multiset S of pure strategies with $|S| = k$ (we count the elements of S with their multiplicities). We first prove the following statement.

Claim 2.39. *For every Nash equilibrium of G with mixed strategy vector (x^*, y^*) and for every $\varepsilon > 0$, there exists, for every integer $k \geq 48 \ln m/\varepsilon^2$, a pair (x, y) of k-uniform strategies such that (x, y) is an ε-Nash equilibrium, $|x^\top My - (x^*)^\top My^*| < \varepsilon$, and $|x^\top Ny - (x^*)^\top Ny^*| < \varepsilon$.*

The last two conditions say that players 1 and 2, respectively, get almost the same payoff as in the Nash equilibrium (x^*, y^*).

To see that Claim 2.39 implies Theorem 2.37, fix $k = 48 \ln m/\varepsilon^2$ and perform an exhaustive search for all k-uniform ε-Nash equilibria of G. There are at most $(m^k)^2$ pairs of multisets X and Y to look at and verifying ε-Nash equilibrium condition is easy, as we need to check only for deviations to pure strategies. By the choice of k, the algorithm runs in time $m^{O(\log m/\varepsilon^2)}$. By Nash's theorem (Theorem 2.16) and Claim 2.39, at least one k-uniform ε-Nash equilibrium exists and thus the algorithm eventually finds one.

The proof of Claim 2.39 is based on a probabilistic argument. For a given $\varepsilon > 0$, we fix an integer $k \geq 48 \ln m/\varepsilon^2$. Let X be a multiset obtained by sampling k times from the set A_1 of actions of player 1 according to the distribution x^*. Analogously, let Y be a multiset obtained by sampling k times from the set A_2 of actions of player 2 according to y^*. We choose x to be the mixed strategy vector that assigns a probability $\mu(a_i)/k$ to each action $a_i \in A_1$, where $\mu(a_i)$ is the multiplicity of a_i in X. The mixed strategy vector y is then defined analogously using the multiset Y.

Let $A_1 = \{a_1, \ldots, a_m\}$ and $A_2 = \{b_1, \ldots, b_m\}$ be labelings of actions of both players. We let x^i be the mixed strategy vector for each pure strategy $a_i \in A_1$ and y^j be the mixed strategy vector for each pure strategy $b_j \in A_2$. To check that (x, y) is an ε-Nash equilibrium, it suffices to consider only deviations to pure strategies. In order to do so, we define the following events:

$$\phi_1 = \{(x, y) \colon |x^\top My - (x^*)^\top My^*| < \varepsilon/2\},$$
$$\pi_{1,i} = \{(x, y) \colon (x^i)^\top My < x^\top My + \varepsilon\} \quad \text{for every } i \in \{1, \ldots, m\},$$
$$\phi_2 = \{(x, y) \colon |x^\top Ny - (x^*)^\top Ny^*| < \varepsilon/2\},$$
$$\pi_{2,j} = \{(x, y) \colon x^\top Ny^j < x^\top Ny + \varepsilon\} \quad \text{for every } j \in \{1, \ldots, m\},$$
$$GOOD = \phi_1 \cap \phi_2 \cap \bigcap_{i=1}^m \pi_{1,i} \cap \bigcap_{j=1}^m \pi_{2,j}.$$

Note that to prove Claim 2.39 it suffices to show $\Pr[GOOD] > 0$. Indeed, if the event GOOD holds with positive probability, then there is a pair of (X, Y) of multisets that determine the mixed strategy vectors (x, y) satisfying the events ϕ_1 and ϕ_2, which guarantees that both players get almost the same payoff as in the Nash equilibrium (x^*, y^*). Moreover, the vectors x and y satisfy all the events $\pi_{i,j}$ for $i \in \{1, 2\}$ and $j \in \{1, \ldots, m\}$, which, by the linearity of expectation, means that (x, y) is an ε-Nash equilibrium.

We first estimate the probabilities of the events $\overline{\phi_1}$ and $\overline{\phi_2}$, which are the complements of the events ϕ_1 and ϕ_2, respectively. To do so, we consider the following auxiliary events:

$$\phi_{1,a} = \{(x, y) \colon |x^\top My^* - (x^*)^\top My^*| < \varepsilon/4\},$$
$$\phi_{1,b} = \{(x, y) \colon |x^\top My - x^\top My^*| < \varepsilon/4\},$$
$$\phi_{2,a} = \{(x, y) \colon |(x^*)^\top Ny - (x^*)^\top Ny^*| < \varepsilon/4\},$$
$$\phi_{2,b} = \{(x, y) \colon |x^\top Ny - (x^*)^\top Ny| < \varepsilon/4\}.$$

We then have $\phi_{1,a} \cap \phi_{1,b} \subseteq \phi_1$, since

$$|x^\top My - (x^*)^\top My^*| \leq |x^\top My - x^\top My^*| + |x^\top My^* - (x^*)^\top My^*| < \varepsilon/4 + \varepsilon/4 = \varepsilon/2.$$

Analogously, $\phi_{2,a} \cap \phi_{2,b} \subseteq \phi_2$.

The choice of x corresponds to choosing each element of the multiset X independently at random with probability $1/k$. Thus, the expression $x^\top My^*$ is essentially a sum of k independent random variables, where each variable takes a value between 0 and 1 and has expected value $(x^*)^\top My^*$. Thus, the Hoeffding bound (Theorem 2.38) gives $\Pr[\overline{\phi_{1,a}}] \leq 2e^{-k\varepsilon^2/8}$ and, analogously, $\Pr[\overline{\phi_{1,b}}] \leq 2e^{-k\varepsilon^2/8}$. By the union bound, we obtain $\Pr[\overline{\phi_1}] \leq 4e^{-k\varepsilon^2/8}$ using the fact $\phi_{1,a} \cap \phi_{1,b} \subseteq \phi_1$. Analogously, we derive the same estimate for ϕ_2.

It remains to estimate probabilities for the events $\pi_{i,j}$. Again, we first introduce some auxiliary events. More specifically, we define

$$\psi_{1,i} = \{(x,y)\colon (x^i)^\top My < (x^i)^\top My^* + \varepsilon/2\} \quad \text{for every } i \in \{1,\dots,m\}$$

and

$$\psi_{2,j} = \{(x,y)\colon x^\top Ny^j < (x^*)^\top Ny^j + \varepsilon/2\} \quad \text{for every } j \in \{1,\dots,m\}.$$

Then we obtain $\psi_{1,i} \cap \phi_1 \subseteq \pi_{1,i}$ and $\psi_{2,j} \cap \phi_2 \subseteq \pi_{2,j}$ for all $i,j \in \{1,\dots,m\}$, as the fact that (x^*,y^*) is a Nash equilibrium gives $(x^i)^\top My^* \leq (x^*)^\top My^*$ for every $a_i \in A_1$ and thus, by ϕ_1 and $\psi_{1,i}$,

$$(x^i)^\top My < (x^i)^\top My^* + \varepsilon/2 \leq (x^*)^\top My^* + \varepsilon/2 < x^\top My + \varepsilon.$$

An analogous inequality is true for $\pi_{2,j}$ for every $j \in \{1,\dots,m\}$.

Each of the expressions $(x^i)^\top My$ and $x^\top Ny^j$ is essentially a sum of k independent random variables, where each variable takes value between 0 and 1 and has expected value $(x^i)^\top My^*$ and $(x^*)^\top Ny^j$, respectively. Applying the Hoeffding bound (Theorem 2.38), we get $\Pr[\overline{\psi_{1,i}}] \leq e^{-k\varepsilon^2/2}$ and $\Pr[\overline{\psi_{2,j}}] \leq e^{-k\varepsilon^2/2}$. Summing up the estimates obtained so far and using the choice of k and the fact $m \geq 2$, we derive from the union bound that

$$\Pr[\overline{GOOD}] \leq \Pr[\overline{\phi_1}] + \Pr[\overline{\phi_2}] + \sum_{i=1}^{m} \Pr[\overline{\pi_{1,i}}] + \sum_{j=1}^{m} \Pr[\overline{\pi_{2,j}}]$$

$$\leq 4e^{-k\varepsilon^2/8} + 4e^{-k\varepsilon^2/8} + m(e^{-k\varepsilon^2/2} + 4e^{-k\varepsilon^2/8}) + m(e^{-k\varepsilon^2/2} + 4e^{-k\varepsilon^2/8}) < 1.$$

Thus, $\Pr[GOOD] > 0$ and, as discussed before, we obtain the desired ε-Nash equilibrium (x,y). This finishes the proof of Claim 2.39. □

Note that the above proof actually produces a quasi-polynomial algorithm that finds all k-uniform ε-Nash equilibria of G.

2.5.2 Correlated equilibria

The following solution concept, introduced by [Aum74], might take some getting used to, but this is the most fundamental solution concept of all according to several researchers. For example, Myerson, a Nobel-prize-winning game theorist, said that "If there is intelligent life on other planets, in majority of them, they would have discovered correlated equilibrium before Nash equilibrium." [LBS08]. We first state the definition of correlated equilibria, which might look rather non-transparent at first sight, and then illustrate it with some examples.

Definition 2.40 (Correlated equilibrium). *For a normal-form game $G = (P, A, u)$ of n players, let p be a probability distribution on A, that is, $p(a) \geq 0$ for every $a \in A$ and $\sum_{a \in A} p(a) = 1$. The distribution p is a* correlated equilibrium[32] *in G if*

$$\sum_{a_{-i} \in A_{-i}} u_i(a_i; a_{-i})p(a_i; a_{-i}) \geq \sum_{a_{-i} \in A_{-i}} u_i(a_i'; a_{-i})p(a_i; a_{-i}) \tag{2.7}$$

for every player $i \in P$ and all pure strategies $a_i, a_i' \in A_i$.

---

[32] Korelované ekvilibrium.

Informally, a correlated equilibrium is a randomized assignment of (potentially correlated) action recommendations to players such that nobody wants to deviate. So a correlated equilibrium is a Nash equilibrium when the players receive payoff-irrelevant signals before playing the game. We can imagine that these recommendations are delivered using a trusted third party with the distribution p being publicly known. The trusted third party samples a strategy profile $a = (a_1, \ldots, a_n)$ according to p. Before playing the game, for each player $i \in P$, the trusted third party privately suggests the strategy a_i to i, but does not reveal a_{-i} to i. The player i can follow this suggestion, or not. At the time of decision-making, each player i knows the distribution p, one component a_i of the strategy profile a, and accordingly has a posterior distribution on suggested strategies a_{-i} of other players. The correlated equilibrium condition (2.7) then says that every player maximizes his expected utility by playing the suggested strategy a_i. The expectation is conditioned on i's information — p and a_i — and assumes that other players play their recommended strategies.

We illustrate the definition on two games, the Chicken game and the Battle of Sexes.

Example 2.41. *Consider the Game of chicken introduced in Example 2.12. Here we have two drivers, 1 and 2, driving towards each other in a collision course so one of them has to swerve or both die in the crash. This is captured by the following table.*

	Swerve	Straight
Swerve	(0,0)	(-1,1)
Straight	(1,-1)	(-10,-10)

There are two pure Nash equilibria with $(s_1(Swerve), s_2(Swerve)) = (1, 0)$ and $(s_1(Swerve), s_2(Swerve)) = (0, 1)$, and one fully mixed Nash equilibrium with $(s_1(Swerve), s_2(Swerve)) = (9/10, 9/10)$. The expected payoffs are $(-1, 1)$, $(1, -1)$, and $(-1/10, -1/10)$, respectively.

Now, assume there is a trusted third party, a traffic light. The traffic light chooses each of the three options $(Swerve, Swerve)$, $(Swerve, Straight)$, and $(Straight, Swerve)$ independently at random with probability $1/3$. The option $(Straight, Straight)$ is never chosen by the traffic light. No driver knows which action is recommended to the other driver, he only knows the distribution p.

We show that the traffic light constitutes a correlated equilibrium by proving that no driver wants to deviate from the traffic light's instruction. If the traffic light tells driver 1 to go straight, then the driver knows that the recommendation for the other driver is to swerve, so driver 1 will not deviate from the recommendation, as he assumes that driver 2 obeys his recommendation. Now, assume that driver 1 is told to swerve. Then he knows that the other driver is being told to swerve or to go straight, both options with equal probability. The expected payoff of driver 1 conditioned on the fact that he is told to swerve is $\frac{1}{2} \cdot 0 + \frac{1}{2} \cdot (-1) = -1/2$. If driver 1 decides to deviate and goes straight, then his expected payoff is only $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot (-10) = -9/2$. So driver 1 does not deviate also in this case. By symmetry, driver 2 does not deviate as well and thus we have a correlated equilibrium with the expected payoff for each driver $\frac{1}{3} \cdot 0 + \frac{1}{3} \cdot 1 + \frac{1}{3} \cdot (-1) = 0$. Thus, the reward was made better by correlation.

Example 2.42. *We recall the game Battles of sexes from Example 2.11. This game of two players can be described with the following table.*

	Football	Opera
Football	(2,1)	(0,0)
Opera	(0,0)	(1,2)

As we have shown, there are three Nash equilibria of this game: if the players 1 and 2 choose the actions Football with probability p_1 and Opera with probability p_2, respectively, where

$(p_1, p_2) \in \{(1,0),(0,1),(2/3,2/3)\}$. *The expected payoffs are* $(2,1)$, $(1,2)$, *and* $(2/3,2/3)$, *respectively.*

Now, imagine there is a trusted third party that flips a fair coin before the game and, based on the outcome of the coin toss, tells the players what they should do. For example, if the coin shows head, then both players are told to choose Football and, if the coin shows tails, both of them are told to choose Opera. Now, if player 1 is told to choose Opera, he knows that player 2 is also told to choose Opera, since both players know the distribution p. So, player 1 has no incentive to deviate and switch to Football as his payoff would be 0 compared to 1. Analogously, player 2 does not want to deviate so this distribution constitutes a correlated equilibrium. The advantage of this procedure is that the expected payoffs are now higher — $(3/2, 3/2)$.

Importantly, the distribution p in the definition of a correlated equilibrium does not have to be a product distribution; in this sense, the strategies chosen by the players are correlated. This is captured by the following proposition, which says that correlated equilibria are more general than Nash equilibria. Consequently, there is a hope that they are computationally tractable.

Proposition 2.43. *In every normal-form game $G = (P, A, u)$ of n players, for every Nash equilibrium there exists a corresponding correlated equilibrium.*

Proof. Let $s^* = (s_1^*, \ldots, s_n^*)$ be a strategy profile. Consider a function $p_{s^*}(a) = \prod_{j=1}^{n} s_j^*(a_j)$ for every $a = (a_1, \ldots, a_n) \in A$. Clearly, $p_{s^*}(a) \geq 0$ for every $a \in A$ and one can show by induction on n that $\sum_{a \in A} p_{s^*}(a) = 1$, so p_{s^*} is a probability distribution.

We show that if s^* is a Nash equilibrium, then p_{s^*} is a correlated equilibrium. Fix $i \in P$ and $a_i, a_i' \in A_i$. Assume $s_i^*(a_i) > 0$, that is, the pure strategy a_i is in the support of s_i^*. Using the definitions of u_i and p_{s^*}, we have

$$u_i(a_i'; s_{-i}^*) = \sum_{a_{-i} \in A_{-i}} u_i(a_i'; a_{-i}) \prod_{j=1, j \neq i}^{n} s_j^*(a_j) = \frac{1}{s_i^*(a_i)} \sum_{a_{-i} \in A_{-i}} u_i(a_i'; a_{-i}) p_{s^*}(a_i; a_{-i}).$$

Analogously, we see that

$$u_i(a_i; s_{-i}^*) = \sum_{a_{-i} \in A_{-i}} u_i(a_i; a_{-i}) \prod_{j=1, j \neq i}^{n} s_j^*(a_j) = \frac{1}{s_i^*(a_i)} \sum_{a_{-i} \in A_{-i}} u_i(a_i; a_{-i}) p_{s^*}(a_i; a_{-i}).$$

Assuming s^* is a Nash equilibrium, Observation 2.23 implies that the pure strategy a_i in the support of s_i^* is also a best response of player i to s_{-i}^*. Thus, $u_i(a_i; s_{-i}^*) \geq u_i(a_i'; s_{-i}^*)$ and we obtain

$$\frac{1}{s_i^*(a_i)} \sum_{a_{-i} \in A_{-i}} u_i(a_i; a_{-i}) p_{s^*}(a_i; a_{-i}) \geq \frac{1}{s_i^*(a_i)} \sum_{a_{-i} \in A_{-i}} u_i(a_i'; a_{-i}) p_{s^*}(a_i; a_{-i})$$

After multiplying by $s_i^*(a_i)$, this gives

$$\sum_{a_{-i} \in A_{-i}} u_i(a_i; a_{-i}) p_{s^*}(a_i; a_{-i}) \geq \sum_{a_{-i} \in A_{-i}} u_i(a_i'; a_{-i}) p_{s^*}(a_i; a_{-i}).$$

If $s_i^*(a_i) = 0$, then $p_{s^*}(a_i; a_{-i}) = 0$ for every $a_{-i} \in A_{-i}$ and the last inequality is trivially satisfied. Thus, p_{s^*} is a correlated equilibrium. $\square$

Therefore, using Nash's Theorem (Theorem 2.16), we see that correlated equilibria exist in every normal-form game. In fact, the following result implies that there is always exactly one correlated equilibrium or we have infinitely many of them.

Proposition 2.44. *In every normal-form game $G = (P, A, u)$, every convex combination of correlated equilibria is a correlated equilibrium.*

Proof. Assume p and p' are two correlated equilibria of G. We show that $p'' = tp + (1-t)p'$ is a correlated equilibrium as well. First, observe that p'' is a probability distribution. The rest follows immediately, as the fact that p and p' are correlated equilibria gives

$$\sum_{a_{-i} \in A_{-i}} u_i(a_i; a_{-i}) p''(a_i; a_{-i}) = \sum_{a_{-i} \in A_{-i}} u_i(a_i; a_{-i})(t \cdot p(a_i; a_{-i}) + (1-t) \cdot p'(a_i; a_{-i})) \geq$$

$$\sum_{a_{-i} \in A_{-i}} u_i(a_i'; a_{-i})(t \cdot p(a_i; a_{-i}) + (1-t) \cdot p'(a_i; a_{-i})) = \sum_{a_{-i} \in A_{-i}} u_i(a_i'; a_{-i}) p''(a_i; a_{-i})$$

for every player $i \in P$ and all pure strategies $a_i, a_i' \in A_i$. $\qquad\square$

Thus, the set of correlated equilibrium distributions is convex and it can be shown that it is also compact. Moreover, this set is also a set of solutions of some system of linear inequalities. This is much easier to solve than finding a Nash equilibrium. Consider the following linear program

$$\max \left\{ \sum_{i \in P} \left(\sum_{a \in A} u_i(a) p(a) \right) \right\} \text{ subject to}$$

$$\sum_{a_{-i} \in A_{-i}} u_i(a_i; a_{-i}) p(a_i; a_{-i}) \geq \sum_{a_{-i} \in A_{-i}} u_i(a_i'; a_{-i}) p(a_i; a_{-i}) \text{ for all } i \in P, a_i, a_i' \in A_i,$$

$$p(a) \geq 0 \text{ for every } a \in A,$$

$$\sum_{a \in A} p(a) = 1,$$

where variables $(p(a))_{a \in A}$ represent the probabilities of selecting a with p. Any feasible solution of this linear program gives a correlated equilibrium. An optimal solution of this linear program will give a correlated equilibrium that maximizes the social welfare $\sum_{i \in P} \left(\sum_{a \in A} u_i(a) p(a) \right)$ and can be found in polynomial time.

Note, however, that the size of this linear program is polynomial only with respect to the payoff matrices and can be, for example, exponential with respect to the number of players. If we settle for approximate correlated equilibria, we do not need the heavy machinery of linear programming, but, as we will see in the next section, we can use a simpler method based on suitably chosen learning algorithms.

2.5.3 Exercises

Exercise 2.20. *Let $G = (P = \{1, 2\}, A, u)$ be a normal-form game of two players with $A_1 = \{U, D\}$ and $A_2 = \{L, R\}$ with payoff function u depicted in Table 2.10.*

	L	R
U	(1,1)	(0,0)
D	$(1 + \frac{\varepsilon}{2}, 1)$	(500,500)

Table 2.10: The game from Exercise 2.20.

Show that there is an ε-Nash equilibrium s of G such that $u_i(s') > 10u_i(s)$ for every $i \in P$ and every Nash equilibrium s' of G. In other words, there might be games where some ε-Nash equilibria are far away from any Nash equilibrium. **[3]**

Exercise 2.21. *Compute all correlated equilibria in Prisoner's dilemma.* **[2]**

Exercise 2.22. *Consider the following modified Rock-Paper-Scissors game where players get a penalty in case of a tie.*

	T	S
T	(-2,-2)	(0,-3)
S	(-3,0)	(-1,-1)

Table 2.11: The game from Exercise 2.21

	Rock	Paper	Scissors
Rock	(-1,-1)	(0,1)	(1,0)
Paper	(1,0)	(-1,-1)	(0,1)
Scissors	(0,1)	(1,0)	(-1,-1)

Table 2.12: A modified game Rock-Paper-Scissors in Exercise 2.22.

(a) Show that this game has a unique Nash equilibrium that results in expected payoffs of 0 to both players. **[2]**

(b) Show that the following probability distribution is a correlated equilibrium in which the players obtain expected payoffs of $1/2$. **[2]**

	Rock	Paper	Scissors
Rock	0	1/6	1/6
Paper	1/6	0	1/6
Scissors	1/6	1/6	0

Table 2.13: A probability distribution in the modified game Rock-Paper-Scissors from Exercise 2.22.

Exercise 2.23. *Let $G = (P = \{1, 2\}, A, u)$ be a normal-form game of two players with $A_1 = \{U, D\}$ and $A_2 = \{L, R\}$ with payoff function u depicted in Table 2.14.*

(a) Compute all Nash equilibria of G and draw the convex hull of Nash equilibrium payoffs. **[2]**

(b) Is there any correlated equilibrium of G (for some distribution p) that yields payoffs outside this convex hull? **[3]**

Exercise 2.24. *Let $G = (P = \{1, 2\}, A, u)$ be a normal-form game of two players with $A_1 = \{U, D\}$ and $A_2 = \{L, R\}$ with payoff function u depicted in Table 2.15.*

(a) Compute all Nash equilibria of G and draw the convex hull of Nash equilibrium payoffs. **[2]**

(b) Determine the set of all correlated equilibria of G. **[3]**

2.6 Regret minimization

In this section, we explore the concept of online learning and we show some applications in game theory, in particular, we show a connection to some variants of Nash equilibria. Learning is the ability to improve performance by observing data. Consider, an agent A in an unknown adversary environment who has a task to learn the setup in the environment. The agent A may be any forecaster or algorithm. Learning problems can also be thought of as a repeated game

	L	R
U	(6,6)	(2,7)
D	(7,2)	(0,0)

Table 2.14: The game from Exercise 2.23.

	L	R
U	(4,4)	(1,5)
D	(5,1)	(0,0)

Table 2.15: The game from Exercise 2.24.

between the agent and the environment. For example, the weather might be the environment, prediction is the task, and the weather prediction office is the agent.

Formally, we consider the following model. There are N available actions $X = \{1, \ldots, N\}$ and at each time step t the online algorithm A selects a probability distribution $p^t = (p_1^t, \ldots, p_N^t)$ over X. That is, $p_i^t \geq 0$ for every $i \in X$ and $\sum_{i=1}^{N} p_i^t = 1$. After the distribution p^t is chosen at time step t, the adversary chooses a loss vector $\ell^t = (\ell_1^t, \ldots, \ell_N^t) \in [-1, 1]^N$, where the number ℓ_i^t is the loss of action i in time t. The algorithm A then receives the vector ℓ^t and experiences loss $\ell_A^t = \sum_{i=1}^{N} p_i^t \ell_i^t$. This is the expected loss of A at time step t when the algorithm A selects action i with probability p_i^t. After T steps, the loss of action i is $L_i^T = \sum_{t=1}^{T} \ell_i^t$ and the loss of A is $L_A^T = \sum_{t=1}^{T} \ell_A^t$.

2.6.1 External regret

Let $\mathcal{A}$ be a given class of algorithms, called *comparison class*, and let T be a given positive integer. In the *external regret setting*, our aim is to design an online algorithm A whose loss is close to the loss of the best algorithm from $\mathcal{A}$, that is, L_A^T should be close to $L_{\mathcal{A},min}^T = \min_{B \in \mathcal{A}} L_B^T$. In other words, we want to minimize the *external regret* $R_{A,\mathcal{A}}^T = L_A^T - L_{\mathcal{A},min}^T$ of A. In this section, we mostly consider only one of the most studied comparison classes, the class consisting of single actions, that is, $\mathcal{A}_X = \{A_i : i \in X\}$, where the algorithm A_i always chooses action i. For this class $\mathcal{A}_X$, we simply write $L_{min}^T = L_{\mathcal{A}_X,min}^T$ and $R_A^T = R_{A,\mathcal{A}_X}^T$.

Until specified otherwise, we now consider only loss vectors with entries from $\{0, 1\}$, not with real entries from $[-1, 1]$. This is only to simplify the notation, all presented results can be extended to the general case.

First, we show that one cannot guarantee low external regret with respect to the optimal sequence of decisions in hindsight. Let $\mathcal{A}_{all}$ be the set of all online algorithms that assign probability 1 to an arbitrary action from X in every step (unlike the more special class $\mathcal{A}_X$, where every algorithm selects the same action in all steps).

Observation 2.45. *For any online algorithm A and every $T \in \mathbb{N}$, there is a sequence of T loss vectors such that the external regret $R_{A,\mathcal{A}_{all}}^T$ of A is at least $T(1 - 1/N)$.*

Proof. For every $t \in \{1, \ldots, T\}$, we choose the loss vector ℓ^t by choosing an action $i_t \in X$ with lowest probability p_i^t and we set $\ell_{i_t}^t = 0$ and $\ell_i^t = 1$ for every $i \neq i_t$. Since $p_{i_t}^t \leq 1/N$, we have $\ell_A^t \geq 1 - 1/N$ for every $t \in \{1, \ldots, T\}$ and thus the loss L_A^T of A after T steps is at least $T(1 - 1/N)$.

On the other hand, the algorithm $B \in \mathcal{A}_{all}$ with B that selects the action i_t in step t with probability 1 has the total loss $L_B^T = 0$. Therefore $R_{A,\mathcal{A}_{all}}^T \geq L_A^T - L_B^T \geq T(1 - 1/N)$. $\square$

Thus, considering all possible algorithms leads to a very large regret. This is why we restrict

ourselves to special classes of algorithms, namely, to the class $\mathcal{A}_X$, which corresponds to comparing the online algorithm to the best single action.

As our first attempt to construct an online algorithm with small external regret, we try a greedy approach. In step t, the *Greedy algorithm* selects an action $i \in X$ for which the cumulative loss L_i^{t-1} at step $t-1$ is the smallest (the ties are broken by choosing minimum action).

Algorithm 2.6.1: GREEDY ALGORITHM(X, T)

$Input$: A set of actions $X = \{1, \ldots, N\}$ and a number of steps $T \in \mathbb{N}$.
$Output$: A probability distribution p^t for every $t \in \{1, \ldots, T\}$.
$p^1 \leftarrow (1, 0, \ldots, 0)$,
for $t = 2, \ldots, T$
$\quad$**do** $\begin{cases} L_{min}^{t-1} \leftarrow \min_{j \in X}\{L_j^{t-1}\}, \\ S^{t-1} \leftarrow \{i \in X \colon L_i^{t-1} = L_{min}^{t-1}\}, \\ k \leftarrow \min S^{t-1}, \\ p_k^t \leftarrow 1, p_i^t \leftarrow 0 \text{ for } i \neq k, \end{cases}$
Output $\{p^t \colon t \in \{1, \ldots, T\}\}$.

Proposition 2.46. *The cumulative loss L_{Greedy}^T of the Greedy algorithm at time $T \in \mathbb{N}$ satisfies*

$$L_{\text{Greedy}}^T \leq N \cdot L_{min}^T + (N - 1)$$

for any sequence of $\{0, 1\}$-valued loss vectors.

Proof. At time step t, if the Greedy algorithm incurs a loss (of 1) and L_{min}^t does not increase, then at least one action disappears from S^t in the next step. This can occur at most N times and then L_{min}^t increases by 1. In other words, the Greedy algorithm incurs a loss of at most N between successive increments of L_{min}^t by 1. It follows that

$$L_{\text{Greedy}}^T \leq N \cdot L_{min}^T + N - |S^T| \leq N \cdot L_{min}^T + (N - 1). \qquad \square$$

The bound obtained in Proposition 2.46 is rather weak since it says that the cumulative loss of the Greedy algorithm is roughly at most N times larger than the cumulative loss L_{min}^T of the best action. There is a reason for this poor behavior, as the following result shows that no deterministic algorithm can perform well. Here, an algorithm is *deterministic* if, for every step t, there is action i with $p_i^t = 1$.

Proposition 2.47. *For every deterministic algorithm D and $T \in \mathbb{N}$, there is a sequence of loss vectors such that $L_D^T = T$ and $L_{min}^T \leq \lfloor T/N \rfloor$. Thus, in particular, $L_D^T \geq N \cdot L_{min}^T + (T \bmod N)$.*

Proof. For every time step t, let i_t be the action selected by D, that is, the action with $p_{i_t}^t = 1$. We choose the loss vector ℓ^t such that $\ell_{i_t}^t = 1$ and such that all other entries of ℓ^t are zero. Then the cumulative loss L_D^T of D in time T is exactly T.

There are N actions in total and thus, by the pigeonhole principle, there is an action i that has been selected by D at most $\lfloor T/N \rfloor$ times. By the construction of ℓ^t, only actions selected by D can have a loss. Thus, $L_{min}^T \leq \lfloor T/N \rfloor$. $\qquad \square$

In view of Proposition 2.47, it makes perfect sense to consider randomized algorithms in order to minimize external regret. A natural approach how to randomize the Greedy algorithm is to break ties at random, not deterministically. The hope is that then the algorithm splits weights between the currently best actions, which might produce better results. More specifically, we consider a *Randomized greedy algorithm* that is obtained from the Greedy algorithm by assigning a uniform distribution to actions with the minimum cumulative loss so far.

Algorithm 2.6.2: RANDOMIZED GREEDY ALGORITHM(X, T)

$Input$: A set of actions $X = \{1, \ldots, N\}$ and a number of steps $T \in \mathbb{N}$.
$Output$: A probability distribution p^t for every $t \in \{1, \ldots, T\}$.
$p^1 \leftarrow (1/N, \ldots, 1/N)$,
for $t = 2, \ldots, T$
$\quad$ **do** $\begin{cases} L_{min}^{t-1} \leftarrow \min_{j \in X}\{L_j^{t-1}\}, \\ S^{t-1} \leftarrow \{i \in X \colon L_i^{t-1} = L_{min}^{t-1}\}, \\ p_i^t \leftarrow 1/|S^{t-1}| \text{ for every } i \in S^{t-1} \text{ and } p_i^t \leftarrow 0 \text{ otherwise.} \end{cases}$
Output $\{p^t \colon t \in \{1, \ldots, T\}\}$.

Proposition 2.48. *The cumulative loss L_{RG}^T of the Randomized greedy algorithm at time $T \in \mathbb{N}$ satisfies*

$$L_{\mathrm{RG}}^T \leq (1 + \ln N) \cdot L_{min}^T + \ln N$$

for any sequence of $\{0, 1\}$-valued loss vectors.

Proof. Similarly as in the proof of Proposition 2.46, we estimate the maximum loss incurred by the algorithm between successive increases of L_{min}^t. This time, we estimate this loss from above by $1 + \ln N$ instead of N. For $j \in \mathbb{N}$, let t_j be the time step t at which the loss L_{min}^t first reaches value j. We estimate the loss of the Randomized greedy algorithm between steps t_j and t_{j+1}.

Note that $1 \leq |S^t| \leq N$ for every step $t \in \{1, \ldots, T\}$. If the size of S^t shrinks by k from n' to $n' - k$ at some time $t \in (t_j, t_{j+1}]$, then the loss of the Randomized greedy algorithm at step t is k/n', since the weight of each such action is $1/n'$. Clearly, $k/n' \leq 1/n' + 1/(n'-1) + \cdots + 1/(n'-k+1)$, so we obtain that the loss for the entire time interval $(t_j, t_{j+1}]$ is at most

$$1/N + 1/(N-1) + \cdots + 1/1 \leq 1 + \ln N.$$

It follows that

$$L_{\mathrm{RG}}^T \leq (1 + \ln N) \cdot L_{min}^T + (1/N + 1/(N-1) + \cdots + 1/(|S^T| + 1)). \qquad \square$$

Thus, the Randomized greedy algorithm performs much better than the Greedy algorithm. Still, the bound obtained for the Randomized greedy algorithm is a logarithmic factor of N apart from L_{min}^T. By analyzing the proof of Proposition 2.48, we see that the losses are greatest when the sets S^t are small since the loss can be viewed as proportional to $1/|S^t|$. A natural idea how to overcome this issue is to assign weights to actions such that the actions that are close to the best have large weights. This is captured by the following algorithm that is again a modification of the Greedy algorithm. In the analysis of this algorithm, we consider the general case with losses ℓ_i^t in $[-1, 1]$, not only in $\{0, 1\}$.

The *Polynomial weights algorithm* receives the set of actions X, the number of steps T, and a parameter $\eta \in (0, 1/2)$ on the input. The algorithm is initialized by setting the probabilities p_i^1 to $1/N$ and weights w_i^1 to 1 for every action $i \in X$. In time step $t \geq 2$, we assign to each action $i \in X$ the weight $w_i^t = w_i^{t-1}(1 - \eta \ell_i^{t-1})$ and probability $p_i^t = w_i^t / \sum_{j=1}^N w_j^t$.

Algorithm 2.6.3: Polynomial weights algorithm(X, T, η)

$Input$: A set of actions $X = \{1, \ldots, N\}$, $T \in \mathbb{N}$, and $\eta \in (0, 1/2]$.
$Output$: A probability distribution p^t for every time step $t \in \{1, \ldots, T\}$.
$w_i^1 \leftarrow 1$ for every $i \in X$,
$p^1 \leftarrow (1/N, \ldots, 1/N)$,
for $t = 2, \ldots, T$
    do $\begin{cases} w_i^t \leftarrow w_i^{t-1}(1 - \eta \ell_i^{t-1}), \\ W^t \leftarrow \sum_{i \in X} w_i^t, \\ p_i^t \leftarrow w_i^t / W^t \text{ for every } i \in X. \end{cases}$
Output $\{p^t : t \in \{1, \ldots, T\}\}$.

Theorem 2.49. *For $\eta \in (0, 1/2]$, every sequence of $[-1, 1]$-valued loss vectors, and every $k \in X$, the cumulative loss L_{PW}^T of the Polynomial weights algorithm at time $T \in \mathbb{N}$ satisfies*

$$L_{\mathrm{PW}}^T \leq L_k^T + \eta Q_k^T + \ln N / \eta,$$

where $Q_k^T = \sum_{t=1}^T (\ell_k^t)^2$. In particular, if $T \geq 4 \ln N$, then by setting $\eta = \sqrt{\ln N / T}$ and noting that $Q_k^T \leq T$, we obtain

$$L_{\mathrm{PW}}^T \leq L_{min}^T + 2\sqrt{T \ln N}.$$

Proof. We first show that if the algorithm has a significant loss, then the total weight W^t must drop substantially. For a time step t, we have $\ell_{PW}^t = \sum_{i=1}^N w_i^t \ell_i^t / W^t$, that is, ℓ_{PW}^t is the expected loss of the Polynomial weights algorithm at step t. The weight w_i^t of every action i is multiplied by $(1 - \eta \ell_i^{t-1})$ at step t. Thus, $W^{t+1} = W^t - \sum_{i=1}^N \eta w_i^t \ell_i^t = W^t(1 - \eta \ell_{PW}^t)$. In other words, the proportion of the total weight W^t removed at step t is exactly η-fraction of the expected loss ℓ_{PW}^t of the algorithm. Using $W^1 = N$, we obtain

$$W^{T+1} = W^1 \prod_{t=1}^T (1 - \eta \ell_{PW}^t) = N \prod_{t=1}^T (1 - \eta \ell_{PW}^t).$$

Using the inequality $1 - z \leq e^{-z}$ for every $z \in \mathbb{R}$, we obtain the estimate

$$W^{T+1} \leq N \prod_{t=1}^T e^{-\eta \ell_{PW}^t} = N e^{-\eta \sum_{t=1}^T \ell_{PW}^t}.$$

Taking the logarithms, we obtain

$$\ln W^{T+1} \leq \ln N - \eta \sum_{t=1}^T \ell_{PW}^t = \ln N - \eta L_{\mathrm{PW}}^T,$$

where the equality follows from the definition of ℓ_{PW}^t.

    For the lower bound on W^{T+1}, we have $W^{T+1} \geq w_k^{T+1}$ and thus, by taking logarithms and using the recursive definition of weights, we obtain

$$\ln W^{T+1} \geq \ln w_k^{T+1} = \sum_{t=1}^T \ln(1 - \eta \ell_k^t).$$

Using the inequality $\ln(1 - z) \geq -z - z^2$ for $z \leq 1/2$, we bound this expression from below by

$$-\eta L_k^T - \eta^2 Q_k^T.$$

Combining the lower and the upper bound on W^{T+1}, we have

$$-\eta L_k^T - \eta^2 Q_k^t \leq \ln N - \eta L_{\mathrm{PW}}^T,$$

which finishes the proof.   □

We assume that the number of steps T is known in advance in the second part of the statement. If it is not, then a "guess and double" approach can be used, which results with a bound that is worse by only a constant factor; see Exercise 2.25. A potentially better bound $L_{\text{PW}}^T \leq L_{min}^T + 2\sqrt{\ln N / L_{min}^T}$ can be achieved by choosing $\eta = \min\{\sqrt{\ln N / L_{min}^T}, 1/2\}$.

To answer the natural question of whether the upper bound on the external regret from Theorem 2.49 can be improved, we show that the bound is close to optimal. First, we prove that it is not possible to get an $o(T)$-bound on the regret when T is small when compared to $\log_2 N$.

Proposition 2.50. *For integers N and T with $T < \lfloor \log_2 N \rfloor$, there exists a stochastic generation of losses such that, for every online algorithm A, we have $\mathbb{E}[L_A^T] \geq T/2$ and yet $L_{min}^T = 0$.*

Proof. First, we assume that N is a power of two. We consider the following sequence of loss vectors. At time step 1, a random subset of $N/2$ actions gets a loss of 0 and the rest gets a loss of 1. At time step $t \geq 2$, a random subset of half of the actions that have received loss 0 so far gets loss 0 and all remaining actions get loss of 1.

First, since $T < \log_2 N$, the number of actions with total loss zero is at least

$$N - \sum_{t=1}^{T} N/2^t = N(1 - 1 + 2^{-T}) = N \cdot 2^{-T} > 1.$$

Thus, there is at least one action i with the total loss $L_i^T = 0$ and we have $L_{min}^T = 0$.

Every online algorithm A incurs the expected loss of at least $1/2$ in every time step t, since the expected fraction of probability mass p_i^t on actions that receive a loss of 0 is at most $1/2$. Consequently, $L_A^T \geq T/2$.

If N is not a power of two, we restrict ourselves to a set of $2^{\lfloor \log_2 N \rfloor}$ actions during the entire process, setting the losses of the remaining action to 1 at every time step. The proof above then gives the statement. $\qquad\square$

As a second lower bound, we show that we cannot achieve $o(\sqrt{T})$ regret even when $N = 2$.

Proposition 2.51. *In the case of $N = 2$ actions, there exists a stochastic generation of losses such that, for every online algorithm A, we have $\mathbb{E}[L_A^T - L_{min}^T] \geq \Omega(\sqrt{T})$.*

Proof. Let $e_1 = (1,0)$ and $e_2 = (0,1)$. At each time step $t \in \{1, \ldots, T\}$, we choose $\ell^t = e_1$ with probability $1/2$ and $\ell^t = e_2$ with probability $1/2$. Then, for every distribution p^t, the expected loss at time step t is exactly $1/2$. Thus, every online algorithm A has the expected loss of $T/2$.

Let $X \in \mathbb{Z}$ be the random variable denoting the number of e_1 losses minus $T/2$. This means that the number of e_2 losses minus $T/2$ is $-X$. Then $T/2 - L_{min}^T = |X|$ and it remains to estimate $\mathbb{E}[|X|]$. The probability $\Pr[X = m]$ is

$$\binom{T}{T/2 + m}/2^T \leq O(1/\sqrt{T}),$$

where we used Stirling's approximation $k! = (1 + o(1))\sqrt{2\pi k}(k/e)^k$. By the Union bound, $\Pr[|X| \leq c\sqrt{T}] < 1 - c'$ for some constants $c, c' > 0$. In particular, we have $T/2 - L_{min}^T \geq \Omega(\sqrt{T})$ with constant probability and thus $\mathbb{E}[|X|] = \Omega(\sqrt{T})$. The rest of the statement follows from the linearity of the expected value. $\qquad\square$

2.6.2 No-regret dynamics

In this section, we present a method of how to use regret minimization in the theory of normal-form games. Let $G = (P, A, C)$ be a normal-form game of n players. It is more natural in this setting to work with a *cost function* C instead of the utility function u. The setting is the same, that is, $C = (C_1, \ldots, C_n)$, where $C_i \colon A \to [-1, 1]$. The only difference is that we now want to minimize the cost function (not maximize it as in the case of the utility function).

The game theoretic model for regret minimization is then as follows. Each player $i \in P$ plays G T-times using an online algorithm ON_i that has probability distribution p_i^t on A_i

in every step $t = 1, \ldots, T$ while the other players play the joint distribution p^t_{-i}. In other words, p^t_i is the mixed strategy vector of player i in step t. We let $\ell^t_{ON_i} = \mathbb{E}_{a^t \sim p^t}[C_i(a^t)]$ be the loss of player i in step t. The cumulative loss of player i is then $L^T_{ON_i} = \sum^T_{t=1} \ell^t_{ON_i}$. For $a^t_j \in A_i$, we also have the loss $\mathbb{E}_{a^t_{-i} \sim p^t_{-i}}[C_i(a^t_j; a^t_{-i})]$ of player i if he played the action a^t_j at step t. The corresponding cumulative loss is then $L^T_{ON_i,j} = \sum^T_{t=1} \mathbb{E}_{a^t_{-i} \sim p^t_{-i}}[C_i(a^t_j; a^t_{-i})]$ and we use $L^T_{ON_i,min}$ to denote $\min_{a_j \in A_i} L^T_{ON_i,j}$.

We consider the following procedure, called *No-regret dynamics*, that uses algorithms with small external regret to generate mixed strategies in a normal-form game $G = (P, A, C)$ of n players. We have T time steps and we proceed as follows in each time step t. First, each player independently chooses a mixed strategy $p^t_i = (p^t_i(a_i))_{a_i \in A_i}$ using some algorithm with small external regret such that actions correspond to pure strategies. Each player $i \in P$ then receives a loss vector $\ell^t_i = (\ell^t_i(a_i))_{a_i \in A_i}$, where

$$\ell^t_i(a_i) = \mathbb{E}_{a^t_{-i} \sim p^t_{-i}}[C_i(a_i; a^t_{-i})]$$

for the product distribution $p^t_{-i} = \prod_{j \neq i} p^t_j$. That is, $\ell^t_i(a_i)$ is the expected cost of the pure strategy a_i given the mixed strategies chosen by the other players.

Algorithm 2.6.4: NO-REGRET DYNAMICS(G, T, ε)

$Input$: A normal-form game $G = (P, A, C)$ of n players, $T \in \mathbb{N}$, and $\varepsilon > 0$.
$Output$: A probability distribution p^t_i on A_i for each $i \in P$ and $t \in \{1, \ldots, T\}$.
for every step $t = 1, \ldots, T$

do $\begin{cases} \text{Each player } i \in P \text{ independently chooses a mixed strategy } p^t_i \text{ using} \\ \text{an algorithm with average regret at most } \varepsilon, \text{ with actions} \\ \text{corresponding to pure strategies.} \\ \text{Each player } i \in P \text{ receives a loss vector } \ell^t_i = (\ell^t_i(a_i))_{a_i \in A_i}, \text{ where} \\ \ell^t_i(a_i) \leftarrow \mathbb{E}_{a^t_{-i} \sim p^t_{-i}}[C_i(a_i; a^t_{-i})] \text{ for the product distribution} \\ p^t_{-i} = \prod_{j \neq i} p^t_j. \end{cases}$

Output $\{p^t \colon t \in \{1, \ldots, T\}\}$.

For example, if every player uses the Polynomial weights algorithm, then in every time step each player simply updates the weight of each of his pure strategies. Assuming that m is the maximum number of pure strategies of each player and $C_i(s) \in [-c_{max}, c_{max}]$ for every player $i \in P$ and every strategy profile s, it is sufficient to choose $T \geq 4c^2_{max} \ln m / \varepsilon^2$.

2.6.3 New proof of the Minimax Theorem

We recall that a normal-form game $G = (\{1, 2\}, A, C)$ of two players 1 and 2 is zero-sum if the loss of one player equals the gain of the other player, that is, $C_1(a) + C_2(a) = 0$ for every action profile $a \in A$.

Theorem 2.52. *Let $G = (P = \{1, 2\}, A, C)$ be a zero-sum game with the value of the game equal to v; see Theorem 2.21 for the definition. If player $i \in \{1, 2\}$ plays G using the online algorithm ON_i with external regret R, then his cumulative loss $L^T_{ON_i}$ satisfies*

$$L^T_{ON_i} \leq Tv + R.$$

Proof. By symmetry, it suffices to prove the statement for player 1. Let $p^t_{2,j}$ be the probability that player 2 chooses the jth action from A_2 in step t. We then define the mixed strategy vector corresponding to the observed frequencies of the actions player 2 has played as $q = (q_1, \ldots, q_{|A_2|})$, where $q_j = \sum^T_{t=1} p^t_{2,j}/T$. We already know from Section 2.3 that, for any mixed strategy vector q of player 2, player 1 has some action a_i such that $\mathbb{E}_{b_2 \sim q}[C_1(a_i, b_2)] \leq v$. Thus, if player 1 always plays action a_i, we obtain $L^T_{ON_1,min} \leq L^T_{ON_1,i} \leq vT$. Now, it follows from the assumption that ON_1 has external regret R that $L^T_{ON_1} \leq L^T_{ON_1,min} + R \leq vT + R$. $\qquad \square$

In particular, using a procedure with external regret $O(\sqrt{T \log N})$ (for example, using the Polynomial weights algorithm from Subsection 2.6.1), Theorem 2.52 implies that the average cumulative loss $L_{ON_i}^T/T$ of size at most $v + O(\sqrt{\log N/T})$. By choosing $T = T(N)$ large enough, we can have the loss of size at most $v + \varepsilon$ for any given $\varepsilon > 0$.

Using the above setting, we now present another proof of the Minimax theorem (Theorem 2.21). We recall that each zero-sum game $G = (P = \{1, 2\}, A, C)$ can be represented with an $m \times n$ matrix $M = (m_{i,j})$, where $A_1 = \{a_1, \ldots, a_m\}$, $A_2 = \{b_1, \ldots, b_n\}$, and $m_{i,j} = -C_1(a_i, b_j) = C_2(a_i, b_j)$. With this representation, the expected cost $C_2(s)$ for player 2 and strategy profile $s = (s_1, s_2)$ equals $x^\top M y$, where x and y are the mixed strategy vectors for s_1 and s_2, respectively. The Minimax theorem (Theorem 2.21) then states

$$\max_{x \in S_1} \min_{y \in S_2} x^\top M y = \min_{y \in S_2} \max_{x \in S_1} x^\top M y.$$

That is, it does not matter which player commits to his mixed strategy first, under optimal play, the expected payoff of each player is the same in the two scenarios.

Another proof of the Minimax theorem (Theorem 2.21). First, the inequality $\max_x \min_y x^\top M y \leq \min_y \max_x x^\top M y$ follows easily, since if x^* is an optimal strategy for player 1 if he plays first, he can always choose x^* when he plays second. In other words, it is only worse to go first.

To prove the other inequality $\max_x \min_y x^\top M y \geq \min_y \max_x x^\top M y$, we assume that all payoffs $m_{i,j}$ are between -1 and 1, otherwise we can rescale them. We choose a parameter $\varepsilon \in (0, 1]$ and run a no-regret dynamics for a sufficient number T of time steps so that both players have expected external regret at most ε. For example, if we run the Polynomial weights algorithm from Subsection 2.6.1, then it is sufficient to choose $T \geq 4 \ln (\max\{m, n\})/\varepsilon^2$, since the regret per time step is $2\sqrt{\ln (\max\{m, n\})/T}$ for this algorithm.

Let $p^1, \ldots, p^T$ and $q^1, \ldots, q^T$ be the mixed strategies played by players 1 and 2, respectively, as advised by the algorithm. The payoff vector revealed to each no-regret algorithm after iteration t is the expected payoff of each strategy, given the mixed strategy played by the other player in iteration t. This translates to the payoff vectors Mq^t and $-(p^t)^\top M$ for players 1 and 2, respectively. We let $\overline{x} = \frac{1}{T} \sum_{t=1}^T p^t$ be the time-averaged mixed strategy of player 1. Similarly, we let $\overline{y} = \frac{1}{T} \sum_{t=1}^T q^t$ be the time-averaged mixed strategy of player 2. The time-averaged expected payoff v of player 1 is then $\frac{1}{T} \sum_{t=1}^T (p^t)^\top M q^t$.

For every $i \in \{1, \ldots, m\}$, let e_i be the mixed strategy vector for the pure strategy a_i, that is, the ith coordinate of e_i is 1 and all other coordinates of e_i are 0. Since the external regret of player 1 is at most ε, we have

$$e_i^\top M \overline{y} = \frac{1}{T} \sum_{t=1}^T e_i^\top M q^t \leq \frac{1}{T} \sum_{t=1}^T (p^t)^\top M q^t + \varepsilon = v + \varepsilon$$

for every $i \in \{1, \ldots, m\}$. Since every mixed strategy x of player 1 is a convex combination of the pure strategies e_i, the linearity of expectation gives $x^\top M \overline{y} \leq v + \varepsilon$ for every $x \in S_1$. For player 2, a symmetric argument that uses the fact that the regret of player 2 is also at most ε gives $(\overline{x})^\top M y \geq v - \varepsilon$ for every $y \in S_2$.

Putting these facts together, we obtain

$$\max_{x \in S_1} \min_{y \in S_2} x^\top M y \geq \min_{y \in S_2} (\overline{x})^\top M y \geq v - \varepsilon$$
$$\geq \max_{x \in S_1} x^\top M \overline{y} - 2\varepsilon \geq \min_{y \in S_2} \max_{x \in S_1} x^\top M y - 2\varepsilon.$$

For $T \to \infty$, we get $\varepsilon \to 0$ and thus we obtain the desired inequality. $\square$

2.6.4 Coarse correlated equilibria

We now show a close connection between regret minimization and correlated equilibria. In Subsection 2.5.2, we introduced correlated equilibria, which generalize the concept of Nash

equilibrium and, unlike Nash equilibria, can be found efficiently using linear programming. Here we generalize Nash equilibria even further by defining so-called coarse correlated equilibria, which are even easier to compute than correlated equilibria. we show that in order to find a coarse correlated equilibrium of a normal-form game, it is sufficient to use one of the simple regret-minimizing algorithms.

Let $G = (P, A, C)$ be a normal-form game of n players. We recall that a probability distribution p on A is a correlated equilibrium in G if

$$\sum_{a_{-i} \in A_{-i}} C_i(a_i; a_{-i}) p(a_i; a_{-i}) \leq \sum_{a_{-i} \in A_{-i}} C_i(a_i'; a_{-i}) p(a_i; a_{-i})$$

for every player $i \in P$ and all pure strategies $a_i, a_i' \in A_i$. That is, for every player, it is a best response to play the suggested action by a correlating device, provided that the other players also do not deviate. Thus, one may express the correlated equilibrium condition as

$$\mathbb{E}_{a \sim p}[C_i(a) \mid a_i] \leq \mathbb{E}_{a \sim p}[C_i(a_i'; a_{-i}) \mid a_i]$$

for every $i \in P$ and all $a_i, a_i' \in A_i$.

We would like to enlarge the set of correlated equilibria to obtain an even more tractable concept. This is done by considering equilibrium that protects against unconditional unilateral deviations, as opposed to the conditional unilateral deviations addressed in the definition of correlated equilibria.

Definition 2.53 (Coarse correlated equilibrium). *For a normal-form game $G = (P, A, C)$ of n players, a probability distribution p on A is a* coarse correlated equilibrium *in G if*

$$\sum_{a \in A} C_i(a) p(a) \leq \sum_{a \in A} C_i(a_i'; a_{-i}) p(a)$$

for every player $i \in P$ and every $a_i' \in A_i$.

In terms of expected value, a coarse correlated equilibrium can be expressed as

$$\mathbb{E}_{a \sim p}[C_i(a)] \leq \mathbb{E}_{a \sim p}[C_i(a_i'; a_{-i})]$$

for every $i \in P$ and each $a_i' \in A_i$. The difference between a coarse correlated equilibrium and a correlated equilibrium is that a coarse correlated equilibrium only requires that following your suggested action a_i when a is drawn from p is only a best response in expectation before you see a_i. This makes sense if you have to commit to following your suggested action or not upfront, and do not have the opportunity to deviate after seeing it.

It follows from this definition that every correlated equilibrium is a coarse correlated equilibrium; see Exercise 2.27. Thus, in particular, every normal-form game has a coarse correlated equilibrium and we can find one in polynomial time using linear programming. The hierarchy of some of the discussed equilibrium notions is depicted in Figure 2.4.

We show that strategy profiles in regret-minimizing algorithms converge to coarse correlated equilibria. First, we need to define an approximate version of coarse correlated equilibria. For $\varepsilon \geq 0$, an ε-*coarse correlated equilibrium* in G is a probability distribution p on A such that

$$\sum_{a \in A} C_i(a) p(a) \leq \left(\sum_{a \in A} C_i(a_i'; a_{-i}) p(a) \right) + \varepsilon$$

for every player $i \in P$ and every $a_i' \in A_i$. In other words, p must satisfy $\mathbb{E}_{a \sim p}[C_i(a)] \leq \mathbb{E}_{a \sim p}[C_i(a_i'; a_{-i})] + \varepsilon$. Note that, for $\varepsilon = 0$, we obtain exactly coarse correlated equilibria.

The next result shows that the time-averaged history of joint play under no-regret dynamics converges to the set of coarse correlated equilibria. In particular, it implies that we can use natural learning dynamics to find ε-coarse correlated equilibria.
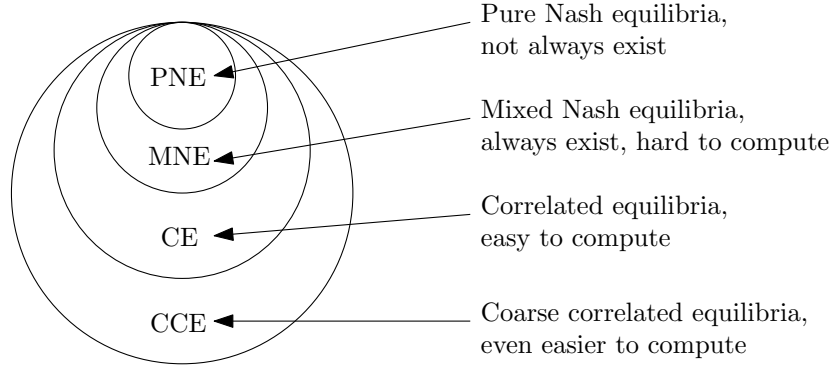
Figure 2.4: A hierarchy of variants of Nash equilibria.

Theorem 2.54. *Let $G = (P, A, C)$ be a normal-form game of n players, let $\varepsilon > 0$ be a given parameter, and $T = T(\varepsilon) \in \mathbb{N}$ be a given number of steps. Assume that after T time steps of the No-regret dynamics, each player $i \in P$ has time-averaged expected regret at most ε. Let $p^t = \prod_{i=1}^n p_i^t$ denote the outcome probability distribution at time step t and let $p = \frac{1}{T} \sum_{t=1}^T p^t$ be the time-averaged history of these distributions. Then p is an ε-coarse correlated equilibrium, that is,*

$$\mathbb{E}_{a \sim p}[C_i(a)] \leq \mathbb{E}_{a \sim p}[C_i(a_i'; a_{-i})] + \varepsilon$$

for every player $i \in P$ and $a_i' \in A_i$.

Proof. By the definition of p, we have, for every player $i \in P$ and $a_i' \in A_i$,

$$\mathbb{E}_{a \sim p}[C_i(a)] = \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(a)]$$

and

$$\mathbb{E}_{a \sim p}[C_i(a_i'; a_{-i})] = \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(a_i'; a_{-i})].$$

The right-hand sides of both equalities are the time-averaged expected costs of player i when playing according to his algorithm with small external regret and when playing the fixed action a_i' every iteration, respectively. Since every player has regret at most ε, we obtain

$$\frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(a)] \leq \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(a_i'; a_{-i})] + \varepsilon.$$

This verifies the ε-coarse correlated equilibrium condition for $p = \frac{1}{T} \sum_{t=1}^T p_t$. $\qquad \square$

2.6.5 Swap regret and correlated equilibria

External regret uses the fixed comparison class $\mathcal{A}_X$, but we can also study comparison classes that depend on the actions of our online algorithm A. We consider modification rules that modify action selected by A and thus produce an alternative strategy which we will want to compete against. So-called *internal regret,* deals with the situation when one is allowed to change the online action sequence by changing every occurrence of action i by a different action j. In *swap regret,* we allow a more general modification rule by allowing any mapping from X to X. We also present a simple way to efficiently convert any external regret-minimizing algorithm into an algorithm that minimizes swap regret with only a factor N increase in the regret term.

First, we state the definitions of internal and swap regret more formally. Again, there are N available actions $X = \{1, \ldots, N\}$ and at each time step $t \in \{1, \ldots, T\}$ the online algorithm A

selects a probability distribution $p^t = (p_1^t, \ldots, p_N^t)$ over X. After the distribution p^t is chosen at time step t, the adversary chooses a loss vector $\ell^t = (\ell_1^t, \ldots, \ell_N^t) \in [-1, 1]^N$, where the number ℓ_i^t is the loss of action i in time t. We also recall that $L_A^T = \sum_{t=1}^T \ell_A^t$ denotes the loss of A, where $\ell_A^t = \sum_{i=1}^N p_i^t \ell_i^t$ is the loss experienced by A in time step t.

A *modification rule F* is a function of the type $X \to X$. The function F outputs a possibly different action than the current one selected by A. Given a sequence $(p^t)_{t=1}^T$ of the probability distributions used by A and a modification rule F, we define a *modified sequence* $(f^t)_{t=1}^T = (F(p^t))_{t=1}^T$, where $f^t = (f_1^t, \ldots, f_N^t)$ and $f_i^t = \sum_{j: \, F(j)=i} p_j^t$. Note that each term of the modified sequence is a probability distribution. The *loss of the modified sequence* is $L_{A,F}^T = \sum_{t=1}^T \sum_{i=1}^N f_i^t \ell_i^t$.

Let $\mathcal{F}$ be a finite set of modification rules. Given a sequence ℓ^t of loss vectors, the *regret of A with respect to modification rules from $\mathcal{F}$* is

$$R_{A,\mathcal{F}}^T = \max_{F \in \mathcal{F}} \left\{ L_A^T - L_{A,F}^T \right\}.$$

In particular, for a set $\mathcal{F}^{ex} = \{F_i : i \in X\}$ of N modification rules F_i, where F_i always outputs action i, we obtain exactly the external regret setting with the external regret of A equal to

$$R_{A,\mathcal{F}^{ex}}^T = \max_{F \in \mathcal{F}^{ex}} \left\{ L_A^T - L_{A,F}^T \right\} = \max_{j \in X} \left\{ \sum_{t=1}^T \left(\left(\sum_{i \in X} p_i^t \ell_i^t \right) - \ell_j^t \right) \right\}.$$

In the *internal regret* setting, we consider the set $\mathcal{F}^{in} = \{F_{i,j} : (i,j) \in X \times X, i \neq j\}$ of $N(N-1)$ modification rules $F_{i,j}$, where, for every time step t, $F_{i,j}(i) = j$ and $F_{i,j}(i') = i'$ for each $i' \neq i$. That is, the modification rule $F_{i,j}$ always keeps all actions besides i, which is changed to a different action j. The *internal regret of A* is then

$$R_{A,\mathcal{F}^{in}}^T = \max_{F \in \mathcal{F}^{in}} \left\{ L_A^T - L_{A,F}^T \right\} = \max_{i,j \in X} \left\{ \sum_{t=1}^T p_i^t (\ell_i^t - \ell_j^t) \right\}.$$

In a more general *swap regret* setting, we work with the set $\mathcal{F}^{sw}$ of all N^N modification rules $F : X \to X$, where a function F swaps the current online action i with $F(i)$ (we might have $i = F(i)$). Then the *swap regret of A* is

$$R_{A,\mathcal{F}^{sw}}^T = \max_{F \in \mathcal{F}^{sw}} \left\{ L_A^T - L_{A,F}^T \right\} = \sum_{i=1}^N \max_{j \in X} \left\{ \sum_{t=1}^T p_i^t (\ell_i^t - \ell_j^t) \right\}.$$

That is, in swap regret, we want the best performance with respect to a fixed modification rule, not with respect to a fixed action as in the case of the external regret. Since $\mathcal{F}^{ex}, \mathcal{F}^{in} \subseteq \mathcal{F}^{sw}$, we immediately have $R_{A,\mathcal{F}^{ex}}^T, R_{A,\mathcal{F}^{in}}^T \leq R_{A,\mathcal{F}^{sw}}^T$ for every A.

We now show a black-box reduction that shows how to use an algorithm with a good bound on external regret to produce an algorithm that achieves good swap regret. An *R-external regret algorithm A* guarantees that for every sequence $(\ell^t)_{t=1}^T$ of T loss vectors and for every action $j \in X$, we have

$$L_A^T = \sum_{t=1}^T \ell_A^t \leq \sum_{t=1}^T \ell_j^t + R = L_j^T + R.$$

Theorem 2.55. *For every R-external regret algorithm A, there exists an online algorithm $M = M(A)$ such that, for every function $F : X \to X$ and $T \in \mathbb{N}$, we have*

$$L_M^T \leq L_{M,F}^T + NR.$$

That is, the swap regret of M is at most NR.

Proof. Assume that $A_1, \ldots, A_N$ are copies of the R-external regret algorithm A. We construct the master algorithm $M = M(A)$ by combining these copies of A. In every time step t, each

A_i outputs a probability distribution $q_i^t = (q_{i,1}^t, \ldots, q_{i,N}^t)$, where $q_{i,j}^t$ is the fraction A_i assigns to an action $j \in X$. We construct a single probability distribution $p^t = (p_1^t, \ldots, p_N^t)$ by letting $p_j^t = \sum_{i=1}^N p_i^t q_{i,j}^t$ for every $j \in X$. That is, $(p^t)^\top = (p^t)^\top Q^t$, where Q^t is an $N \times N$ matrix with $(Q^t)_{i,j} = q_{i,j}^t$. It can be shown that such a distribution p^t exists and is efficiently computable as if we interpret Q^t as the transition matrix of a Markov chain, then p^t is its stationary distribution. This choice of p^t guarantees that we can consider action selection in two equivalent ways. An action $j \in X$ is either selected with a probability p_j^t or we first select an algorithm A_i with probability p_i^t and then use the algorithm A_i to select j with probability $q_{i,j}^t$.

When the adversary returns the loss vector ℓ^t, we give, for each $i \in X$, a loss vector $p_i^t \ell^t$ to the algorithm A_i. The algorithm A_i then experiences loss $(p_i^t \ell^t) \cdot q_i^t = p_i^t (q_i^t \cdot \ell^t)$ at time step t (we recall that q_i^t is a vector). Now, A_i is an R-external regret algorithm, so, for every action $j \in X$, we have

$$\sum_{t=1}^T p_i^t (q_i^t \cdot \ell^t) \leq \sum_{t=1}^T p_i^t \ell_j^t + R. \tag{2.8}$$

Summing the losses of all algorithms A_i at time step t over $i \in X$, we get the total loss $\sum_{i=1}^N p_i^t (q_i^t \cdot \ell^t) = (p^t)^\top Q^t \ell^t$ of all algorithms A_i at time step t. By the choice of p^t, we have $(p^t)^\top = (p^t)^\top Q^t$. Thus, the total loss of all algorithms A_i at time step t equals $p^t \cdot \ell^t$, the actual loss of M at time step t.

Therefore, summing (2.8) over all actions $i \in X$, the left-hand side equals L_M^T. The right-hand side of (2.8) is true for every action $j \in X$, so we obtain, for every function $F \colon X \to X$,

$$L_M^T \leq \sum_{i=1}^N \sum_{t=1}^T p_i^t \ell_{F(i)}^t + NR = L_{M,F}^T + NR. \qquad \square$$

In particular, Theorem 2.55 together with Theorem 2.49 yield the following corollary.

Corollary 2.56. *There exists an online algorithm A such that, for every function $F \colon X \to X$ and $T \in \mathbb{N}$, we have*

$$L_A^T \leq L_{A,F}^T + O(N\sqrt{T \log N}).$$

That is, the swap regret of A is at most $O(N\sqrt{T \log N})$.

In Theorem 2.54, we have seen that online algorithms with external regret $o(T)$ can be used in so-called No-regret dynamics to converge to a coarse correlated equilibrium of a given normal-form game G. Similarly, we show that a *No-swap-regret dynamics* converges to a correlated equilibrium of G. This dynamics is the same as for external regret, we just use an online algorithm with small swap regret from Corollary 2.56 instead of the Polynomial weights algorithm.

Algorithm 2.6.5: NO-SWAP-REGRET DYNAMICS(G, T, ε)

$Input$: A normal-form game $G = (P, A, C)$ of n players, $T \in \mathbb{N}$ and $\varepsilon > 0$.
$Output$: A probability distribution p_i^t on A_i for each $i \in P$ and $t \in \{1, \ldots, T\}$.
for every step $t = 1, \ldots, T$

do $\begin{cases} \text{Each player } i \in P \text{ independently chooses a mixed strategy } p_i^t \text{ using} \\ \text{an algorithm with swap regret at most } \varepsilon, \text{ with actions corresponding to} \\ \text{pure strategies.} \\ \text{Each player } i \in P \text{ receives a loss vector } \ell_i^t = (\ell_i^t(a_i))_{a_i \in A_i}, \text{ where} \\ \ell_i^t(a_i) \leftarrow \mathbb{E}_{a_{-i}^t \sim p_{-i}^t}[C_i(a_i; a_{-i}^t)] \text{ for the product distribution} \\ p_{-i}^t = \prod_{j \neq i} p_j^t. \end{cases}$

Output $\{p^t \colon t \in \{1, \ldots, T\}\}$.

The connection between correlated equilibria and no-swap-regret dynamics is then the same as the connection between coarse correlated equilibria and the no-regret dynamics.

Theorem 2.57. *Let $G = (P, A, C)$ be a normal-form game of n players, let $\varepsilon > 0$ be a given parameter, and $T = T(\varepsilon) \in \mathbb{N}$ be a given number of steps. Assume that after T time steps of the No-swap-regret dynamics, each player $i \in P$ has time-averaged expected swap regret at most ε. Let $p^t = \prod_{i=1}^n p_i^t$ denote the outcome probability distribution at time step t and let $p = \frac{1}{T} \sum_{t=1}^T p^t$ be the time-averaged history of these distributions. Then p is an ε-correlated equilibrium, that is,*

$$\mathbb{E}_{a \sim p}[C_i(a)] \le \mathbb{E}_{a \sim p}[C_i(F(a_i); a_{-i})] + \varepsilon$$

for every player $i \in P$ and each modification rule $F \colon A_i \to A_i$.

We note that a probability distribution p is a correlated equilibrium, that is, $\mathbb{E}_{a \sim p}[C_i(a) \mid a_i] \le \mathbb{E}_{a \sim p}[C_i(a_i'; a_{-i}) \mid a_i]$ for every $i \in P$ and all $a_i, a_i' \in A_i$, if and only if $\mathbb{E}_{a \sim p}[C_i(a)] \le \mathbb{E}_{a \sim p}[C_i(F(a_i); a_{-i})]$ for every player $i \in P$ and each modification rule $F \colon A_i \to A_i$; see Exercise 2.31.

Proof. By the definition of p, we have, for every player $i \in P$ and every modification rule $F \colon A_i \to A_i$,

$$\mathbb{E}_{a \sim p}[C_i(a)] = \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(a)]$$

and

$$\mathbb{E}_{a \sim p}[C_i(F(a_i); a_{-i})] = \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(F(a_i); a_{-i})].$$

The right-hand sides of both qualities are the time-averaged expected costs of player i when playing according to his algorithm with small swap regret and when playing the action $F(a_i)$ instead of a_i every iteration, respectively. Since every player has average swap regret at most ε, we obtain

$$\frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(a)] \le \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(F(a_i); a_{-i})] + \varepsilon.$$

This verifies the ε-correlated equilibrium condition for $p = \frac{1}{T} \sum_{t=1}^T p^t$. $\qquad\square$

Finally, we note that we could also replace swap regret with the internal regret in Theorem 2.57 and p would still be a ε-correlated equilibrium.

2.6.6   Exercises

Exercise 2.25. *In this exercise, we get rid of the assumption in the second part of the statement of Theorem 2.49, where we assume the number of steps T is given in advance.*

Let A be the Polynomial weights algorithm with parameter $\eta \in (0, 1/2]$ and with external regret at most $\alpha/\eta + \beta\eta T$ for some constants α, β (that may depend on the number N of actions). We showed that choosing $\eta = \sqrt{\alpha/(T\beta)}$ minimizes the bound. Modify this algorithm so that we obtain an external regret bound that is at most $O(1)$-times larger than the original bound for any T. In particular, you cannot run A with a parameter η that depends on T.

Hint: Partition the set $\{1, \ldots, T\}$ into suitable intervals I_m for $m = 0, 1, 2, \ldots$ and run A with a suitable parameter η_m in every step from I_m. **[3]**

Exercise 2.26. *Consider the following setting in which the agent A tries to learn the setup in an adversary environment while using information given to him by a set S_0 of N experts. The setting proceeds in a sequence of steps $t = 1, \ldots, T$. In every step t, the environment picks $y_t \in \{0, 1\}$, which is unknown to A and to the experts, and each expert i gives a recommendation $f_{i,t} \in \{0, 1\}$ to A. The agent A then makes prediction $z_t \in \{0, 1\}$ based on the experts' advice and then sees y_t. The goal of A is to minimize the number $M^T(A)$ of steps t in which $z_t \ne y_t$.*

(a) *Assume that, in each step t, the agent A selects z_t to be the majority vote of all experts from S_{t-1} and, after seeing y_t, he lets S_t be the number of agents $i \in S_{t-1}$ with the right guess $f_{i,t} = z_t$. Also, assume that there is a perfect expert who always guesses right. Prove that then $M^T(A) \leq \log_2 N$.* **[1]**

(b) *Modify the above algorithm of the agent so that $M^T(A) \leq O((m+1) \log_2 N)$ when the best expert makes $m \geq 0$ mistakes.* **[2]**

Exercise 2.27. *Show formally that every correlated equilibrium is a coarse correlated equilibrium.* **[1]**

Exercise 2.28. *Show that the swap regret is at most N times larger than the internal regret.* **[2]**

Exercise 2.29. *Show an example with $N = 3$ where the external regret is zero and the swap regret goes to infinity with T.* **[3]**
  Clarification: you need to choose only a sequence of actions $a^1, \ldots, a^T$, $a^t \in X = \{1, 2, 3\}$, and a loss sequence $\ell_a^1, \ldots, \ell_a^T$ for every $a \in X$.

Exercise 2.30. *Let $G = (P = \{1, 2\}, A, u)$ be a normal-form game of two players with $A_1 = \{a, b, c\}$ a $A_2 = \{d, e, f\}$ and with the utility function from Table 2.16.*

	d	e	f
a	(1,1)	(-1,-1)	(0,0)
b	(-1,-1)	(1,1)	(0,0)
c	(0,0)	(0,0)	(-1.1,-1.1)

Table 2.16: Thee game from Exercise 2.30.

Show that the probability distribution p on A with $p(a, d) = p(b, e) = p(c, f) = 1/3$ is a coarse correlated equilibrium in G (CCE), but it is not a correlated equilibrium in G (CE). **[2]**

Exercise 2.31. *Show that a probability distribution p is a correlated equilibrium, that is, we have $\mathbb{E}_{a \sim p}[C_i(a) \mid a_i] \leq \mathbb{E}_{a \sim p}[C_i(a_i'; a_{-i}) \mid a_i]$ for every $i \in P$ and all $a_i, a_i' \in A_i$, if and only if $\mathbb{E}_{a \sim p}[C_i(a)] \leq \mathbb{E}_{a \sim p}[C_i(F(a_i); a_{-i})]$ for every player $i \in P$ and each modification rule $F : A_i \to A_i$* **[3]**

2.7 Games in extensive form

In normal-form games, players act simultaneously resulting in a static description of an interactive situation. Here, we describe a different representation of games which provides a dynamic description where players act sequentially. For a subclass of such games, we show how to efficiently compute Nash equilibria using linear programming.

Extensive game[33] consists of a directed tree where nodes represent game states. The tree encodes the full history of play. The game starts at the root of the tree and ends at a leaf, where each player receives a payoff. A tree edge corresponds to one player moving from one state to a different state of the game. Each node that is not a leaf is called a *decision node*. *Moves* a player can make in a given state are assigned to the outgoing edges of the corresponding decision node.

For example, the extensive form of Chess consists of a tree whose root corresponds to the standard initial position of the chessboard. Each decision node represents a position on the chessboard and its outgoing edges correspond to possible moves in such a position. Nodes of this tree on odd levels are positions where the white player is on the move and the nodes on even levels are positions where the black player is on the move.

If players always know the node they are in (that is, in every node they know the history of the play that led to it), then we have a *perfect-information game*[34]. For example, Chess is a

[33] Hra v rozšířené formě či v explicitní formě.
[34] Hry s dokonalou informací.

perfect-information game. However, in general, there are many situations where the players have only partial or no knowledge of the action taken by other players or even by themselves. Consider, for example, the game of Poker. Such *imperfect-information games*[35] can be modeled if we extend the definition of extensive games by partitioning decision nodes into *information sets* where all nodes in an information set belong to the same player, and have the same moves. The interpretation is that when a player makes a move, he only knows the information set but not the particular node he is at. Thus, perfect-information games are a special case, where all information sets are singletons. For player i, we let H_i be the set of information sets of i and, for an information set $h \in H_i$, we let C_h be the set of moves at h.

Example 2.58 (A game in extensive form). *An example of an imperfect-information game in extensive form; see Figure 2.5. Player 1 has two information sets. Note that player 1 does not know whether player 2 played action ℓ or r if he is in a node from the bottom information set.*

(a)

(b)

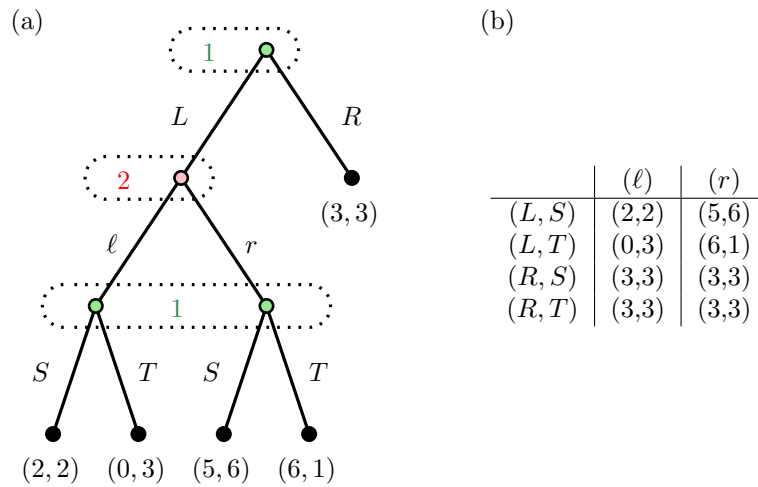|	(ℓ)	(r)
(L,S)	(2,2)	(5,6)
(L,T)	(0,3)	(6,1)
(R,S)	(3,3)	(3,3)
(R,T)	(3,3)	(3,3)

Figure 2.5: (a) An example of an extensive game G. The decision nodes are green for player 1 and red for player 2. The leaves are black and the information sets are represented by dotted ovals. (b) The normal-form game of G.

Example 2.59 (Prisoner's dilemma in extensive form). *An equivalent imperfect-information game in extensive form to Prisoner's dilemma is shown in Figure 2.6. Note that we could have chosen to make player 2 choose first and player 1 choose second.*

(a)

(b)

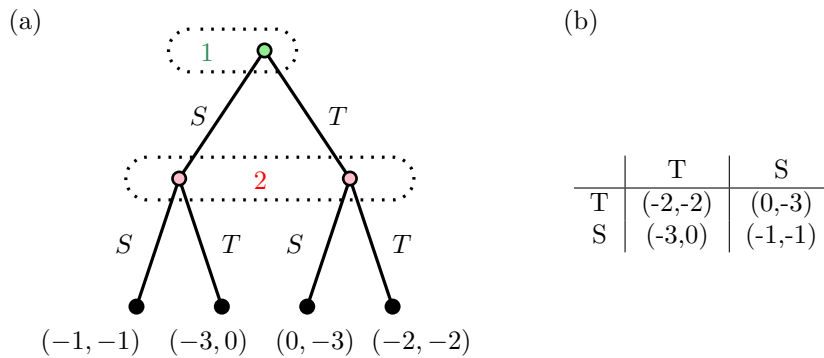|	T	S
T	(-2,-2)	(0,-3)
S	(-3,0)	(-1,-1)

Figure 2.6: (a) An extensive form of Prisoner's dilemma. (b) Its normal form.

It should be obvious from Example 2.59 that every normal-form game can be trivially transformed into an equivalent imperfect-information game in extensive form. Note that this transformation is not one-to-one, as mentioned in Example 2.59.

---

[35] Hry s nedokonalou informací.

2.7.1   Strategies and equilibria in extensive games

A *pure strategy* for player i is a complete specification of which deterministic action to take at every information set belonging to that player. Formally, a pure strategy of player i is a vector $(c_h)_{h \in H_i}$ from the Cartesian product $\prod_{h \in H_i} C_h$.

Using pure strategies, we can transform an extensive game G into a normal-form game G' simply by tabulating all pure strategies of the players and recording the resulting expected payoffs. Then, it is natural to define *mixed strategies* of G as the mixed strategies of G'. In the same way, we can also define the set of *Nash equilibria* of G.

We can also define a *behavioral strategy*[36] of player i as a probability distribution on C_h for each $h \in H_i$. That is, a behavioral strategy is a strategy in which each player's probabilistic choice at each information set is made independently of his choices at other information sets. So a behavioral strategy is a vector of probability distributions while a mixed strategy is a probability distribution over vectors. For example, in mixed strategy, a player plays the same move whenever he encounters an information set h while in behavioral strategy, a player might play different moves in different encounters of h.

Example 2.60. *Consider the information-perfect game from Figure 2.7. Then, a strategy of player 1 that selects A with probability $\frac{1}{2}$ and G with probability $\frac{1}{3}$ is a behavioral strategy. On the other hand, the mixed strategy $(\frac{3}{5}(A, G), \frac{2}{5}(B, H))$ is not a behavioral strategy for player 1 as the choices made by him at the two nodes are not independent.*

(a)

(b)

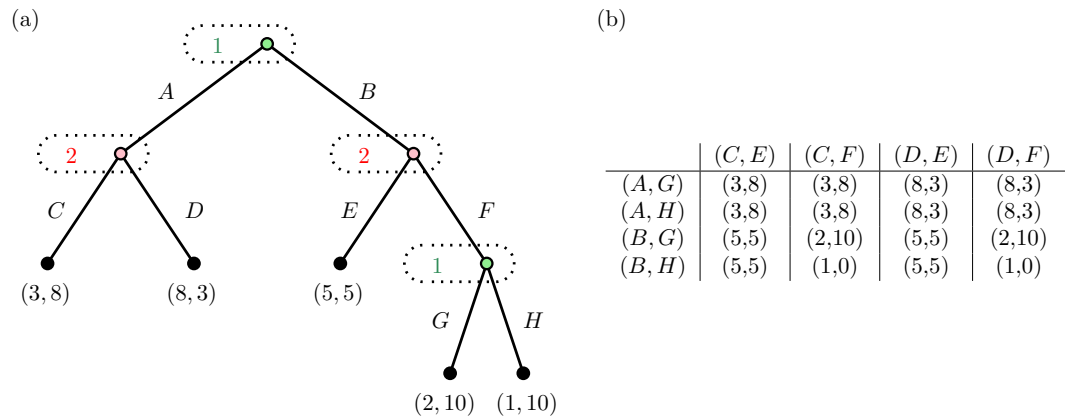|	(C, E)	(C, F)	(D, E)	(D, F)
(A, G)	(3,8)	(3,8)	(8,3)	(8,3)
(A, H)	(3,8)	(3,8)	(8,3)	(8,3)
(B, G)	(5,5)	(2,10)	(5,5)	(2,10)
(B, H)	(5,5)	(1,0)	(5,5)	(1,0)

Figure 2.7: A perfect-information game in extensive form.

We note that some games, such as *Russian roulette* contain certain randomness where, at some decision nodes, the next move is determined at random. To model such situations, we use an additional player, numbered 0, who receives no payoff and who plays according to a known behavior strategy β_0. His moves are then called *chance moves*.

Example 2.61 (Russian roulette). *In Russian roulette, we have two players with a six-shot revolver containing a single bullet. Each player has two moves: shoot or give up. If a player gives up, he loses the game immediately. If he shoots, then he either dies or survives, in which case the other player is on a turn. Consider that player 1 has payoffs $(10, 2, 1)$ for (Win, Loss, Death) and that player 2 has payoffs $(10, 0, 0)$. The extensive form of this game is depicted in Figure 2.8. Note that the chance moves that correspond to occurrences of the bullet are simulated using the additional player 0.*

In general, the expressive power of behavioral strategies and the expressive power of mixed strategies are incomparable as in some games there are outcomes that can be achieved via mixed strategies and not by behavioral strategies and in some games it is the other way around. However, there is a large class of extensive games for which the two definitions coincide. This
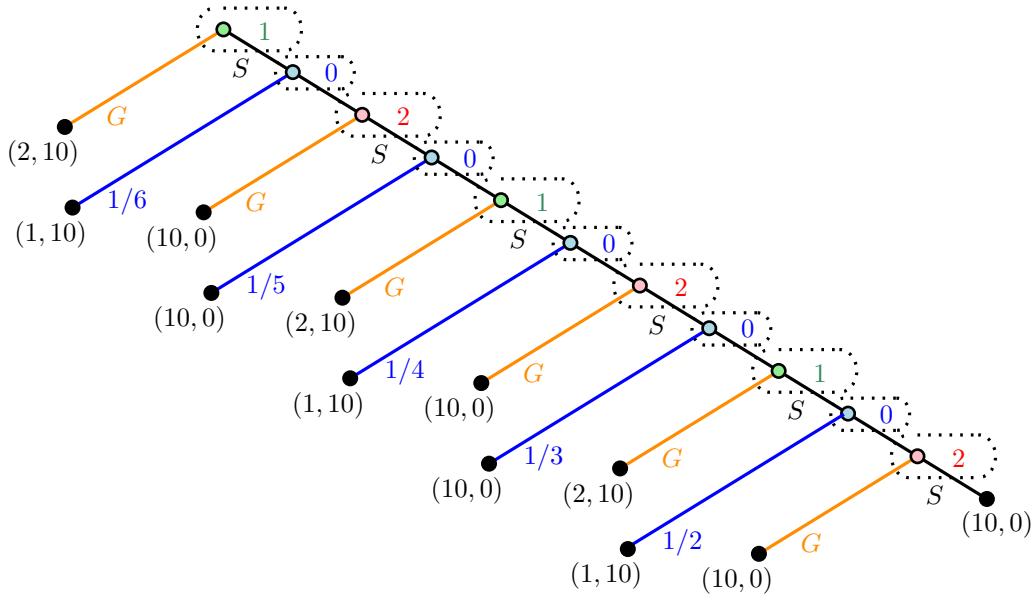
---

[36] Behaviorální strategie.

Figure 2.8: An extensive form of Russian roulette using chance moves. The probability of each chance move corresponding to the revolver shooting is denoted blue and the corresponding edge is also blue. The moves of players 1 and 2 are "shoot" (S, black) or "give up" (G, orange).

is the class of *games of perfect recall*[37] where no player forgets any information he knew about moves made so far; in particular, he remembers precisely all his own moves. To define this class formally, we introduce some auxiliary terms.

A *sequence* $\sigma_i(t)$ of moves of player i to a node t is the sequence of his moves (disregarding the moves of other players) on the unique path from the root of the tree to t. The empty sequence is denoted $\emptyset$. Player i has *perfect recall* if and only if, for every $h \in H_i$ and any nodes $t, t' \in h$, we have $\sigma_i(t) = \sigma_i(t')$. In such case, we use σ_h to denote the unique sequence leading to any node t in h. A game G is a *game of perfect recall* if every player has perfect recall in G. That is, each player remembers what he did in prior moves, and each player remembers everything that he knew before. Note that every perfect-information game is a game of perfect recall.

The following result states that in games of perfect recall, mixed strategies and behavioral strategies are equivalent. Here, two strategies are *equivalent* if they reach any node of the tree with the same probabilities, given any strategy (mixed or behavioral) of the other players.

Theorem 2.62 (Kuhn's theorem, 1953)**.** *In a game of perfect recall, any mixed strategy of a given player can be replaced by an equivalent behavioral strategy, and any behavioral strategy can be replaced by an equivalent mixed strategy.*

It follows from Theorem 2.62 that the set of Nash equilibria does not change if we restrict ourselves to behavioral strategies in games of perfect recall. In general imperfect-information games, mixed and behavioral strategies yield incomparable sets of Nash equilibria.

From now on, *we consider only games of perfect recall*.

2.7.2   Sequence form

Since every extensive game G can be converted into an equivalent normal-form game G', we can find a Nash equilibrium of G by converting it into G' and applying, for example, the Lemke–Howson algorithm to G'. However, this is inefficient, as the number of actions in G' is

[37] Hry s dokonalou pamětí.

exponential in the size of G. To avoid this problem, we will work directly with G using so-called *sequence form*[38].

The *sequence form* of an imperfect-information game G is a 4-tuple (P, S, u, C) where P is a set of n players, $S = (S_1, \ldots, S_n)$, where S_i is a set of sequences of player i, $u = (u_1, \ldots, u_n)$, where $u_i \colon S \to \mathbb{R}$ is the payoff function of player i, and $C = (C_1, \ldots, C_n)$ is a set of linear constraints on the realization probabilities of player i. Now, we will define all these terms.

Any sequence σ from S_i is either the empty sequence $\emptyset$ or it is uniquely determined by the last move c at the information set h, that is, $\sigma = \sigma_h c$. Thus, $S_i = \{\emptyset\} \cup \{\sigma_h c \colon h \in H_i, c \in C_h\}$. It follows that $|S_i| = 1 + \sum_{h \in H_i} |C_h|$, which is linear in the size of the tree of G.

For player i and sequences $\sigma = (\sigma_1, \ldots, \sigma_n) \in S$, the *payoff* $u_i(\sigma)$ equals $u_i(\ell)$ where ℓ is the leaf that would be reached if each player j played his sequence σ_j. Otherwise, $u_i(\sigma) = 0$. Similarly as in the normal-form games, we can represent the payoffs u using a table with entries indexed by elements from S. Note that this table is sparse as most entries are 0; see Example 2.64.

We still do not have everything we need to describe our game since a player cannot choose sequences as actions because other players might not play in a way that would allow him to follow to a leaf. What we want is for players to select behavioral strategies since, by Theorem 2.62, equilibrium will be expressible using behavioral strategies. However, we cannot work with behavioral strategies directly as then the resulting optimization problems would be computationally difficult. Instead, we develop an alternate concept of a realization plan, which is the probability that a sequence arises under a given behavioral strategy. The *realization plan*[39] of a behavioral strategy β_i for player i is a mapping $x \colon S_i \to [0, 1]$ defined as $x(\sigma_i) = \prod_{c \in \sigma_i} \beta_i(c)$. The value $x(\sigma_i)$ is called the *realization probability*.

There is an equivalent definition of the realization plan that is more convenient, as it involves a set of linear equations, although it is probably not so transparent. A realization plan for player i is a mapping $x \colon S_i \to [0, 1]$ satisfying the following two constraints:

1. $x(\emptyset) = 1$, and

2. $\sum_{c \in C_h} x(\sigma_h c) = x(\sigma_h)$ for every $h \in H_i$.

We let C_i be the set of constraints of the second type. This finishes the definition of the sequence form of G.

Note that we can uniquely recover behavioral strategy β_i from a realization plan x for player i in all relevant information sets, that is, sets $h \in H_i$ with $x(\sigma_h) > 0$. For each such $h \in H_i$ and $c \in C_h$, we set $\beta_i(h, c) = \frac{x(\sigma_h c)}{x(\sigma_h)}$. The irrelevant information sets ($h \in H_i$ with $x(\sigma_h) = 0$) are nodes that can never be reached in a play due to the player's own previous decisions.

2.7.3 Computing equilibria in two-player extensive games

We now show how to use the sequence form to compute equilibria much more efficiently than using the induced normal-form game. We consider only games of two players 1 and 2 with a possible chance player 0.

Consider realization plans as vectors $x = (x_\sigma)_{\sigma \in S_1} \in \mathbb{R}^{|S_1|}$ and $y = (y_\tau)_{\tau \in S_2} \in \mathbb{R}^{|S_2|}$. Then, the linear constraints from the definition of sequence form can be written as

$$Ex = e, \quad x \geq \mathbf{0} \quad \text{and} \quad Fy = f, \quad y \geq \mathbf{0}$$

where the *constraint matrices* E and F have $1 + |H_1|$ and $1 + |H_2|$ rows, respectively, with first row of $Ex = e$ and $Fy = f$ corresponding to $x(\emptyset) = 1$ for E and $y(\emptyset) = 1$ for F; see Example 2.63. The other rows of $Ex = e$ are the equations $-x(\sigma_h) + \sum_{c \in C_h} x(\sigma_h c) = 0$ for every $h \in H_1$. For $Fy = f$, we have the rows $-y(\sigma_h) + \sum_{c \in C_h} y(\sigma_h c) = 0$ for every $h \in H_2$. The vectors e and f also have $1 + |H_1|$ and $1 + |H_2|$ rows, respectively.

---

[38] Reprezentace posloupnostmi.
[39] Realizační plán.

Example 2.63. *Consider the extensive game from Example 2.58. The sets of sequences in this game are $S_1 = \{\emptyset, L, R, LS, LT\}$ and $S_2 = \{\emptyset, \ell, r\}$. The corresponding constraint matrices E and F and the right hand side vectors e and f are*

$$E = \begin{pmatrix} 1 & & & & \\ -1 & 1 & 1 & & \\ & -1 & & 1 & 1 \end{pmatrix}, \quad e = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad F = \begin{pmatrix} 1 & & \\ -1 & 1 & 1 \end{pmatrix}, \quad f = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

The *sequence payoff matrices* A and B are $|S_1| \times |S_2|$ matrices where

$$A_{\sigma,\tau} = \sum_{\text{leaves } t \,:\, \sigma_1(t)=\sigma, \sigma_2(t)=\tau} u_1(t)\beta_0(\sigma_0(t)) \text{ and } B_{\sigma,\tau} = \sum_{\text{leaves } t \,:\, \sigma_1(t)=\sigma, \sigma_2(t)=\tau} u_2(t)\beta_0(\sigma_0(t))$$

for all $\sigma \in S_1$ and $\tau \in S_2$. These two matrices are sparse as they have nonzero entries only for pairs (σ, τ) that lead to a leaf. The expected payoffs of players 1 and 2 are then $x^\top A y$ and $x^\top B y$, respectively.

Example 2.64. *Consider the extensive game from Example 2.58 and see Figure 2.9 for its sequence form payoff matrices with omitted 0 entries. Note that the matrices are sparse and their size is linear with respect to the size of the tree of the game.*
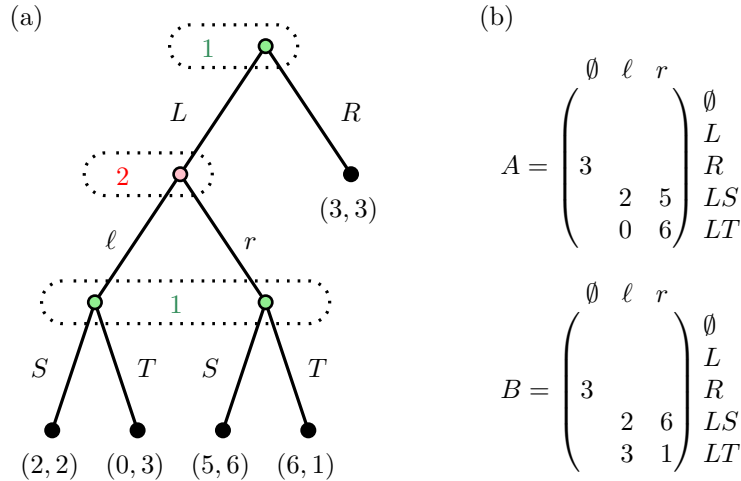

Figure 2.9: (a) An extensive game from Example 2.58. (b) Its sequence form payoff matrices.

We first show how to compute best responses. For a fixed realization plan y of player 2, a best response of player 1 is a realization plan x that maximizes the expected payoff. Thus, x is a solution to the following linear program P

$$\max x^\top A y \text{ subject to}$$
$$Ex = e,$$
$$x \geq \mathbf{0}.$$

The dual D of P uses unconstrained variables u and is of the form

$$\min e^\top u \text{ subject to}$$
$$E^\top u \geq Ay.$$

Analogous linear programs can be used to compute best responses of player 2.

Assume now that our game is *zero sum*, that is, $B = -A$. Then, using the Duality Theorem (Theorem 2.22) to these linear programs, similarly as we did in Section 2.3, will give a linear program for finding equilibrium in a given zero-sum game. The reason is that player 2 wants to minimize $x^\top A y$, which by duality equals $e^\top y$ if player 1 maximizes his payoff $x^\top A y$. It is important that the constraints in the program D remain linear even if y is treated as a variable.

Theorem 2.65. *The equilibria of a two-player zero-sum game in an extensive form of perfect recall are the solutions to the following linear program:*

$$\min_{u,y} e^\top u \text{ subject to } Fy = f, E^\top u - Ay \geq \mathbf{0}, y \geq \mathbf{0}.$$

The dual of this program has variables x and v and is of the form

$$\max_{v,x} f^\top v \text{ subject to } Ex = e, F^\top v - A^\top x \leq \mathbf{0}, x \geq \mathbf{0}.$$

This linear program then finds realization plan x with payoff $f^\top v$ for player 1.

The number of nonzero entries in matrices $E, F.A, B$ is linear in the size of the game tree, thus the linear program from Theorem 2.65 can be solved in polynomial time with respect to the size of the given extensive game. This is an exponential improvement over trying to solve the induced normal-form game.

We can proceed similarly for general extensive games of 2 players with perfect recall. However, in the end we do not obtain a linear program, but a so-called *linear complementarity problem*.

Again, the best response of player 1 to a fixed realization plan y is given by the linear program P. By the Duality Theorem (Theorem 2.22) a feasible solution x to P is optimal if and only if there is a solution u of D satisfying $E^\top u \geq Ay$ and $x^\top(Ay) = e^\top u$. That is, $x^\top(Ay) = (x^\top E^\top)u$ as $Ex = e$. Equivalently, $x^\top(E^\top u - Ay) = 0$. Since the vectors x and $E^\top u - Ay$ are non-negative, this equality states that they are *complementary*, meaning that they cannot both have positive components in the same position. This characterization of an optimal primal-dual pair of feasible solutions is known as *complementary slackness* in linear programming. For player 2, we analogously derive the equality $y^\top(F^\top v - B^\top x) = 0$.

Considering these conditions for both players, we obtain the following result.

Theorem 2.66. *A pair (x, y) of realization plans in a 2-player game in the extensive form of perfect recall is equilibrium if and only if there are vectors u and v such that the following conditions are satisfied:*

$$\begin{aligned}
x^\top(E^\top u - Ay) = 0, \quad &y^\top(F^\top v - B^\top x) = 0, \\
Ex = e, x \geq \mathbf{0}, \quad &Fy = f, y \geq \mathbf{0}, \\
E^\top u - Ay \geq \mathbf{0}, \quad &F^\top v - B^\top x \geq \mathbf{0}.
\end{aligned}$$

Note that some of the conditions are not linear and thus we do not have a linear program. However, this is the so-called *linear complementarity problem*, where we are given a matrix M and a vector q and we want to find vectors z and w such that $w, z \geq \mathbf{0}$, $z^\top w = 0$, and $w = Mz + q$. The perhaps most well-known method for solving linear complementarity problems is *Lemke's algorithm*, which has exponential running time in the worst case, similarly as the Lemke–Howson algorithm. Nevertheless, it is exponentially faster to run Lemke's algorithm on a game in a sequence form than to run the Lemke–Howson algorithm on its induced normal-form game.

2.7.4 Exercises

Exercise 2.32. *Construct an extensive form of the Game of chicken from Table 2.17 and write its sequence form and the linear complementarity problem for finding Nash equilibria in this game.* **[2]**

Exercise 2.33. *Construct an extensive form of the Rock-Paper-Scissors game from Table 2.18 and write its sequence form and the linear program for finding Nash equilibria in this game.* **[2]**

	Turn	Straight
Turn	(0,0)	(-1,1)
Straight	(1,-1)	(-10,-10)

Table 2.17: A normal form of the Game of chicken.

	Rock	Paper	Scissors
Rock	(0,0)	(-1,1)	(1,-1)
Paper	(1,-1)	(0,0)	(-1,1)
Scissors	(-1,1)	(1,-1)	(0,0)

Table 2.18: A normal form of the game Rock-paper-scissors.

3 Mechanism design

Contents of this chapter

This chapter is devoted to *mechanism design*[1], a subarea of game theory that deals with designing games toward desired objectives. Unlike the previous chapter, where we adopted a somewhat passive perspective by assuming the rules of the game to be fixed and then analyzed the strategic outcomes of this game, we now try to design the rules of the game so that strategic behavior by participants leads to a desirable outcome.

First, in Section 3.1, we consider *single-item auctions* and identify an auction mechanism called *Vickrey auction* with great properties. These properties will serve as a gold standard against which we will evaluate our solutions. We then extend these desired properties to a more general setting of *single-parameter environments* using so-called *Myerson's lemma* and we present some applications to illustrate the strength of this famous result. We also show, using *Knapsack auctions*, that we cannot keep all the desired properties that we had for single-item auctions in single-parameter environments. Namely, we lose the existence of polynomial-time implementations, assuming $P \neq NP$.

In all instances from the previous part, we consider only mechanisms that maximize social surplus. In Section 3.2, we study auctions that maximize *revenue* as their primary goal. We show that the situation then becomes more involved already for simple instances and introduce most classical model to study such auctions, the *Bayesian model*. Then we characterize a class of auctions that maximize the expected revenue in this model and, using this characterization, we show that Vickrey auctions with reserved prices are an optimal auction format in some cases. We finish this part by proving the *Bulow–Klemperer Theorem*, which roughly states that adding an extra bidder is as good as knowing the underlying distribution and running an optimal auction.

Finally, in Section 3.3, we consider multi-parameter environments, where every bidder can have more different private valuations for different goods, and we show that so-called *VCG mechanism* maximizes social surplus even in this very general setting. Finally, we prove the so-called *Revelation principle*, which says that we do not lose anything by restricting ourselves to a special type of dominant strategies in any mechanism that admits a dominant strategy for each bidder.

---

[1] Návrh mechanismů.

3.1 Mechanism design basics

We start our study of mechanism design with *single item auctions*[2]. In such auctions, there is a *seller* selling a single good (a painting, for example) to some number n of strategic *bidders* who are potentially interested in buying the item. Each bidder i has a *valuation* v_i that he is willing to pay for the item and he, of course, wants to buy the item as cheaply as possible, provided the selling price is at most v_i. The valuation of bidder i is private, meaning that the other bidders nor the seller know v_i.

We assume that our auction is a *sealed-bid auction*, that is, each bidder i privately communicates a *bid* b_i to the seller. The seller then decides who receives the item (if any) and decides the selling price p.

We consider perhaps the simplest natural utility model, so-called *quasilinear utility model*, in which if a bidder loses the auction, then his utility is 0. If the bidder wins the auction at price p, then his utility is $v_i - p$.

3.1.1 Vickrey's auction

Our goal is to design a mechanism how to decide the allocation of the item to a bidder in a way that cannot be strategically manipulated. To do so, we need to appropriately implement the rules for the seller how to decide the winner of the auction and how to decide the selling price.

One possible choice of payment is to sell the item for free to bidder i with the highest bid b_i. However, this is not a very good choice, as then the utility of each bidder is independent of the payment and thus bidders will benefit from exaggerating their valuations v_i by reporting b_i that is much larger than v_i. So our auction turns into a game of "who can name the highest number".

Another possible and natural approach is to sell the item to bidder i with the highest bid b_i and set the selling price to b_i. That is, the winner has to pay his declared bid b_i. This *first-price auction* looks much more reasonable and such auctions are common in practice. However, there are still some drawbacks to this approach. For example, it is difficult for the bidders to figure out how to bid. If bidder i wins and pays $b_i = v_i$, then his utility is $v_i - b_i = 0$, the same as if he loses the bid. Thus, it is clear that he should be declaring somehow lower bid b_i than v_i. Then he can still win and receive a positive utility, but what is the value b_i he should bid?

A *dominant strategy*[3] for bidder i is a strategy that maximizes the utility of bidder i, no matter what the other bidders do. The *social surplus*[4] is defined as

$$\sum_{i=1}^{n} v_i x_i,$$

where $x_i = 1$ if bidder i wins and $x_i = 0$ otherwise subject to the obvious constraint $\sum_{i=1}^{n} x_i \leq 1$ (the seller sells only a single item).

Definition 3.1 (Awesome auction). *Optimally, we would like our auction to be* awesome*, that is, to satisfy the following three conditions:*

(a) Strong incentive guarantees*: The auction is* dominant-strategy incentive-compatible (DSIC)*, that is, it satisfies the following two properties. Every bidder has a dominant strategy:* bid truthfully*, that is, set his bid b_i to his private valuation v_i. Moreover, the utility of every truth-telling bidder is guaranteed to be non-negative.*

(b) Strong performance guarantees*: If all bidders bid truthfully then the auction maximizes the social surplus.*

(c) Computational efficiency*: The auction can be implemented in polynomial time.*

[2] Jednopoložkové aukce.
[3] Dominantní strategie.
[4] Sociální zisk.

Let us justify this definition by showing that all these three properties are important. The DSIC property makes it particularly easy to choose a bid for each bidder, it suffices to bid $b_i = v_i$ for bidder i, since then i is guaranteed to achieve maximized utility, which is non-negative. That is, a bidder does not need to reason about other bidders in any way. It is also easy for the seller to reason about the auction's outcome. His prediction of the outcome does not have to be predicated on strong assumptions about how bidders behave. Instead, he can only assume that a bidder with an obvious dominant strategy will play it.

The DSIC property by itself is not enough, since, for example, an auction where the item goes for free to a random bidder is DSIC. Similarly, giving the item to nobody is also DSIC. We would like to identify which bidders actually want the good and this is why we require the strong performance guarantee. This property states that, even though the bidder valuations were a priori unknown to the seller, the auction nevertheless successfully identifies the bidder with the highest valuation (Assuming truthful bids, which is a reasonable assumption in light of the DSIC property.). That is, an awesome auction solves the surplus-maximization optimization problem as well as if the valuations were known in advance.

Finally, the importance of the third condition is rather obvious for computer scientists. An auction should run in a reasonable amount of time in order to have practical utility.

Now that we have defined the properties an ideal auction should satisfy, it remains to show that there is an awesome auction. Amazingly, there is a simple and elegant rule for the seller that leads to an awesome auction. This rule is very similar to the first-price auction, the only difference is that the seller gives the item to the highest bidder for the price equal to the second-highest declared bid.

Definition 3.2 (Vickrey's second price auction). *Let the winner be the bidder i with the highest declared value b_i and let i pay the second highest declared bid $p = \max_{j \in \{1,\dots,n\} \setminus \{i\}} b_j$.*

The following result says that Vickrey's auctions indeed satisfy all the desired properties.

Theorem 3.3 (Vickrey, [Vic61a]). *Vickrey's auction is awesome.*

Proof. It suffices to verify that Vickrey's auction satisfies all three conditions from the definition of an awesome auction.

1. Strong incentive guarantees: First, we show that each player has the dominant strategy in which he sets $b_i = v_i$. Fix bidder i .We need to show that the utility of i is maximized for $b_i = v_i$. Let $B = \max_{j \in \{1,\dots,n\} \setminus \{i\}} b_j$ be the maximum bid among bids of other bidders. If $b_i < B$, then i loses and receives utility 0. If $b_i \geq B$, then i wins and receives utility $v_i - B$ (assuming the ties are broken in favor of i, but it is easy to check that the claim holds no matter how ties are broken). Now, if $v_i < B$, then the highest utility i can get is $\max\{0, v_i - B\} = 0$, and this is achieved by bidding truthfully (and losing). Otherwise $v_i \geq B$ and the highest utility i can receive is $\max\{0, v_i - B\} = v_i - B \geq 0$ and he achieves this by bidding truthfully (and winning). Observe that the utility of i is non-negative whenever he bids truthfully. Thus, the second part of the DSIC condition is also verified.

2. Strong performance guarantees: To verify that the surplus is maximized, note that if i is the winner of the auction, then $v_i \geq v_j$ for every $j \in \{1, \dots, n\}$, as all bidders bid truthfully. Since $x_i = 1$ and $x_j = 0$ for $j \neq i$, the social surplus is then equal to v_i and is clearly maximum.

3. Computational efficiency: Vickrey's auction clearly runs in linear time, so this condition is satisfied easily. $\square$

We note that it can be shown that, in a certain sense, truthful bidding is the unique dominant strategy for a bidder in Vickrey's auction, as for every false bid $b_i \neq v_i$ by bidder i in Vickrey's auction, there is b_{-i} such that the utility of i is strictly less than when bidding v_i.

3.1.2   Myerson's lemma

Now, we would like to extend the three desired properties from the definition of single term auctions beyond the single-item auctions to more complex situations. Fortunately, there is a very satisfactory result, Myerson's lemma, stating that it is possible to obtain the DSIC property for single-parameter environments.

In a *single-parameter environment*[5], there are n bidders, each bidding for a certain goods. Each bidder i has a private *valuation* v_i, which denotes a value "per unit of the goods" that he gets. There is a *feasible set* $X \subseteq \mathbb{R}^n$ corresponding to feasible outcomes. Each element of X is a vector $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, where x_i denotes the part of the outcome that bidder i is interested in. The sealed-bid auction in this environment then proceeds in three steps.

(*a*) Collect bids $b = (b_1, \ldots, b_n)$, where b_i is the bid of bidder i.

(*b*) *Allocation rule*[6]: Choose a feasible outcome (allocation) $x = x(b)$ from the feasible set X as a function of the bids b.

(*c*) *Payment rule*[7]: Choose payments $p(b) = (p_1(b), \ldots, p_n(b)) \in \mathbb{R}^n$ as a function of the bids b.

The pair (x, p) then forms a *(direct) mechanism*[8]. The *utility* $u_i(b)$ of bidder i is defined as

$$u_i(b) = v_i \cdot x_i(b) - p_i(b).$$

Here, we consider only payments $p_i(b) \in [0, b_i \cdot x_i(b)]$ for every bidder i and all bids b. Since $p_i(b) \geq 0$, the seller never pays the bidders. The condition $p_i(b) \leq b_i \cdot x_i(b)$ says that we never charge a bidder more than his value b_i per good (that he told us) times the amount $x_i(b)$ of stuff that we gave him. It ensures that a truthtelling bidder receives non-negative utility. The basic dilemma of mechanism design is that the mechanism designer wants to optimize some global objective such as the *social surplus* $\sum_{i=1}^{n} v_i \cdot x_i(b)$.

We now illustrate the definition of the single-parameter environment with a few specific examples. We start by showing that singe-parameter environments comprise single-item auctions we met earlier.

Example 3.4 (Single-item auctions). *In single-item auctions, n bidders compete for a single item of the seller. Each bidder either gets the item or not, but only one bidder can get it. Thus, $x_i \in \{0, 1\}$ for each bidder i and thus the feasible set X is of the form $X = \{(x_1, \ldots, x_n) \in \{0, 1\}^n \colon \sum_{i=1}^{n} x_i \leq 1\}$. The goal is to design the auction so that the bidder with the highest valuation v_i wins.*

The following example is more complex than single-item auctions, as it allows more items, which, in addition, are not identical (that is, some items are more valuable than others).

Example 3.5 (Sponsored-search auctions[9]). *Here, we have n bidders competing for k positions. Each position j has a click-through-rate α_j, where $\alpha_1 > \cdots > \alpha_k > 0$. The feasible set X consists of vectors $(x_1, \ldots, x_n)$, where each x_i lies in $\{\alpha_1, \ldots, \alpha_k, 0\}$ and if $x_i = x_j$, then $x_i = 0 = x_j$. The value of slot j to bidder i is then $v_i \alpha_j$. The goal is to maximize the social surplus.*

The motivation for this example is as follows. A Web search results page contains a list of organic search results and a list of k sponsored links, which have been paid for by advertisers. Every time a search query is typed into a search engine, an auction is run in real-time to decide which sponsored links are shown, in what order, and how they are charged. The positions for sale are the k "slots" for sponsored links and slots with higher positions on the search page are more valuable than lower ones, since people generally scan the page from top to bottom. The bidders are the advertisers who have a standing bid on the keyword that was searched on. We can then

---

[5] Jednoparametrové protředí.
[6] Alokační pravidlo.
[7] Platební pravidlo.
[8] (Přímý) mechanismus.
[9] Aukce pro sponzorované vyhledávání.

view each α_j as the probability that the jth slot is clicked on. We have two assumptions: first, the more the slot is on the top, the higher the probability that the slot is clicked on, and, second, the click-through rates do not depend on the occupant of the slot.

It is worth noting that sponsored-search auctions were responsible for 98% revenue of Google in 2006, so these auctions have been really important to the Internet economy.

An elegant way how to resolve the potential conflict of interest between the mechanism designer and the bidder is to ensure that each bidder wants to bid truthfully. That is, to ensure that the mechanism (x, p) has the DSIC property. Then each bidder has an obvious dominant strategy and thus the auction becomes much more predictable for the seller.

Thus, we want to identify those allocation rules x for which we can find payment rules p such that the resulting mechanism (x, p) has the DSIC property. To do so, we first introduce some definitions.

Definition 3.6 (Implementable allocation rule[10])**.** *An allocation rule x for a single-parameter environment is* implementable *if there is a payment rule p such the mechanism (x, p) is DSIC.*

Thus, we want to identify implementable allocation rules, as these extend to DSIC mechanisms. We note that the allocation rules of DSIC mechanisms form the set of implementable allocation rules. To give an example, we know that the allocation rule "give the item to the bidder with the highest bid" is implementable in the case of single-item auctions, as the second-price rule provides DSIC mechanism. On the other hand, the situation is much less clear for the allocation rule "give the item to the bidder with the second highest bid". We have not seen a payment rule that would extend it to DSIC mechanism and it seems difficult to argue that the no payment rule should not work.

Definition 3.7 (Monotone allocation rule[11])**.** *An allocation rule x for a single-parameter environment is* monotone *if, for every bidder i and all bids b_{-i} of the other bidders, the allocation $x_i(z; b_{-i})$ to i is nondecreasing in his bid z. Here, $x_i(z; b_{-i})$ denotes the function x_i applied to a vector that is obtained from b by replacing the ith coordinate with z.*

That is, in a monotone allocation rule, bidder i can only get more goods by bidding higher. Usually, it is not difficult to check whether an allocation rule is monotone. For example, the single-item auction allocation rule "give the item to the bidder with the highest bid" is monotone. If a bidder raises his bid while other bids remain the same, he cannot lose the auction if he is the winner before raising his bid. In contrast, the allocation rule "give the item to the bidder with the second highest bid" is not monotone, as raising a bid can lose the auction to a bidder.

The following result, *Myerson's lemma*, is a powerful tool for designing DSIC mechanisms.

Theorem 3.8 (Myerson's lemma [Mye81])**.** *In a single-parameter environment, the following three claims hold.*

*(*a*) An allocation rule is implementable if and only if it is monotone.*

*(*b*) If an allocation rule x is monotone, then there exists a unique payment rule p such that the mechanism (x, p) is DSIC (assuming that $b_i = 0$ implies $p_i(b) = 0$).*

*(*c*) The payment rule p is given by the following explicit formula*

$$p_i(b_i; b_{-i}) = \int_0^{b_i} z \cdot \frac{\mathrm{d}}{\mathrm{d}z} x_i(z; b_{-i}) \, \mathrm{d}z$$

for every $i \in \{1, \ldots, n\}$.

---

[10] Implementovatelné alokační pravidlo.
[11] Monotónní alokační pravidlo.

Myerson's lemma says that implementable allocation rules and monotone allocation rules form exactly the same class. This equivalence is quite powerful, as implementable allocation rules describe our design goal but are difficult to verify, while the monotonicity property is usually easy to check. The second part states that, when an allocation rule is implementable, there is only one way how to assign payments to achieve the DSIC property (assuming bidding zero guarantees zero payment, but this follows from our assumption $p_i(b) \in [0, b_i \cdot x_i(b)]$). Moreover, there is a relatively simple and explicit formula for this payment rule.

Sketch of the proof of Myerson's lemma (Theorem 3.8). We prove all three claims at once. First, let x be an allocation rule and p be a payment rule. We recall that the mechanism (x, p) is DSIC if every bidder maximizes his utility by setting $b_i = v_i$ and, moreover, his utility $u_i(b) = v_i \cdot x_i(b) - p_i(b)$ is guaranteed to be non-negative. Expressing the utility of bidder i as a function of his bid z, we write $u_i(z; b_{-i}) = v_i \cdot x_i(z; b_{-i}) - p_i(z; b_{-i})$.

Assume the mechanism (x, p) is DSIC. The DSIC property says $u_i(z; b_{-i}) = v_i \cdot x_i(z; b_{-i}) - p_i(z; b_{-i}) \leq v_i \cdot x_i(v_i; b_{-i}) - p_i(v_i; b_{-i})$ for every z. We use a clever swapping trick. For two possible bids y and z with $0 \leq y < z$, bidder i might as well have private valuation z and can submit the false bid y if he wants, thus the DSIC condition gives

$$u_i(y; b_{-i}) = z \cdot x_i(y; b_{-i}) - p_i(y; b_{-i}) \leq z \cdot x_i(z; b_{-i}) - p_i(z; b_{-i}) = u_i(z; b_{-i}). \qquad (3.1)$$

Analogously, bidder i may have his private valuation $v_i = y$ and can submit the false bid z and thus the mechanism (x, p) must satisfy

$$u_i(z; b_{-i}) = y \cdot x_i(z; b_{-i}) - p_i(z; b_{-i}) \leq y \cdot x_i(y; b_{-i}) - p_i(y; b_{-i}) = u_i(y; b_{-i}). \qquad (3.2)$$

By rearranging inequalities (3.1) and (3.2) and putting them together, we obtain the following inequality called the *payment difference sandwich*:

$$z(x_i(y; b_{-i}) - x_i(z; b_{-i})) \leq p_i(y; b_{-i}) - p_i(z; b_{-i}) \leq y(x_i(y; b_{-i}) - x_i(z; b_{-i})). \qquad (3.3)$$

Since $0 \leq y < z$, we obtain, by ignoring the middle part of this inequality, $x_i(y; b_{-i}) \leq x_i(z; b_{-i})$. Thus, if the mechanism (x, p) is DSIC, then x is monotone.

In the rest of the proof, we assume that the mechanism x is monotone. Let i and b_{-i} be fixed, so, in particular, we consider x_i and p_i as functions of z. First, we also assume that the function x_i is piecewise constant. Thus, the graph of x_i consists of a finite number of intervals with "jumps" between consecutive intervals; see Figure 3.1.
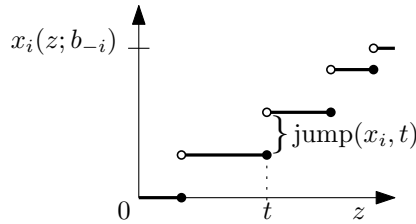

Figure 3.1: A piecewise constant function.

For a piecewise constant function f, we use $\mathrm{jump}(f, t)$ to denote the magnitude of the jump of f at point t. If we fix z in the "payment difference sandwich" inequality (3.3) and let y approach z from below in this inequality, then both sides become 0 if there is no jump of x_i at z (that is, if $\mathrm{jump}(x_i, z) = 0$). If $\mathrm{jump}(x_i, z) = h > 0$, then both sides tend to $z \cdot h$. Thus, if the mechanism (x, p) is supposed to be DSIC, then the following constraint on p must hold for every z:

$$\mathrm{jump}(p_i, z) = z \cdot \mathrm{jump}(x_i, z).$$

If we combine this constraint with the initial condition $p_i(0; b_{-i}) = 0$, we obtain a formula for the payment function p for every bidder i and bids b_{-i} of other bidders,

$$p_i(b_i; b_{-i}) = \sum_{j=1}^{\ell} z_j \cdot \mathrm{jump}(x_i(\,\cdot\,; b_{-i}), z_j), \qquad (3.4)$$

where $z_1, \dots, z_\ell$ are the breakpoints of the allocation function $x_i(\,\cdot\,; b_{-i})$ in the interval $[0, b_i]$.

With some additional facts from calculus, this argument can be generalized to general monotone functions x_i. We omit the details here and only sketch the idea for differentiable x_i. If we divide the "payment difference sandwich" inequality (3.3) by $z - y$ and take the limit of the resulting function as z approaches y from above, then we obtain the constraint

$$p_i'(y; b_{-i}) = y \cdot x_i'(y; b_{-i}).$$

Combining this constraint with the initial condition $p_i(0; b_{-i}) = 0$, we obtain the formula

$$p_i(b_i; b_{-i}) = \int_0^{b_i} z \cdot \frac{\mathrm{d}}{\mathrm{d}z} x_i(z; b_{-i})\, dz$$

for every z. Note that we showed that this is the only possibility for the function p if we want to extend the allocation rule x to a DSIC mechanism (x, p).

It remains to show that if x is monotone, then the mechanism (x, p) is indeed DSIC. This argument works also for monotone allocation rules that are not necessarily piecewise constant. However, for the sake of clarity, we present it only for piecewise constant functions. The proof is illustrated in Figure 3.2. We recall that the utility of bidder i satisfies $u_i(b_i; b_{-i}) = v_i \cdot x_i(b_i; b_{-i}) - p_i(b_i; b_{-i})$ when he bids b_i and the other bidders bid b_{-i}. The value $v_i \cdot x_i(b_i; b_{-i})$ is represented by a blue rectangle in Figure 3.2. Using the expression (3.4), we see that the payment $p_i(b_i; b_{-i})$ of bidder i corresponds to the part of $[0, b_i] \times [0, x_i(b_i; b_{-i})]$ lying to the left of the curve $x_i(\,\cdot\,; b_{-i})$; this is represented by the red areas in Figure 3.2. Clearly, it is optimal for bidder i to bid $b_i = v_i$. Otherwise he either overbids $b_i > v_i$, in which case his utility is smaller by the area above $x_i(\,\cdot\,; b_{-i})$ in the range $[v_i, b_i]$, or he underbids $b_i < v_i$, in which case his utility is smaller by the area below $x_i(\,\cdot\,; b_{-i})$ in the range $[0, v_i]$; see Figure 3.2. $\qquad\square$

3.1.3 Some applications of Myerson's lemma

In this subsection, we give some applications of Myerson's lemma (Theorem 3.8) to illustrate its strength and, in particular, to show the use of the explicit payment rule. In the case of piecewise constant allocation function, this payment rule for bidder i and bids b_{-i} of other bidders becomes

$$p_i(b_i; b_{-i}) = \sum_{j=1}^{\ell} z_j \cdot \mathrm{jump}(x_i(\,\cdot\,; b_{-i}), z_j),$$

where $z_1, \dots, z_\ell$ are the breakpoints of the allocation function $x_i(\,\cdot\,; b_{-i})$ in the interval $[0, b_i]$.

Single-item auctions. We start with single-item auctions mentioned in Example 3.4. Recall that, in single-item auctions, n bidders compete for a single item of the seller. Each bidder either gets the item or not, but only one bidder can get it. The goal is to design the auction so that the bidder with the highest valuation v_i wins. Let bidder i and bids b_{-i} of the other bidders be fixed and set $B = \max_{j \in \{1, \dots, n\} \setminus \{i\}} b_j$. If we allocate the item to the highest bidder, then the allocation function $x_i(\,\cdot\,; b_{-i})$ is 0 up to B and 1 thereafter. Clearly, this allocation rule is monotone.

If i is the highest bidder, then $b_i > B$ and the (unique) payment formula from Myerson's lemma becomes $p_i(b_i; b_{-i}) = B$ and the utility of i is $v_i \cdot x_i(b_i; b_{-i}) - B = v_i - B$; see Figure 3.3. Otherwise, $b_i \le B$ and the payment function and the utility of i is zero. If $v_i > B$, then his utility is positive and the utility of all other bidders is zero. It follows from the form of p_i that the utility of i is maximized when $v_i = b_i$. Altogether, we obtain the second-price payment rule discussed in Section 3.1.1.
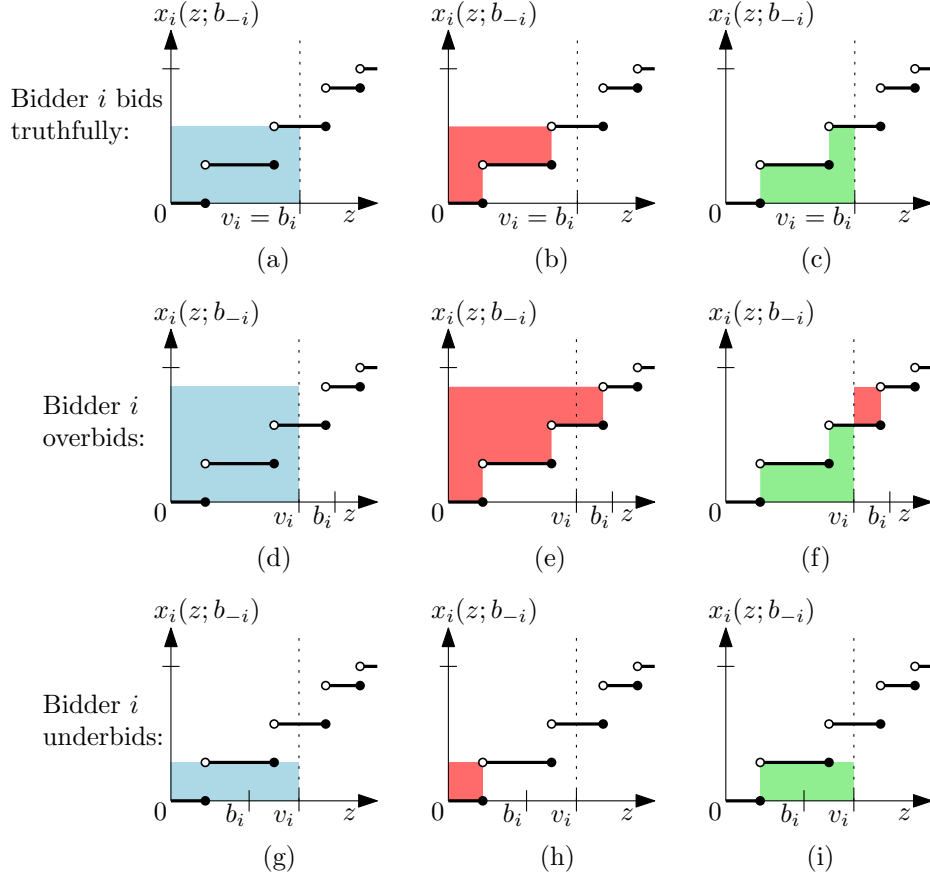
Figure 3.2: A sketch of the proof of Myerson's lemma. The blue areas represent the value $v_i \cdot x_i(b_i; b_{-i})$, the red areas represent the payment $p_i(b_i; b_{-i})$, and the green parts correspond to the utility $u_i(b_i; b_{-i}) = v_i \cdot x_i(b_i, b_{-i}) - p_i(b_i; b_{-i})$ of bidder i. Parts (a)–(c) represent the situation when bidder i bids $b_i = v_i$, parts (d)–(f) correspond to a situation when i overbids, that is, $b_i > v_i$. Parts (g)–(i) illustrate the case when bidder i underbids, that is, $b_i < v_i$. Clearly, the utility is largest if $b_i = v_i$.

Sponsored-search auctions.   Recall that, in a sponsored-search auction, we have n bidders competing for k positions. Each position j has a *click-through-rate* α_j, where $\alpha_1 > \cdots > \alpha_k > 0$. The feasible set X consists of vectors $(x_1, \ldots, x_n)$, where each x_i lies in $\{\alpha_1, \ldots, \alpha_k, 0\}$ and if $x_i = x_j$, then $x_i = 0 = x_j$. The goal is to maximize social surplus. Let x be the allocation rule that assigns the ith best slot to the ith highest bidder. The rule x is then monotone, as one can easily verify, and, assuming truthful bids, x is also maximizing social surplus. By Myerson's lemma (Theorem 3.8), there is a unique and explicit formula for a payment rule p such that the mechanism (x, p) is DSIC. This formula can be described as follows. Assume without loss of generality that bidder i bids the ith highest bid, that is, $b_1 \geq \cdots \geq b_n$. Consider bidder 1. Imagine that he increases his bid z from 0 to b_1, while other bids are fixed. The allocation function $x_1(z; b_{-1})$ increases from 0 to α_1 as z increases from 0 to b_1, with a jump of $\alpha_j - \alpha_{j+1}$ at the point where z becomes the jth highest bid in the profile $(z; b_{-1})$, namely b_{j+1}. In general, for ith highest bidder, Myerson's lemma gives the payment formula

$$p_i(b) = \sum_{j=i}^{k} b_{j+1}(\alpha_j - \alpha_{j+1}),$$

where $\alpha_{k+1} = 0$. The per-click payment for bidder i is then this value simply scaled by $1/\alpha_i$, that is, $\frac{1}{\alpha_i} \sum_{j=i}^{k} b_{j+1}(\alpha_j - \alpha_{j+1})$. Thus, when the bidder's link is clicked, he pays a suitable
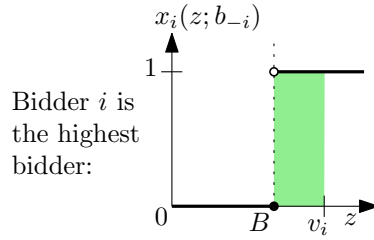
Figure 3.3: Myerson's lemma in case of single-item auctions.

convex combination of the lower bids. Altogether, we see that the natural greedy algorithm in allocation is optimal.

3.1.4 Knapsack auctions

In this subsection, we introduce another type of auctions, so-called *knapsack auctions*. These auctions are single-parameter environments, so we can use Myerson's lemma again to design a DSIC mechanism for them. However, as we will see, although knapsack auctions admit DSIC mechanism, they are not awesome in the sense of Definition 3.1.

Definition 3.9 (Knapsack auction[12])**.** *In a* knapsack auction *of n bidders $1, \ldots, n$, each bidder i has two parameters: publicly known* size $w_i \geq 0$ *and a private* valuation $v_i \geq 0$. *There is a single seller who has a* capacity $W \geq 0$. *The feasible set X consists of $\{0, 1\}$-vectors $(x_1, \ldots, x_n)$ such that $\sum_{i=1}^{n} x_i w_i \leq W$, where $x_i = 1$ indicates that bidder i is a winning bidder.*

That is, knapsack auctions are auctions where multiple bidders might win, but there is a limited capacity. As an example, we might imagine that the bidders are companies such that each company has its own TV commercial of length w_i and is willing to pay v_i in order to have the commercial presented during a commercial break. The seller is the television company and the capacity W is the length of the commercial break. Other examples include bidders who want files stored on a shared server, data streams sent through a shared communication channel, or processes to be executed on a shared supercomputer. Note that a k-*item auction* is a special case of a knapsack auction, where $w_i = 1$ for each bidder i and $W = k$.

We now try to design a mechanism for knapsack auctions so that we satisfy all three conditions from the definition of an awesome auction (Definition 3.1). Recall that we would like to have a DSIC property, maximize the social surplus, and implement the auction in polynomial time.

Since we want to maximize the social surplus $\sum_{i=1}^{n} v_i x_i$, the choice of the allocation rule $x = (x_1, \ldots, x_n) \in X$ is determined. For bids $b = (b_1, \ldots, b_n)$, we choose $x(b)$ from X such that $\sum_{i=1}^{n} b_i x_i$ is maximized. Then, when bidders bid truthfully (that is, $b_i = v_i$ for every i), the social surplus is maximized. Note that the problem of finding such x solves an instance of the *Knapsack problem*: given a capacity W and n items of values $v_1, \ldots, v_n$ and sizes $w_1, \ldots, w_n$, find a subset of the items having a maximum total value such that the total size is at most W.

Now, having the allocation rule x, it remains to find an appropriate payment rule p so that the resulting mechanism (x, p) is DSIC. To do so, we apply Myerson's lemma. It is not difficult to verify that the allocation rule x is monotone and thus Myerson's lemma (Theorem 3.8) gives us a payment rule p such that (x, p) is DSIC. Moreover, since the allocation rule x is very simple (it is 0 until some point z in which it jumps to 1), we know that p is defined as follows. If $b_i < z$, then bidder i pays nothing, otherwise he pays $z \cdot (1 - 0) = z$ (for fixed bids b_{-i} of the other bidders). That is, bidder i pays the lowest bid he could make and continues to win (holding the other bids b_{-i} fixed).

Thus, we have satisfied the first two conditions from the definition of an awesome auction. However, assuming $P \neq NP$, we cannot satisfy the third condition (implementing the auction in polynomial time), since the Knapsack problem is known to be NP-hard, and choosing the

---

[12] Batohová aukce.

allocation rule so that the social surplus is maximized solves this problem. Thus, knapsack auctions are not awesome.

Relaxing the first condition (DSIC property) does not help, since it is the last two conditions that collide. We might relax the third condition since it is known that the allocation rule can be implemented in pseudopolynomial time using dynamic programming. In general, relaxing this condition is helpful if our instances are small or structured enough and we have enough time and computing power to implement optimal surplus-maximization. Then the resulting allocation rule is monotone and can be extended to a DSIC mechanism.

The dominant paradigm in algorithmic mechanism design is to relax the second constraint (optimal surplus) as little as possible, subject to the first (DSIC) and the third (polynomial-time) constraints. For single-parameter environments, Myerson's Lemma implies that the following goal is equivalent: design a polynomial-time and monotone allocation rule that comes as close as possible to maximizing the social surplus.

This approach resembles the primary goal in approximation algorithms: to design algorithms for NP-hard problems that are as close to optimal as possible, subject to a polynomial-time constraint. Algorithmic mechanism design for single-parameter problems has exactly the same goal, except the algorithms must additionally obey a monotonicity constraint. The "holy grail" in algorithmic mechanism design is as follows: for as many NP-hard problems of interest as possible, match the best-known (or even best possible) approximation guarantee for (not necessarily monotone) approximate surplus maximization algorithms, subject to P $\neq$ NP. That is, we would like the DSIC/monotone constraint to cause no additional surplus loss, beyond the loss we already have to suffer due to the polynomial-time constraint. So far this was not an issue, as, with exact surplus-maximization, the DSIC/monotone constraint was satisfied "for free", and exact surplus-maximization with unknown data reduced to exact surplus-maximization with known data.

We now illustrate this approach in the specific setting of knapsack auctions by designing an allocation rule that will lead to at least one half of the optimum social surplus, assuming truthful bids. We assume without loss of generality that there is no bidder i with $w_i > W$, since we can discard such bidders with no effect on the optimal solution. We also assume that the bidders $1, \ldots, n$ are sorted in the order $<$ so that

$$\frac{b_1}{w_1} \geq \cdots \geq \frac{b_n}{w_n}.$$

Consider the following *greedy allocation rule* $x^G = (x_1^G, \ldots, x_n^G) \in X$, which for given bids $b = (b_1, \ldots, b_n)$ selects a subset of bidders so that $\sum_{i=1}^n x_i^G w_i \leq W$ using the following procedure.

1. Pick winners in the order $<$ until one does not fit and then halt.

2. Return either the solution from the first step or the highest bidder, whichever creates more social surplus.

The reason for the second step is that the solution in the first step might be highly suboptimal if there is a very valuable and very large bidder. Consider, for example, $n = 2$ with $b_1 = 2$, $w_1 = 1$, $b_2 = W$, and $w_2 = W$ for a very large W.

Theorem 3.10. *Assuming truthful bids, the social surplus of the greedy allocation rule x^G is at least one half of the maximum possible social surplus.*

Proof. Let $w_1, \ldots, w_n$ be the given sizes, $v_1, \ldots, v_n$ the valuations, which are also the bids, as bidders bid truthfully, and W be the capacity. First, we consider a relaxation of the problem, where a bidder can be chosen fractionally with its value pro-rated accordingly. That is, we can choose each bidder i with fraction $\alpha_i \in [0, 1]$ so that i contributes with $\alpha_i \cdot v_i$ to the solution. The greedy algorithm to solve this fractional version of the problem is as follows: pick winners in the order $<$ until the entire capacity W is fully used with the possibility to pick the last winner fractionally, if needed.

We argue that this algorithm maximizes the surplus over all feasible solutions to the fractional knapsack problem using a straightforward exchange argument. Let $1, \ldots, k$ be the winners selected by the greedy algorithm and suppose for contradiction that there is another feasible solution that gives higher social surplus. Then our solution can be improved by changing one of the constraints α_i to some larger β_i. Since $\alpha_1 = \cdots = \alpha_{k-1} = 1$, we have $i \geq k$. Since $\sum_{l=1}^{k} \alpha_l w_l = W$, there is a $j \in \{1, \ldots, k\}$ with $j < i$ and $\beta_j < \alpha_j$. We can assume that these two coefficients are the only changed ones; see Exercise 3.5. Then $(\beta_i - \alpha_i)w_i \leq (\alpha_j - \beta_j)w_j$, as $\sum_{l=1}^{k} \alpha_l w_l = W$ and we add $(\beta_i - \alpha_i)w_i$ to the size, while removing $(\alpha_j - \beta_j)w_j$. On the other hand, since the social surplus is now larger, we have $(\beta_i - \alpha_i)v_i > (\alpha_j - \beta_j)v_j$. By dividing the left side of the second inequality with the left side of the first inequality and doing the same for the right sides, we obtain $v_i/w_i > v_j/w_j$, which, since $j < i$, contradicts our choice of the order $<$.

Now, assume that in the fractional setting the first $k-1$ winners i have $\alpha_i = 1$ while the kth winner won only fractionally, that is, $\alpha_k < 1$. Then the social surplus achieved by step 1 is exactly $\sum_{i=1}^{k-1} \alpha_i v_i = \sum_{i=1}^{k-1} v_i$. The social surplus achieved by step 2 is at least v_k. Thus, both steps together yield a social surplus of size at least $\max\{v_k, \sum_{i=1}^{k-1} v_i\}$. This is at least half of the surplus of the optimal fractional solution, which is at least the surplus of an optimal (non-fractional) solution to the original problem. $\qquad\square$

It can be shown that the greedy allocation rule x^G is monotone; see Exercise 3.5. Thus, by Myerson's lemma (Theorem 3.8), there is a payment rule that together with the greedy allocation rule forms a DSIC mechanism. In particular, this justifies the assumption about truthful bidders in Theorem 3.10.

3.1.5 Exercises

Exercise 3.1. *Consider a single-item auction with at least three bidders. Prove that selling the item to the highest bidder at a price equal to the third-highest bid yields an auction that is not dominant-strategy incentive compatible (DSIC).* **[2]**

Exercise 3.2. *Assume there are k identical items and $n > k$ bidders. Also, assume that each bidder can receive at most one item. What is the analog of the second-price auction? Prove that your auction is DSIC.* **[2]**

Exercise 3.3. *Use Myerson's Lemma (Theorem 3.8) to prove that the Vickrey auction is the unique single-item auction that is DSIC, always awards the good to the highest bidder, and charges the other bidders 0.* **[2]**

Exercise 3.4. *Give a purely algebraic proof that coupling a monotone and piecewise constant allocation rule x with the following payment rule*

$$p_i(b_i; b_{-i}) = \sum_{j=1}^{\ell} z_j \cdot \mathrm{jump}(x_i(\,\cdot\,; b_{-i}), z_j),$$

where $z_1, \ldots, z_\ell$ are the breakpoints of the allocation function $x_i(\,\cdot\,; b_{-i})$ in the interval $[0, b_i]$, yields a DSIC mechanism. **[2]**

Exercise 3.5. (a) *Prove that the Knapsack auction allocation rule x^G induced by the greedy $(1/2)$-approximation algorithm is monotone.* **[1]**

(b) *Prove that it suffices to change only two coefficients α_i and β_j in the proof of Theorem 3.10 (the correctness of the $(1/2)$-approximation algorithm that is based on x^G).* **[2]**

3.2 Revenue-Maximizing Auctions

So far we have focused on designs that maximize the social surplus $\sum_{i=1}^{n} v_i x_i(b)$ over all feasible outcomes $(x_1, \ldots, x_n)$ of some set X. We did not care much about the *revenue* $\sum_{i=1}^{n} p_i(b)$.

Payments were present even in Vickrey's auctions, but they served only to encourage the bidders to bid truthfully so that we receive DSIC property. In this section, we focus on auctions that maximize the revenue as their primary goal.

Let us first state a simple example that illustrates that revenue maximization is not as simple as social surplus maximization. Consider a single-item auction with a single bidder, whose valuation v is unknown to the designer of the auction. With only one bidder, the space of all DSIC auctions we can run is quite small, all the seller can do is to offer the bidder the take-it-or-leave-it offer, since the allocation function is monotone (by Myerson's lemma) and thus there is a value r for which the bidder gets the item for the first time. This auction works as follows: the seller posts a price r, then his revenue is either r if $v \geq r$ and 0 otherwise. Thus, maximizing the social surplus is trivial, as then the seller just puts $r = 0$, not caring what v is. However, when maximizing the revenue, it is not clear how the mechanism designer should set r, since he does not know the valuation v. The main issue is that different auctions do better on different inputs. For example, with a single item and a bidder, a posted price $r = 10$ will do very well on inputs with v not much larger than 10, and terribly on inputs with $v < 10$.

Thus, we need a model that helps us to reason about trade-offs between our performance on different inputs. We consider the following most classical model, called *Bayesian model* or the *average case analysis*. We will have a probability distribution over inputs (the private valuations of the bidders) and we will seek to perform as well as possible on expected revenue. Formally, the model consists of

(a) a single-parameter environment (x, p). That is, there are n bidders with private valuations $(v_1, \ldots, v_n)$, an allocation rule x assigning outcomes from a feasible set $X \subseteq \mathbb{R}^n$ of outcomes, and a payment rule p.

(b) For each bidder i, the private valuation v_i of i is drawn from a probability distribution F_i with density function f_i and with support contained in $[0, v_{max}]$. That is, $F_i(z)$ is the probability that v_i from F_i has value at most z. We assume that the distributions $F_1, \ldots, F_n$ are independent, but not necessarily the same.

(c) The distributions $F_1, \ldots, F_n$ are known to the mechanism designer. The bidders do not know the distributions $F_1, \ldots, F_n$. Actually, since we discuss only DSIC auctions, the bidders do not need to know the distributions $F_1, \ldots, F_n$, as they have dominant strategies.

The goal is to maximize, among all DSIC auctions, the expected revenue

$$\mathbb{E}_{v=(v_1,\ldots,v_n)\sim(F_1\times\cdots\times F_n)}\left[\sum_{i=1}^{n} p_i(v)\right], \tag{3.5}$$

where the expectation is taken with respect to the given distribution $F_1 \times \cdots \times F_n$ over the valuations $(v_1, \ldots, v_n)$.

We recall that if F is a probability distribution with density f and with support $[0, v_{max}]$, then $f(x) = \frac{\mathrm{d}}{\mathrm{d}x}F(x)$ and $F(x) = \int_0^x f(z)\,\mathrm{d}z$. Also, for a random variable X, we have $\mathbb{E}_{z\sim F}[X(z)] = \int_0^{v_{max}} X(z) \cdot f(z)\,\mathrm{d}z$.

Example 3.11 (Single-bidder single-item auction in the Bayesian model). *We return to the previous example with a single bidder in a single-item auction. It now becomes much easier to handle, as the expected revenue of a posted price r equals $r \cdot (1 - F_1(r))$, since $(1 - F_1(r))$ is the probability that v is at least r. Given a probability distribution F_1, it is usually a simple matter to maximize this value. For example, if F_1 is the* uniform distribution *on $[0, 1]$, where $F_1(x) = x$ for every $x \in [0, 1]$, then we achieve the maximum revenue $1/4$ by choosing $r = 1/2$. The posted price r that makes the expected revenue as high as possible is called the* monopoly price.

Example 3.12 (Two-bidders single-item auction in the Bayesian model). *Consider a single-item auction with two bidders that have their valuations v_1 and v_2 drawn uniformly from $[0, 1]$. Now, there are more DSIC auctions to think of, for example, the Vickrey's auction, for which the expected revenue is $1/3$; see part (a) of Exercise 3.6). Besides Vickrey's auctions, there are other*

DSIC auctions and some may perform better with respect to the expected revenue; see part (b) of Exercise 3.6).

3.2.1 Maximizing expected revenue

We now characterize DSIC auctions that maximize the expected revenue. We already have a formula for the expected revenue (see (3.5)), but it is not very useful, as it is not clear from it what auctions make the revenue as large as possible. As a main result of this section, we find another, much more useful, formula for the expected revenue.

Theorem 3.13. *Let (x, p) be a DSIC mechanism in a single-parameter environment forming a Bayesian model with n bidders, probability distributions $F_1, \ldots, F_n$, and density functions $f_1, \ldots, f_n$. Let $F = F_1 \times \cdots \times F_n$. Then*

$$\mathbb{E}_{v \sim F} \left[\sum_{i=1}^n p_i(v) \right] = \mathbb{E}_{v \sim F} \left[\sum_{i=1}^n \varphi_i(v_i) \cdot x_i(v) \right],$$

where

$$\varphi_i(v_i) = v_i - \frac{1 - F_i(v_i)}{f_i(v_i)}$$

is the virtual valuation *of bidder i with valuation v_i drawn from F_i.*

In other words, for every DSIC auction, the expected revenue is equal to the expected *virtual social surplus* $\sum_{i=1}^n \varphi_i(v_i) \cdot x_i(v)$. Note that the virtual valuation $\varphi_i(v_i)$ of a bidder can be negative and that it depends on the valuation v_i of the bidder i and his distribution F_i, and not on those of the others. For example, consider a bidder i with valuation v_i drawn from the uniform distribution on $[0, 1]$. Then $F_i(z) = z$, $f_i(z) = 1$, and $\varphi_i(z) = z - (1 - z)/1 = 2z - 1 \in [-1, 1]$. We can also give some coarse intuition behind the virtual valuations $\varphi_i(v_i)$. Recall that $\varphi_i(v_i) = v_i - (1 - F_i(v_i))/f_i(v_i)$. The first term v_i in this expression can be thought of as the maximum revenue obtainable from bidder i and the second term as the inevitable revenue loss caused by not knowing v_i in advance.

Proof of Theorem 3.13. Since the mechanism (x, p) is DSIC, we can assume $b_i = v_i$ for every bidder i. The proof is an application of Myerson's lemma (Theorem 3.8). By this lemma, we have the following payment formula:

$$p_i(b_i; b_{-i}) = \int_0^{b_i} z \cdot \frac{d}{dz} x_i(z; b_{-i}) \, dz.$$

As noted in Section 3.1.2, it can be shown by results from calculus that this formula holds for any monotone function $x_i(z; b_{-i})$, if we suitably interpret the derivation $\frac{d}{dz} x_i(z; b_{-i})$. Similarly, all arguments that we use later in the proof, such as integration by parts, can be made rigorous for bounded monotone functions, but we omit the details.

In particular, Myerson's payment formula says that the payments are determined by the allocation rules x_i. We now fix a bidder i and the bids v_{-i} of other bidders. We can also use Myerson's payment formula to express the expected payment by bidder i for a given v_{-i} as

$$\mathbb{E}_{v_i \sim F_i} [p_i(v)] = \int_0^{v_{max}} p_i(v) f_i(v_i) \, dv_i = \int_0^{v_{max}} \left(\int_0^{v_i} z \cdot \frac{d}{dz} x_i(z; v_{-i}) \, dz \right) f_i(v_i) \, dv_i,$$

where the first equality is just expanding the expectation and the second one is expanding $p_i(v)$ using Myerson's payment formula.

As a next step, we reverse the integration order and obtain

$$\int_0^{v_{max}} \left(\int_0^{v_i} z \cdot \frac{d}{dz} x_i(z; v_{-i}) \, dz \right) f_i(v_i) \, dv_i = \int_0^{v_{max}} \left(\int_z^{v_{max}} f_i(v_i) \, dv_i \right) z \cdot \frac{d}{dz} x_i(z; v_{-i}) \, dz.$$

We have used the fact that $z \leq v_i$.

Since f_i is the density function, the inner integral can be simplified and we can rewrite the above expression as

$$\int_0^{v_{max}} (1 - F_i(z)) \cdot z \cdot \frac{d}{dz} x_i(z; v_{-i}) \, dz.$$

We now apply integration by parts, that is $\int f g' = f g - \int f' g$, choosing $f(z) = (1 - F_i(z)) \cdot z$ and $g'(z) = \frac{d}{dz} x_i(z; v_{-i})$, rewriting the expression as

$$[(1 - F_i(z)) \cdot z \cdot x_i(z; v_{-i})]_0^{v_{max}} - \int_0^{v_{max}} x_i(z; v_{-i}) \cdot (1 - F_i(z) - z f_i(z)) \, dz.$$

The first term is zero, as it equals to 0 for $z = 0$ and also for $z = v_{max}$, as v_{max} is a finite number. The second term can be rewritten so that we obtain

$$\int_0^{v_{max}} \left(z - \frac{1 - F_i(z)}{f_i(z)} \right) x_i(z; v_{-i}) f_i(z) \, dz = \int_0^{v_{max}} \varphi_i(z) x_i(z; v_{-i}) f_i(z) \, dz$$

by the definition of $\varphi_i(v_i)$.

This final expression as an expected value where z is drawn from F_i and we have thus derived, for every v_{-i},

$$\mathbb{E}_{v_i \sim F_i} [p_i(v_i; v_{-i})] = \mathbb{E}_{v_i \sim F_i} [\varphi_i(v_i) \cdot x_i(v_i; v_{-i})].$$

Taking the expectation with respect to v_{-i} on both sides of this equality ("integrating out over everybody else's valuations"), we obtain

$$\mathbb{E}_{v \sim F} [p_i(v)] = \mathbb{E}_{v \sim F} [\varphi_i(v_i) \cdot x_i(v)].$$

Finally, applying the linearity of expectation twice, we get the final form

$$\mathbb{E}_{v \sim F} \left[\sum_{i=1}^n p_i(v) \right] = \sum_{i=1}^n \mathbb{E}_{v \sim F} [p_i(v)] = \sum_{i=1}^n \mathbb{E}_{v \sim F} [\varphi_i(v_i) \cdot x_i(v)] = \mathbb{E}_{v \sim F} \left[\sum_{i=1}^n \varphi_i(v_i) \cdot x_i(v) \right].$$

$\square$

Note that if we omitted the virtual valuations from the derived expression, we would obtain the expected social surplus, which we already know how to optimize. Thus, for DSIC auctions, we can interpret the statement of Theorem 3.13 as "maximizing expected revenue = maximizing expected virtual social surplus". Note that, somewhat surprisingly, even though we are interested in the payment rule p, the formula tells us that it suffices to deal with the allocation function x.

3.2.2 Maximizing expected virtual social surplus

By Theorem 3.13, maximizing the expected revenue in DSIC auctions is the same as maximizing the expected virtual social surplus. In this section, we characterize the auctions that do that.

We start with a simpler setting by considering only single-item auctions of n bidders, that is we have $\sum_{i=1}^n x_i(v) \leq 1$ for each v and $x_i(v) \in \{0, 1\}$. We also assume that there is a probability distribution F such that $F_1 = \cdots = F_n = F$ and thus all virtual valuations $\varphi_1, \ldots, \varphi_n$ are the same, say equal to a function φ (recall that we are still assuming that the probability distributions $F_1, \ldots, F_n$ are independent). Finally, we assume that the probability distribution F is *regular*, that is, the corresponding virtual valuation function $\varphi(v) = v - (1 - F(v))/f(v)$ is strictly increasing in v. All these conditions are satisfied, for example, if F is the uniform probability distribution on $[0, 1]$, as then $\varphi(v) = 2v - 1$, as we have already seen.

We show that for such auctions the Vickrey auction with reserved price is the optimal auction format for maximizing expected revenue. In a *Vickrey auction with reserve price r*, the allocation rule awards the item to the highest bidder, unless all bids are less than r, in which case no one gets the item. The corresponding payment rule charges the winner (if any) the second-highest bid or r, whichever is larger. We remark that Vickrey auctions with reserved prices correspond to eBay auctions with an opening bid.

Proposition 3.14. *Let F be a regular probability distribution with density f and the virtual valuation φ and let $F_1, \ldots, F_n$ be independent probability distributions on valuations of n bidders such that $F = F_1 = \cdots = F_n$ (and thus $\varphi = \varphi_1 = \cdots = \varphi_n$). Then Vickrey auction with reserve price $\varphi^{-1}(0)$ maximizes the expected revenue.*

Proof. By Theorem 3.13, maximizing the expected revenue is the same as maximizing the expected virtual social surplus. To maximize expected virtual social surplus, we have the freedom of choosing $x(v)$ for each input v (subject to feasibility constraints $(x_1, \ldots, x_n) \in X$), while we do not have any control over the probability distribution F and the virtual values $\varphi(v_i)$. In a single-item auction, the feasibility constraint is $\sum_{i=1}^{n} x_i(v) \leq 1$ for each v, so to maximize the expected virtual social surplus, we just give the item to bidder i with the highest virtual valuation $\varphi(v_i)$. This is the so-called *virtual social surplus-maximizing mechanism*. However, the virtual valuations can be negative, so if every bidder has a negative virtual valuation then the expected virtual social surplus is maximized by not awarding the item to anyone.

So we have an allocation rule that maximizes the expected virtual social surplus over all allocation rules (monotone or not). It remains to show that this allocation rule is monotone, as then, by Myerson's lemma (Theorem 3.8), it can be extended to a DSIC auction (x, p) by choosing an appropriate payment rule p. This is where we use the fact that F is regular, as then the virtual valuation function φ shared by all bidders is strictly increasing. Thus, it follows that our allocation rule is thus equivalent to the Vickrey auction with a reserve price $\varphi^{-1}(0)$. $\quad\square$

Thus, under our assumptions, eBay with a suitable opening bid is the optimal auction format. A natural question is whether we can weaken the assumptions in the statement of Proposition 3.14. For an arbitrary single-parameter environment and probability distributions $F_1, \ldots, F_n$, the allocation rule that maximizes the expected revenue is defined as that, for an input v, chooses the feasible allocation that maximizes the expected virtual social surplus $\sum_{i=1}^{n} \varphi(v_i) x_i(v)$. If every F_i is regular, then this allocation rule is monotone; see Exercise 3.8. Thus, by Myerson's lemma (Theorem 3.8), we obtain the optimal DSIC auction. The theory developed in this lecture, which is due to Myerson [Mye81], is even more general, as it can be extended to accommodate valuation distributions that are not regular, but this requires more work; see [Har17, Chapter 3].

To summarize, if the probability distributions $F_1, \ldots, F_n$ of n bidders are identical, independent, and regular, then choosing the allocation rule

$$x(v) = \text{argmax}_{(x_1, \ldots, x_n) \in X} \sum_{i=1}^{n} \varphi(v_i) x_i$$

for every v, we obtain a monotone allocation rule that can be extended to a DSIC mechanism by Myerson's lemma (Theorem 3.8). We thus see that for such setting, the Vickrey auction with appropriately chosen reserve price is optimal single-item auction format. We also noted that there are optimal DISC auctions even if we relax the conditions a bit by not insisting on the probability distributions $F_1, \ldots, F_n$ being identical. However, an optimal auction can get weird, and it does not generally resemble any auctions used in practice; see Exercise 3.9.

3.2.3   The Bulow–Klemperer Theorem

In Subsection 3.2.2, we showed how to find an optimal single-item auction format if we are given independent and regular probability distributions $F_1, \ldots, F_n$. The distributions $F_1, \ldots, F_n$ were known in advance to the seller, which is not always true in practice (in so-called *thin markets*); consider, for example, keyword auctions for rarely used search queries. Here we show how to deal with this issue.

Thus, the goal is to design an auction, whose description is independent of the underlying distributions, which is called *prior-independent auction*, and that performs almost as well as if the distributions were known in advance. The expected revenue of a Vickrey auction with no reserve can obviously only be at most as large as that of an optimal auction. Yet the following result due to Bulow and Klemperer [BK96] shows that this inequality reverses when the Vickrey auction's environment is made slightly more competitive.

Theorem 3.15 (The Bulow–Klemperer Theorem [BK96]). *Let $F = F_1 = \cdots = F_n$ be a regular probability distribution and let n be a positive integer. Then the following inequality holds*

$$\mathbb{E}_{v_1,\ldots,v_{n+1}\sim F}\left[\mathrm{Rev}(VA_{n+1})\right] \geq \mathbb{E}_{v_1,\ldots,v_n\sim F}\left[\mathrm{Rev}(OPT_{F,n})\right], \tag{3.6}$$

where $\mathrm{Rev}(VA_{n+1})$ *denotes the revenue of Vickrey auction* VA_{n+1} *with* $n+1$ *bidders (and no reserve) and* $\mathrm{Rev}(OPT_{F,n})$ *denotes the revenue of the optimal auction* $OPT_{F,n}$ *for F with n bidders.*

We recall that $OPT_{F,n}$ is the Vickrey auction with the monopoly reserve price $\varphi^{-1}(0)$, where φ is the virtual valuation function of F. Note that the auction on the left side of (3.6) does not depend on the probability distribution F while the one on the right side does. The Bulow–Klemperer theorem implies that in all possible single-item auctions with identical regular probability distributions on valuations of $n \geq 2$ bidders, the expected revenue of Vickrey auction is at least $\frac{n-1}{n}$-fraction of the expected revenue of an optimal auction for the same number of bidders; see Exercise 3.10. Informally, the underlying idea behind this result is that extra competition is more important than getting the auction format just right.

Proof of Theorem 3.15. We define an auxiliary auction $\mathcal{A}$ of $n+1$ bidders as follows:

(*a*) Simulate the optimal auction $OPT_{F,n}$ on the bidders $1, \ldots, n$,

(*b*) If the item was not awarded in Step (*a*), then give the item to bidder $n+1$ for free.

By definition, we have

$$\mathbb{E}_{v_1,\ldots,v_{n+1}\sim F}\left[\mathrm{Rev}(\mathcal{A})\right] = \mathbb{E}_{v_1,\ldots,v_n\sim F}\left[\mathrm{Rev}(OPT_{F,n})\right].$$

Also, note that $\mathcal{A}$ always allocates the item.

To finish the proof, we argue that if $F = F_1 = \cdots = F_n$ and F is regular, then the Vickrey auction maximizes expected revenue over all auctions that are guaranteed to allocate the item. It follows from Theorem 3.13 that the optimal auction, which always allocates the item, awards the item to the bidder with the highest virtual valuation (even if this is negative). The Vickrey auction awards the item to the bidder with the highest valuation. Since $F = F_1 = \cdots = F_n$ and F is regular, all bidders share the same increasing virtual valuation function φ. Thus, the bidder with the highest virtual valuation is always the bidder with the highest valuation. We conclude that the Vickrey auction VA_{n+1} has expected revenue at least that of every auction that always allocates the item, including $\mathcal{A}$. Therefore the expected revenue $\mathbb{E}_{v_1,\ldots,v_{n+1}\sim F}\left[\mathrm{Rev}(VA_{n+1})\right]$ is at least as large as the expected revenue $\mathbb{E}_{v_1,\ldots,v_{n+1}\sim F}\left[\mathrm{Rev}(\mathcal{A})\right] = \mathbb{E}_{v_1,\ldots,v_n\sim F}\left[\mathrm{Rev}(OPT_{F,n})\right]$. $\square$

3.2.4 Exercises

Exercise 3.6. *Let F be the uniform probability distribution on $[0,1]$. Consider a single-item auction with two bidders 1 and 2 that have probability distributions $F_1 = F$ and $F_2 = F$ on their valuations.*

(a) *Prove that the expected revenue obtained by the Vickrey auction (with no reserve) is $1/3$.* **[2]**

(b) *Prove that the expected revenue obtained by the Vickrey auction with reserve $1/2$ is $5/12$.* **[3]**

Exercise 3.7. *Compute the virtual valuation function of the following probability distributions and show which of these distributions are regular (meaning the virtual valuation function is strictly increasing).*

(a) *The uniform distribution $F(z) = z/a$ on $[0,a]$ with $a > 0$,* **[1]**

(b) *The exponential distribution $F(z) = 1 - e^{-\lambda z}$ with rate $\lambda > 0$ on $[0,\infty)$,* **[1]**

(c) *The distribution given by $F(z) = 1 - \frac{1}{(z+1)^c}$ on $[0,\infty)$, where $c > 0$ is some constant,* **[2]**

(d) *Consider the probability distribution F in part (c), with $c = 1$. Argue that when bidder valuations are drawn from F, it is not necessarily the case that the expected revenue of an auction equals its expected virtual social surplus. To reconcile this observation with Theorem 3.13, identify which assumption of this result is violated in your example.* **[3]**

Exercise 3.8. *Consider an arbitrary single-parameter environment with feasible set X and n bidders. For every bidder i, the valuation of i is drawn from a regular probability distribution F_i, so the virtual valuation function φ_i of i is strictly increasing. Consider the allocation rule x that maximizes the virtual social surplus for any given input v. That is,*

$$x(v) = \mathrm{argmax}_{(x_1,\ldots,x_n) \in X} \sum_{i=1}^{n} \varphi_i(v_i) x_i.$$

Prove that this allocation rule is monotone. **[2]**

Remark: You should assume that ties are broken in a deterministic and consistent way, such as lexicographically.

Exercise 3.9. *Consider a single-item auction where bidder i draws his valuation from his own regular distribution F_i, that is, the probability distributions $F_1, \ldots, F_n$ can be different but all virtual valuation functions $\varphi_1, \ldots, \varphi_n$ are strictly increasing.*

(a) *Give a formula for the winner's payment in an optimal auction, in terms of the bidders' virtual valuation functions φ_i. Verify that if $F_1 = \cdots = F_n$ are uniform probability distributions on $[0, 1]$, then you obtain Vickrey auction with reserve price $1/2$.* **[2]**

(b) *Find an example of an optimal auction in which the highest bidder does not win, even if he has a positive virtual valuation.* **[2]**

Hint: It suffices to consider two bidders with valuations from different uniform distributions.

Exercise 3.10. *Consider a single-item auction with $n \geq 2$ bidders that draw their valuations from a regular probability distribution F. Prove that the expected revenue of the Vickrey auction with no reserve is at least $\frac{n-1}{n}$-fraction of the expected revenue of the optimal auction with the same number n of bidders.* **[3]**

Hint: deduce this statement from the Bulow–Klemperer theorem. When one new bidder is added, how much can the maximum-possible expected revenue increase?

3.3 Multi-parameter mechanism design

So far we have discussed single-parameter mechanism designs, where each bidder has only a single piece of private information—his valuation v_i. In this section, we consider more general *multi-parameter mechanism design* where bidders have different private valuations for different items. Formally, we have the following setting:

(a) n strategic participants (or bidders),

(b) a finite set Ω of outcomes,

(c) each bidder i has a private valuation $v_i(\omega) \geq 0$ for every outcome $\omega \in \Omega$.

Each bidder i submits his bids $b_i(\omega)$ for each $\omega \in \Omega$ and our goal is to design a mechanism that selects an outcome $\omega \in \Omega$ so that it maximizes the *social surplus* $\sum_{i=1}^{n} v_i(\omega)$. Note that the valuations now depend on possible outcomes, so, for example, if bidders compete for a single item, each bidder can have an opinion about each other bidder winning the item as well. In this section, we assume that all bids $b_i(\omega)$ are non-negative. Also, the set Ω of possible outcomes can be very large, we only require Ω to be finite. We illustrate this definition with a few examples.

Example 3.16 (Single-item auction). *In the single-item auction, the set of outcomes $\Omega = \{\omega_1, \ldots, \omega_n, \omega_\emptyset\}$ has size $n+1$ and each outcome ω_i with $i \in \mathbb{N}$ corresponds to the winner i of the item. The last outcome $\omega_\emptyset$ corresponds to the situation when nobody gets the item. For every $i = 1, \ldots, n$, the valuations are set to be $v_i(\omega_j) = 0$ for every $j \neq i$ and $v_i(\omega_i) = v_i$ otherwise. We thus have only one unknown parameter for each bidder i.*

Example 3.17 (Two-item auction). *Assume there are two bidders 1 and 2 in an auction in which the seller sells two items t_1 and t_2 and each bidder is allowed to obtain only one item. The set of possible outcomes is $\Omega = \{(0,0), (0,1), (0,2), (1,0), (2,0), (1,2), (2,1)\}$, where an outcome (i,j) corresponds to bidder 1 receiving the item t_i (or nothing if $i = 0$) and bidder 2 receiving the item t_j (or nothing if $j = 0$). The valuations of bidder 1 are $v_1((i,j)) = 0$ if $i = 0$, $v_1((i,j)) = 10$ if $i = 1$, and $v_1((i,j)) = 5$ if $i = 2$. Similarly, the valuations of bidder 2 are $v_2((i,j)) = 0$ if $j = 0$, $v_2((i,j)) = 5$ if $j = 1$, and $v_2((i,j)) = 3$ if $j = 2$. Thus, we see that both bidders prefer item t_1 and also want to buy at least something. The optimal allocation is to sell t_1 to bidder 1 and t_2 to bidder 2, which leads to the social surplus $10 + 3 = 13$.*

We now cover a cornerstone of mechanism design theory, the following result by Vickrey [Vic61b], Clarke [Cla71], and Groves [Gro73] called the *VCG mechanism*. It states that at least one result from the single-parameter setting extends to the general multi-parameter setting, namely that we can still do DSIC social surplus maximization.

Theorem 3.18 (The Vickrey–Clarke–Groves (VCG) mechanism [Vic61b, Cla71, Gro73]). *In every multi-parameter mechanism design environment, there is a DSIC social-surplus-maximizing mechanism.*

In other words, we can still have the first two properties from the definition of an awesome auction (Definition 3.1), the *strong incentive guarantees* (every bidder has his dominant strategy by setting $b_i(\omega) = v_i(\omega)$ for every $\omega \in \Omega$) and the *strong performance guarantees* (if bidders bid truthfully, that is $b_i(\omega) = v_i(\omega)$ for all bidders i and outcomes $\omega \in \Omega$, then the mechanism maximizes the social surplus). We recall that we had to give up the third property (polynomial running time) already in single-parameter environments; see Subsection 3.1.4 about Knapsack auctions.

Proof of Theorem 3.18. We proceed in two steps. First, we assume, without justification, that bidders truthfully reveal their private information, and then figure out which outcome from Ω to pick. We want to maximize the social surplus, so our allocation rule is forced to pick an outcome that maximizes the social surplus. So, given bids $b = ((b_1(\omega))_{\omega \in \Omega}, \ldots, (b_n(\omega))_{\omega \in \Omega})$ (note that each bid is a vector indexed by Ω), we define the allocation rule by

$$x(b) = \mathrm{argmax}_{\omega \in \Omega} \sum_{i=1}^{n} b_i(\omega). \tag{3.7}$$

As the second step, we need to choose payments p so that, together with our allocation rule x, they form a DSIC multi-parameter mechanism (x, p). We choose $p = (p_1, \ldots, p_n)$ so that our assumption about bidders revealing their information truthfully is justified. This is what we have already done in the proof of Myerson's lemma (Theorem 3.8), which dealt with single-parameter environments. We recall that allocation rule monotonicity is necessary and sufficient for implementability and, for monotone rules, Myerson's lemma gives an explicit formula for the unique payment rule that meets the DSIC condition. However, Myerson's lemma does not hold beyond single-parameter environments, in particular, it is not even clear how to define monotonicity in multi-parameter environments.

The key idea turns out to be considering the "externality" caused by bidder i, that is, the loss of social surplus inflicted on the other $n - 1$ bidders by the presence of bidder i. For example, in single-item auctions, the winning bidder inflicts a social surplus loss of the second-highest bid to the others, which is the payment rule in the Vickrey auction. Intuitively, we define the payments to force each bidder to care about the other $n - 1$ bidders and not just to care about

himself. Formally, we choose the payment rule as

$$p_i(b) = \max_{\omega \in \Omega} \left\{ \sum_{\substack{j=1 \\ j \neq i}}^{n} b_j(\omega) \right\} - \sum_{\substack{j=1 \\ j \neq i}}^{n} b_j(\omega^*) \tag{3.8}$$

for every bidder i, where $\omega^* = x(b)$ is the outcome chosen by our allocation rule x for given bids b. The first term in the definition of $p_i(b)$ is the surplus of the remaining $n-1$ bidders if we omit bidder i. The second term is the social surplus if we consider bidder i. By definition, the mechanism (x, p) is maximizing social surplus, assuming truthful bids.

It remains to prove that the mechanism (x, p) is DSIC, as then the assumption that bidders bid truthfully is justified by the first property of DSIC mechanisms (the strong incentive guarantees). That is, we need to show that each bidder i maximizes his *utility* $v_i(x(b)) - p_i(b)$ by setting $b_i(\omega) = v_i(\omega)$ for every $\omega \in \Omega$. One can show that $p_i(b)$ is always non-negative and that $p_i(b) \leq b_i(\omega^*)$; see Exercise 3.11. Hence truthtelling agents are guaranteed non-negative utility.

We fix bidder i and the bids of other bidders b_{-i}. If $x(b) = \omega^*$, then the utility of bidder i equals

$$v_i(\omega^*) - p_i(b) = \left(v_i(\omega^*) + \sum_{\substack{j=1 \\ j \neq i}}^{n} b_j(\omega^*) \right) - \left(\max_{\omega \in \Omega} \left\{ \sum_{\substack{j=1 \\ j \neq i}}^{n} b_j(\omega) \right\} \right),$$

where we just expanded the term $p_i(b)$ using (3.8). The second term in the expansion is independent of b_i, thus bidder i needs to maximize the first term of the expansion in order to maximize his utility. However, bidder i cannot influence the outcome ω^* directly, the mechanism (x, p) gets to choose ω^*. The VCG mechanism chooses ω^* according to (3.7), so that sums of bids are maximized. Best case for bidder i is when the mechanism picks ω^* that maximizes the first term of the expansion, that is, bidder i wants to select

$$\operatorname{argmax}_{\omega \in \Omega} \left\{ v_i(\omega) + \sum_{\substack{j=1 \\ j \neq i}}^{n} b_j(\omega) \right\}. \tag{3.9}$$

If i bids truthfully, then (3.9) becomes the expression in (3.7). Thus, bidding truthfully results in the mechanism (x, p) choosing an outcome that maximizes agent i's utility, no other bid could be better. $\square$

Note that we only used the fact that the second term in the expansion of bidder i's utility $v_i(\omega^*) - p_i(b)$ is independent of b_i. The reason why we choose the payment formula as in (3.8) is that, besides natural interpretations, this choice guarantees non-negative payments and also non-negative utilities of truthtelling bidders.

We now present an alternative way how to think about the payments from (3.8). We can rewrite the definition of $p_i(b)$ as

$$p_i(b) = b_i(\omega^*) - \left(\sum_{j=1}^{n} b_j(\omega^*) - \max_{\omega \in \Omega} \left\{ \sum_{\substack{j=1 \\ j \neq i}}^{n} b_j(\omega) \right\} \right).$$

Thus, the payment $p_i(b)$ of bidder i is his bid minus the increase in social surplus attributable to the presence of i. For example, in the Vickrey auction, the highest bidder pays his bid b_1 minus $b_1 - b_2$, where b_2 is the bid of the second-highest bidder. Note that $b_1 - b_2$ is the increase in social surplus that the highest bidder brings.

To summarize, Theorem 3.18 says that DSIC maximization of social surplus is possible even in the very general setting of multi-parameter mechanisms. However, in practice, implementing this can be very hard. Nevertheless, the VCG mechanism serves as a useful benchmark for other, more practical approaches.

3.3.1 The Revelation Principle

So far we have considered only DSIC mechanisms, as they are easy to play and predict. A natural question is whether we lose anything by restricting ourselves to these mechanisms. We split the DSIC property into the following two parts:

(*a*) Each bidder has a dominant strategy, no matter what his private valuation is.

(*b*) This dominant strategy is of a special type, so-called *direct revelation*[13], where the participant truthfully reports all of its private information to the mechanism.

To see an example of an auction that satisfies the first property, but not the second one, consider a single-item auction, where the seller on given bids $(b_1, \ldots, b_n)$ runs a Vickrey's auction on bids $(2b_1, \ldots, 2b_n)$. Then each bidder has a dominant strategy and thus the first property is satisfied. However, this dominant strategy is not direct revelation, as the dominant strategy of each bidder is to bid half of his value, which is not truthful bidding.

Informally, the *Revelation Principle*[14] says that only the first condition matters while the second one then comes for free. Clearly, the first condition is not always satisfied, for example, there are no dominant strategies in first-price auctions. Without this condition, it becomes harder for the seller to predict the behavior of the bidders, so we need some stronger assumptions about the mechanism. We remark that there are such mechanisms that perform better than DSIC mechanisms, but we will not cover those here. So, in general, it sometimes makes sense to relax the first condition. The Revelation Principle says that there is no need to relax the second condition.

Theorem 3.19 (The Revelation Principle). *For every multi-parameter mechanism M in which every bidder has a dominant strategy, no matter what his private valuation is, there is an equivalent mechanism M' in which each bidder has a dominant strategy that is a direct revelation.*

Proof. The proof proceeds by a simulation argument. For each bidder i and his valuations $(v_i(\omega)_{\omega \in \Omega}$, let $s_i(v_i(\omega)_{\omega \in \Omega})$ be the dominant strategy of i in the mechanism M.

We now construct the mechanism M' that additionally satisfies the second condition. The mechanism M' accepts sealed bids $b_1(\omega)_{\omega \in \Omega}, \ldots, b_n(\omega)_{\omega \in \Omega}$ from the bidders. Then M' submits the bids $s_1(b_1(\omega)_{\omega \in \Omega}), \ldots, s_n(b_n(\omega)_{\omega \in \Omega})$ to M and M' outputs the same outcome as M.

The direct revelation is a dominant strategy in M', as if a bidder i has valuations $v_i(\omega)_{\omega \in \Omega}$, then submitting any other bid than $v_i(\omega)_{\omega \in \Omega}$ can only result in playing a different strategy than $s_i(v_i(\omega)_{\omega \in \Omega})$ in M'. This, however, can only decrease the utility of i. $\qquad\square$

3.3.2 Exercises

Exercise 3.11. *Prove that the payment rule from the proof of the VCG mechanism is always non-negative and bounded from above by $b_i(\omega^*)$. That is, show that $0 \leq p_i(b) \leq b_i(\omega^*)$ for every vector b of bids, where $p_i(b)$ is defined in* (3.8) *and ω^* is selected by the allocation rule from* (3.7). **[1]**

Exercise 3.12. *Consider a three-item auction with two bidders 1 and 2. The three items A, B, and C are being auctioned simultaneously, and each bidder can bid on any possible subset of the items. The valuations of the bidders for each subset of the items are shown in Table 3.1.*

(a*) What are the outcomes of this VCG auction? In other words, which of the two bidders will get which item(s) and what payments will they each pay?* **[3]**

(b*) Do you expect the bidders to tell you the truth about their valuations?* **[2]**

---

[13] Přímé odhalení.
[14] Princip odhalení.

bidder i	$v_i(\emptyset)$	$v_i(A)$	$v_i(B)$	$v_i(C)$	$v_i(AB)$	$v_i(AC)$	$v_i(BC)$	$v_i(ABC)$
$i=1$	0	24	4	9	29	38	20	50
$i=2$	0	15	18	11	30	34	32	47

Table 3.1: Valuations of the bidders from Exercise 3.12.

Bibliography

[AF92]   David Avis and Komei Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete Comput. Geom.*, 8(3):295–313, 1992.

[AH02]   Robert J. Aumann and Sergiu Hart, editors. *Handbook of game theory with economic applications. Vol. 3*, volume 11 of *Handbooks in Economics.* Elsevier/North-Holland, Amsterdam, 2002.

[Aum74]   Robert J. Aumann. Subjectivity and correlation in randomized strategies. *J. Math. Econom.*, 1(1):67–96, 1974.

[BK96]   Jeremy Bulow and Paul Klemperer. Auctions versus negotiations. *The American Economic Review*, 86(1):180–194, 1996.

[Bro11]   Luitzen E. J. Brouwer. Über Abbildung von Mannigfaltigkeiten. *Math. Ann.*, 71(1):97–115, 1911.

[CDhT06]   X. Chen, X. Deng, and S. h. Teng. Computing nash equilibria: Approximation and smoothed complexity. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 603–612, 2006.

[CDRP08]   Bruno Codenotti, Stefano De Rossi, and Marino Pagan. An experimental analysis of lemke-howson algorithm. 2008. arxiv.org/abs/0811.3247.

[CDT09]   Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player Nash equilibria. *J. ACM*, 56(3):Art. 14, 57, 2009.

[Cla71]   Edward Clarke. Multipart pricing of public goods. *Public Choice*, 11(1):17–33, 1971.

[Das13]   Constantinos Daskalakis. On the complexity of approximating a Nash equilibrium. *ACM Trans. Algorithms*, 9(3):Art. 23, 35, 2013.

[DGP09]   Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM J. Comput.*, 39(1):195–259, 2009.

[EY10]   Kousha Etessami and Mihalis Yannakakis. On the complexity of Nash equilibria and other fixed points. *SIAM J. Comput.*, 39(6):2531–2597, 2010.

[Gro73]   Theodore Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[GZ89]   Itzhak Gilboa and Eitan Zemel. Nash and correlated equilibria: some complexity considerations. *Games Econom. Behav.*, 1(1):80–93, 1989.

[Har17]   Jason D. Hartline. *Mechanism design and approximation.* 2017. Book draft.

[Hoe63]   Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc.*, 58:13–30, 1963.

[HPV89]   Michael D. Hirsch, Christos H. Papadimitriou, and Stephen A. Vavasis. Exponential lower bounds for finding Brouwer fixed points. *J. Complexity*, 5(4):379–416, 1989.

[Kak41]   Shizuo Kakutani. A generalization of Brouwer's fixed point theorem. *Duke Math. J.*, 8:457–459, 1941.

[LBS08]   Kevin Leyton-Brown and Yoav Shoham. *Essentials of game theory*, volume 3 of *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool Publishers, Williston, VT, 2008. A concise, multidisciplinary introduction.

[LH64]   Carlton E. Lemke and Joseph T. Howson, Jr. Equilibrium points of bimatrix games. *J. Soc. Indust. Appl. Math.*, 12:413–423, 1964.

[LMM03]   Richard J. Lipton, Evangelos Markakis, and Aranyak Mehta. Playing large games using simple strategies. In *Proceedings of the 4th ACM Conference on Electronic Commerce*, EC '03, pages 36–41, New York, NY, USA, 2003. ACM.

[MG06]   Jiří Matoušek and Bernd Gärtner. *Understanding and Using Linear Programming*. Springer-Verlag New York, Inc., 2006.

[MP91]   Nimrod Megiddo and Christos H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theoret. Comput. Sci.*, 81(2, Algorithms Automat. Complexity Games):317–324, 1991.

[Mye81]   Roger B. Myerson. Optimal auction design. *Math. Oper. Res.*, 6(1):58–73, 1981.

[Nas50]   John F. Nash, Jr. Equilibrium points in n-person games. *Proc. Nat. Acad. Sci. U. S. A.*, 36:48–49, 1950.

[Nas51]   John F. Nash, Jr. Non-cooperative games. *Ann. of Math. (2)*, 54:286–295, 1951.

[NRTV07] Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani, editors. *Algorithmic game theory*. Cambridge University Press, Cambridge, 2007.

[Pap94]   Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. System Sci.*, 48(3):498–532, 1994. 31st Annual Symposium on Foundations of Computer Science (FOCS) (St. Louis, MO, 1990).

[Pap07]   Christos H. Papadimitriou. The complexity of finding nash equilibria. In Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani, editors, *Algorithmic Game Theory*, pages 29–52. Cambridge University Press, 2007.

[Rou10]   Tim Roughgarden. Computing equilibria: a computational complexity perspective. *Econom. Theory*, 42(1):193–236, 2010.

[Rou16]   Tim Roughgarden. *Twenty lectures on algorithmic game theory*. Cambridge University Press, Cambridge, 2016.

[Ste02]   Bernhard Von Stengel. Computing equilibria for two-person games. In R. J. Aumann and S. Hart, editors, *Handbook of Game Theory with Economic Applications*, volume 3, chapter 45, pages 1723–1759. Elsevier, 2002.

[Vic61a] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *J. Finance*, 16(1):8–37, 1961.

[Vic61b] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *J. Finance*, 16(1):8–37, 1961.

[vN28]   John von Neumann. Zur Theorie der Gesellschaftsspiele. *Math. Ann.*, 100(1):295–320, 1928.

[vNMKR44] John von Neumann, Oskar Morgenstern, Harold W. Kuhn, and Ariel Rubinstein. *Theory of Games and Economic Behavior (60th Anniversary Commemorative Edition)*. Princeton University Press, 1944.