AI, Games, and Markets

Christian Kroer[1]
Department of Industrial Engineering and Operations Research
Columbia University

December 13, 2024

[1]Email: `ck2945@columbia.edu`.

Preface

Why this book?

There are several great books for game theory and market design from a computational perspective. Notably, I "grew up" with the books Nisan et al. [103], Shoham and Leyton-Brown [116] and Easley et al. [51], which are classic texts in this area. The idea for the present book is that it will be complementary to these books: by heavily leveraging online learning, a topic that has developed tremendously since the publication of these books, we can provide an alternative perspective on computational game theory and market design. Moreover, many of the major application results that we will cover occurred after the publication of these books, e.g. the development of the techniques used for superhuman poker AIs occurred in 2007-2018, the development of a theory of equilibrium for internet ad auctions with budgets in the 2010s, and the application of competitive equilibrium from equal incomes for course seat allocation around 2008-2016. More generally, there has been a proliferation of ways in which online learning can be used to solve and analyze game-theoretic and market problems.

Mechanism Design in This Book

Mechanism design is a mathematically beautiful subject, especially the derivation of Myerson's optimal mechanism. I was tempted to add this to the present book, but I decided to keep formal mechanism design somewhat limited in this book, since it is a big topic, and not really necessary for the types of results that we cover. Moreover, there are several great books covering this topic already. Aspiring researchers would be well served by picking up one of the below books as a complement to this one. I recommend the following:

- *An Introduction to the Theory of Mechanism Design* by Tilman Börgers. This book is a really nice read; it has very nice and concise derivations of the main results.
- *Auction Theory* by Vijay Krishna. As the name suggests, this book is focused on auctions rather than general mechanism design. It also has a nice derivation of Myerson's result.

Target Audience

This book is targeted primarily at graduate students and researchers in operations research, computer science, and adjacent engineering fields. They may also provide an interesting alternative perspective for economics researchers that wish to see a computational perspective on game theory and market design. Senior undergraduate students with a background in optimization and probability should also be able to follow large parts of the book. My senior undergraduate and master's course at Columbia University uses this book, but omits topics like Blackwell approachability and the more advanced parts of the regret minimization chapter. Finally, the book can also be read by practitioners in the tech industry that wish to get an introduction to theoretical models for understanding e.g. internet advertising markets.

The background requirements are as follows:

- Knowledge of linear algebra, probability, and calculus.
- A basic background in convex, linear, and integer optimization.

- I will sometimes refer to basic concepts from computational complexity theory, but a background in this is not required.

The notes do not assume any background in game theory or mechanism design. For convex optimization, it is possible to read up on the basics as you go along. For example, the first few chapters of Boyd and Vandenberghe [17] are a good reference. For linear optimization, I recommend Bertsimas and Tsitsiklis [12].

Organization of the Book

Chapter 1 gives some motivating examples of applications that can be approached with techniques from this book. Chapters 2 and 3 give a fast introduction to the most basic concepts from game theory, auctions, and mechanism design. A reader that is already familiar with these topics can skip or skim these chapters. Chapter 4 gives an introduction to online learning and regret minimization, which is a key tool in the book. That chapter also gives a constructive proof of von Neumann's minimax theorem via online learning. Chapter 5 introduces Blackwell approachability, and derived the *regret matching* algorithm. This algorithm is crucial for large-scale EFG solving. Chapter 5 can be skipped if the reader is not interested in large-scale game solving. Chapters 6 and 7 show how to solve two-player zero-sum games using regret minimization. Chapters 8 and 9 introduce extensive-form games, and show algorithmic approaches for solving large-scale games. Chapters 10 and 11 introduce the problem of fair and efficient allocation of goods, first in the divisible setting, and then the indivisible setting. Chapters 12 to 15 introduce various real-world complications arising from the application of auction theory to the problem of internet advertising, including position auctions, how to handle budget constrained advertisers, and demographic fairness. The remaining chapters of the book are less cohesive, and generally tend to cover more advanced topics, including Stackelberg games, fair combinatorial allocation, and electricity markets.

Acknowledgments

This book owes a large debt to several other professors that have taught courses on Economics and Computation. In particular, Tim Roughgarden's lecture notes [113] and video lectures, John Dickerson's course at UMD[1], and Ariel Procaccia's course at CMU[2] provided inspiration for course topics as well as presentation ideas.

The book owes a debt to several former and current PhD students that have helped me teach the course at Columbia that this book is based upon. In particular, I would like to thank Rachitesh Kumar, Luofeng Liao, Aditya Shankhar Garg, and Steven Sofos DiSilvio for their help with teaching the course, which has indirectly helped shape this book as well.

In a similar vein, I would like to thank the students that have taken my course at Columbia, and provided feedback on first the lecture notes and later the book. This has tremendously improved the book.

I would also like to thank the following people for extensive feedback on the lecture notes: Ryan D'Orazio for both finding mistakes and providing helpful suggestions on presenting Blackwell approachability. Gabriele Farina for discussions around several regret minimization issues, and for helping me develop much of my thinking on regret minimization in general. Shuvomoy Das Gupta for carefully reading many chapters and finding various typos and mistakes.

I also thank the following people who pointed out mistakes and typos in the notes: Mustafa Mert Çelikok, Ajay Sakarwal, Eugene Vinitsky, David Yang.

[1]https://www.cs.umd.edu/class/spring2018/cmsc828m/
[2]http://www.cs.cmu.edu/ arielpro/15896s16/index.html

Contents

Chapter 1

Introduction and Examples

This book provides an introduction to the topics of game theory and market design, with a focus on how AI and optimization methods can be used both to understand these problems, as well as enable them in practical settings. The book covers several application areas for these ideas, where each area will have real-life applications that have been deployed. A common theme underlying the areas covered by the book is that for each area, one or more of the real applications are enabled by AI and optimization. Firstly, we will repeatedly see that economic solution concepts often have some underlying convex or mixed-integer formulation of the problem, that allows us to analyze the problem via optimization theory, as well as enabling algorithms via optimization techniques. Secondly, the book uses the concept of *online learning* as a unifying theme for enabling algorithms and analysis across many of the economic topics that we cover. Thirdly, some applications require scaling at a level where standard optimization and online learning methods are not enough. In those settings, AI methods such as abstraction or machine learning are often used. For example, we may have a game that is way too large to even fit in memory. In that case, we can generate some coarse-grained representation of the problem using abstraction or machine learning. This coarse-grained representation is then typically what we solve with optimization methods.

The following subsections give examples of the types of ideas and applications the book will cover.

1.1 Game Theory

The first pillar of the course will be *game theory*. In classical optimization, we have some form of objective function that we try to minimize or maximize, say $\max_{x \in \mathcal{X}} f(x)$, where $\mathcal{X}$ is a convex set of possible choices, and f is some concave function. For example, perhaps we are thinking of $\mathcal{X}$ as a set of prices that a retailer can set for a given item, and $f(x)$ tells us the revenue that the retailer gets when setting the price x.

In game theory, on the other hand, we study settings where multiple individuals make choices, and the outcome depends on the choices of all the individuals. Suppose that we have two retailers, each choosing prices x_1 and x_2 respectively. Now, suppose that f_1 is a function that tells us the revenue received by retailer 1 in this setup. Since consumers will potentially compare the prices x_1 and x_2, we should expect f_1 to depend on both x_1 and x_2, so we let $f_1(x_1, x_2)$ be the revenue for retailer 1 generated under prices x_1 and x_2. Now we can again try to think of the optimization problem that retailer 1 wishes to solve; first let us assume that x_2 was already chosen and retailer 1 knows its value, in that case they want to solve $\max_{x_1 \in \mathcal{X}} f_1(x_1, x_2)$. However, we could similarly argue that retailer 2 should choose their price x_2 based on the price x_1 chosen by retailer 1. Now we have a problem, because we cannot talk about optimally choosing either of the two prices in isolation, and instead we need a way to reason about how they might be chosen in a way that depends on each other. Game theory provides a formal way to reason about this type of situation. For example, the famous *Nash equilibrium*, which we will study below, specifies that we should find a pair x_1, x_2 such that they are mutually optimal with respect to each other. Another solution concept we will see is the *Stackelberg equilibrium*, where one retailer is assumed to go first, while anticipating the optimization problem being solved by the second retailer. From now on we will refer to each individual optimizer in a problem either as a player or an *agent*.

1.1.1 Nash Equilibrium

One of the most important ideas in game theory is the famous Nash equilibrium. A Nash equilibrium is a specification of an action for each player (or a probability distribution over actions) such that it is a steady state of a game, in the sense that no player wishes to change their action given what everybody else is doing. This is best illustrated with an example. Below are the payoffs of the game of rock-paper-scissors (RPS), specified as a *bimatrix* of payoffs.

	Rock	Paper	Scissors
Rock	0,0	-1,1	1,-1
Paper	1,-1	0,0	-1,1
Scissors	-1,1	1,-1	0,0

In this representation, Player 1 chooses a row to play, and Player 2 chooses a column to play. Player 1 tries to maximize the first value at the resulting entry in the bimatrix, while Player 2 tries to maximize the second value.

Here is an example of something that is *not* a Nash equilibrium: Player 1 always plays rock, and Player 2 always plays scissors. In this case, Player 2 is not playing optimally given the strategy of Player 1, since they could improve their payoff from -1 to 1 by switching to deterministically playing paper. In fact, this argument works for any pair of deterministic strategies, and so we see that there is no Nash equilibrium consisting of deterministic strategies.

Instead, RPS is an example of a game where we need randomization in order to arrive at a Nash equilibrium. The idea is that each player gets to choose a probability distribution over their actions instead (e.g. a distribution over rows for Player 1). Now, the value that a given player receives under a pair of mixed strategies is their expected payoff given the randomized strategies. In RPS, it's easy to see that the unique Nash equilibrium is for each player to play each action with probability $\frac{1}{3}$. Given this distribution, there is no other action that either player can switch to and improve their utility. This is what we call a (mixed-strategy) Nash equilibrium.

The famous result of John Nash from 1951 is that *every* game has a Nash equilibrium:

Theorem 1. *Every bimatrix game has a Nash equilibrium.*

In fact, Nash's result is broader: it covers a quite broad class of n-player games, as we shall see in the next lecture.

The attentive reader may have noticed that the RPS game has a further property: whenever one player wins, the other loses. This means that each player can equivalently reason about minimizing the utility of the other player, rather than maximizing their own utility. More generally, a bimatrix game is a *zero-sum game* if it can be represented in the following form:

$$\min_{x \in \Delta^n} \max_{y \in \Delta^m} \; x^\top A y$$

where $\Delta^n = \{x \in \mathbb{R}^n : \sum_{i=1}^n x_i = 1, x \geq 0\}$ is the probability simplex over n actions, Δ^m the probability simplex of m actions, and A contains the payoff entries to the y-player from the bimatrix representation. Problems of this form are also known as *bilinear saddle-point problems*. The key here is that we can now represent the outcome of the game as a single matrix, where the x-player wishes to minimize the bilinear term $x^\top Ay$ and the y-player wishes to maximize it. Zero-sum matrix games are very special: they can be solved in polynomial time with a linear program whose size is linear in the matrix size.

Rock-paper-scissors is of course a rather trivial example of a game. A more exciting application of zero-sum games is to use it to compute an optimal strategy for two-player poker (AKA heads-up poker). In fact, as we will discuss later, this was the foundation for many recent" "superhuman AI for poker" results [16, 95, 20, 22]. In order to model poker games we will need a more expressive game class called *extensive-form games* (EFGs). These games are played on trees, where players may sometimes have groups of nodes, called *information sets*, that they cannot distinguish among. An example is shown in Figure 8.3.

EFGs can also be represented as a bilinear saddle-point problem:

$$\min_{x \in X} \max_{y \in Y} \; x^\top A y,$$

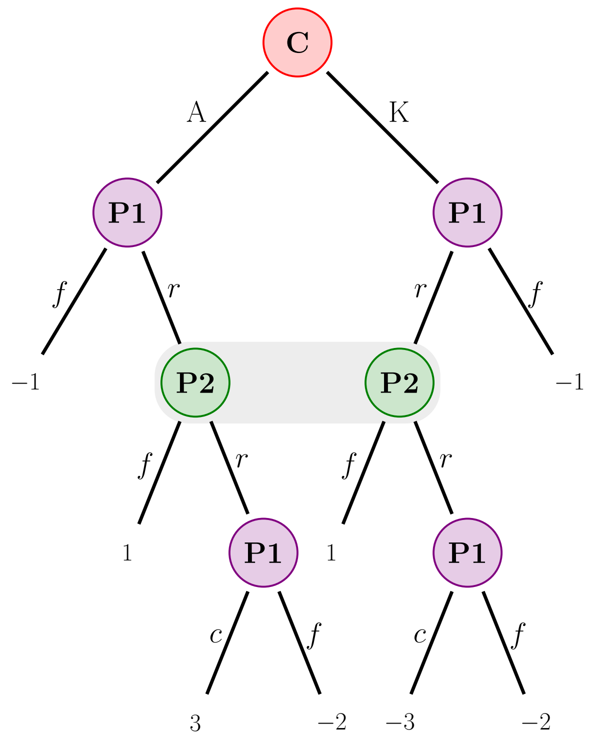


Figure 1.1: A poker game where P1 is dealt Ace or King. "r," "f," and "c" stands for raise, fold, and check respectively. Leaf values denote P1 payoffs. The shaded area denotes an information set: P2 does not know which of these nodes they are at, and must thus use the same strategy in both.

where X, Y are no longer probability simplexes, but more general convex polytopes that encode the sequential decision spaces of each player. This is called the *sequence-form* representation [134], and we will cover that later. Like matrix games, zero-sum EFGs can be solved in polynomial time with linear programming, with an LP whose size is linear in the game tree.

It turns out that in many practical scenarios, the LP for solving a zero-sum game ends up being far too large to solve. This is especially true for EFGs, where the game tree can quickly become extremely large if the game has almost any amount of depth. Instead, iterative methods are used in practice. What is meant by iterative methods here is the class of algorithms that build a sequence of strategies $x_0, x_1, \ldots, x_T, y_0, y_1, \ldots, y_T$ using only some form of oracle access to Ay and $A^\top x$ (this is different from writing down A explicitly!). Typically in such iterative methods, the average of the sequence of strategies $\bar{x}_T = \frac{1}{T}\sum_{t=1}^T x_t, \bar{y}_T = \frac{1}{T}\sum_{t=1}^T y_t$ converge to a Nash equilibrium. The reason these methods are preferred is two-fold. First, by never writing down A explicitly we save a lot of memory (now we just need enough memory to store the much smaller x, y strategy vectors). Secondly, they avoid the expensive matrix inversions involved in the simplex algorithm and interior-point methods.

The algorithmic techniques we will learn for Nash equilibrium computation are largely centered around iterative methods. First, we will do a quick introduction to online learning and online convex optimization. We will learn about two classes of algorithms: ones that converge to an equilibrium at a rate $O(1/\sqrt{T})$. These roughly correspond to saddle-point variants of gradient-descent-like methods. Then we will learn about methods that converge to the solution at a rate of $O(1/T)$. These roughly correspond to saddle-point variants of accelerated gradient methods. Then we will also look at the practical performance of these algorithms. Here we will see that the following quote is very much true:

In theory, theory and practice are the same. In practice, they are not.

In particular, the preferred method in practice is the CFR$^+$ algorithm [124] and later variations [21], all of which have a theoretical convergence rate of $O(1/\sqrt{T})$. In contrast, there are methods that converge at a rate of $O(1/T)$ [76, 88, 86] in theory, but these methods are actually slower than CFR$^+$ for most real games!

Being able to compute an approximate Nash equilibrium with iterative methods is only one part of how superhuman AIs were created for poker. In addition, abstraction and deep learning methods were used to create a small enough game that can be solved with iterative methods. We will also cover how these methods are used.

Killer applications of zero-sum games include poker (as we saw), other recreational two-player games, and generative-adversarial networks (GANs). Other applications that are, as of yet, less proven to be effective in practice are robust sequential decision making (the adversary represents uncertainty), security scenarios where we assume the world is adversarial, and defense applications.

1.1.2 Stackelberg Equilibrium

A second game-theoretic solution concept that has had extensive application in practice is what's called a *Stackelberg equilibrium*. We will primarily study Stackelberg equilibrium in the context of what is called *security games* [122].

Imagine the following scenario: we are designing the patrol schedule for national park rangers that try to prevent poaching of endangered wildlife in the park (such as rhinos, which are poached for their horns). There are 20 different watering holes that the rhinos frequent. We have 5 teams of guards that can patrol watering holes. How can we effectively combat poaching? If we come up with a fixed patrol schedule then the poachers can observe us for a few days and learn our schedule exactly. Afterwards they can strike at a waterhole that is guaranteed to be empty at some particular time. Thus we need to design a schedule that is unpredictable, but which also accounts for the fact that some watering holes are more frequented by rhinos (and are thus higher value), travel constraints, etc.

In the security games literature, the most popular solution concept for this kind of setting is the Stackelberg equilibrium. In a Stackelberg equilibrium, we assume that we, as the leader (e.g. the park rangers), get to commit to our (possibly randomized) strategy first. Then, the follower observes our strategy and best responds. This turns out to yield the same solution concept as Nash equilibrium in zero-sum games, but in general games it leads to a different solution concept.

However, if we want to help the park rangers design their schedules then we will need to be able to compute Stackelberg equilibria of the resulting game model. Again, we will see that optimization is one of the fundamental pillars of the field of security games research. A unique feature of security games is that the strategy space of the leader is typically some combinatorial polytope (e.g. a restriction on the *transportation polytope*), and the problem of computing a Stackelberg equilibrium is intimately related to optimization over the underlying polytope of the defender (see Xu [137] for some nice consequences of this observation). Because of this combinatorial nature, security games often end up being much harder to solve than zero-sum Nash equilibrium. Therefore, the focus of this section will be on combinatorial approaches to this problem, such as mixed-integer programming and decomposition. Another crucial aspect of security games is having good models of the attacker. Thus, if time permits, we will also spend some time learning how one can model adversaries using machine learning.

Killer applications of Stackelberg games are mainly in the realm of security. They have been applied in infrastructure security (airports, coast guard, air marshals) [122], to protect wildlife [58], and to combat fare evasion. A nascent literature is also emerging in cybersecurity. Outside of the world of security, Stackelberg games are also used to model things like first-mover advantage in the business world.

1.2 Market Design

The second pillar of the book will be *market design*. In market design we are typically concerned with how to design the rules of the game, and how to do that in order to achieve "good" outcomes. Thus, game theory is a key tool in market design, since it will be our way of understanding what outcomes we may expect to arise, given a set of market rules.

Market design is a huge area, and so it has many killer applications. The ones we will see in this course include Internet ad auctions and how to fairly assign course seats to students. However there are many others such as how to price and assign rideshares at Lyft/Uber, how to assign NYC kids to schools, how to enable nationwide kidney exchanges, how to allocate spectrum, etc.

For example, imagine that we are designing a mechanism for managing course enrollment. How should we decide which students get to take which courses? What do we do with the fact that our machine learning course has 100 seats and 500 people that want to take it? Overall, we would like the system to somehow be *efficient*, but what does that mean? We would also like the system to be *fair*, but it's not entirely clear what that means either.

At a loss for ideas, we come up with the following solution: we will just have a sign-up system where students can sign up until a course fills up. After that we put other students on a waitlist that we clear on a first-in first-out basis as seats become available. Is this a good system? Well, let's look at a simple example: we will have 2 students and 2 courses, each course having 1 seat. Students are allowed to take at most one course. Let's say that each student values the courses as follows:

	Course A	Course B
Student 1	5	5
Student 2	2	8

Student 1 arrives first and signs up for course B. Then Student 2 arrives and signs up for A. The total *welfare* of this assignment is $5 + 2 = 7$. This does not seem to be an efficient use of resources: we can improve our solution by swapping the courses, since Student 1 gets the same utility as before, and Student 2 improves their utility. This is what's called a *Pareto-improving* allocation because each student is at least as well off as before, and at least one student is strictly better off. One desiderata for efficiency is that no such improvement should be possible; an allocation with this property is called *Pareto efficient*.

Let's look at another example. Now we have 2 students and 4 courses, where each student takes 2 courses. Again courses have only 1 seat.

	Course A	Course B	Course C	Course D
Student 1	10	10	1	1
Student 2	10	10	1	1

Now say that Student 1 shows up first, and signs up for A and B. Then Student 2 shows up and signs up for C and D. Call this assignment A_1. Here we get that A_1 is Pareto efficient, but it does not seem fair. A fairer solution seems to be that each students get a course with value 10 and a course with value 1, let A_2 be such an assignment. One way to look at this improvement is through the notion of *envy*: each student should like their own course schedule at least as well as that of any other student. Under A_1 Student 2 envies Student 1 by a value of 18, whereas under A_2 no student envies the other. An allocation where no student envies another student is called *envy-free*. Fairness turns out to be a complicated idea, and we will see later that there are several appealing notions that we may wish to strive for.

Instead of first-come-first-serve, we can use ideas from market design to get a better mechanism. The solution that we will learn about is based on a fake-money market: we give every student some fixed budget of fake currency (aka funny money). Then, we treat the assignment problem as a market problem under the assigned budgets, and ask for what is called a *market equilibrium*. Briefly, a market equilibrium is a set of prices, one for each item, and an allocation of items to buyers. The allocation must be such that every item is fully allocated (or has a price of zero), and every buyer is getting an assignment that maximizes their utility over all the possible assignments they could afford given the prices and their budget. Given such a market equilibrium, we then take the allocation from the equilibrium, throw away the prices (the money was fake anyway!), and use that to perform our course allocation. This turns out to have a number of attractive fairness and efficiency properties. Course-selection mechanisms based on this idea are deployed at several business schools such as Columbia Business School, the Wharton School at U. of Pennsylvania, University of Toronto's Rotman School of Management, and Darthmouth's Tuck School of Business [26, 27].

Of course, if we want to implement this protocol we need to be able to compute a market equilibrium. This turns out to be a rich research area: in the case of what is called a *Fisher market*, where each agent i has a linear utility function $v_i \in \mathbb{R}^m_+$ over the m items in the market there is a neat convex program that results in a market equilibrium [55]:

$$\begin{aligned} \max_{x \geq 0} \quad & \sum_i B_i \log(v_i \cdot x_i) \\ \text{s.t.} \quad & \sum_i x_{ij} \leq 1, \forall j \end{aligned}$$

Here x_{ij} is how much buyer i is allocated of item j. Notice that we are simply maximizing the budget-weighted logarithmic utilities, with no prices! It turns out that the prices are the dual variables on the supply constraints. We will see some nice applications of convex duality and Fenchel conjugates in deriving this relationship. We will also see that this class of markets have a relationship to the types of auction systems that are used at Google and Facebook [45, 46].

In the case of markets such as those for course seats, the problem is computationally harder and requires combinatorial optimization. Current methods use a mixture of MIP and local search [27].

Chapter 2

Nash Equilibrium Intro

In this lecture we begin our study of Nash equilibrium by giving the basic definitions, Nash's existence result, and briefly touch on computability issues. Then we will make a few observations specifically about zero-sum games, which have much more structure to exploit.

2.1 General-Sum Games

A *normal-form game* consists of:

- A set of players $N = \{1, \dots, n\}$
- A set of actions $A = A_1 \times A_2 \times \cdots \times A_n$
- A utility function $u_i : A \to \mathbb{R}$

A vector $a \in A$ is called a *strategy profile* and it denotes an action choice for every player. We will use the shorthand a_{-i} to denote the subset of a strategy vector a that does not include player i's action. As an aside, game theory often uses both the term "strategy" and "action" to refer to the course of action taken by a player. In the case of a normal-form game, these are the same thing, and we will use them interchangeably. However, once we study sequential games (i.e. extensive-form games) later, actions will be choices performed at individual decision points, whereas a strategy for a player will specify what they intend to do at *every* decision point.

As a first solution concept we will consider *dominant-strategy equilibrium* (DSE). In DSE, we seek a strategy profile $a \in A$ such that each action a_i is a best response *no matter what a_{-i} is.* This is a very strong property, and DSE may not exist in many games. A classic example of DSE is the *prisoner's dilemma*: two prisoners are on trial for a crime. If neither confesses (stay silent) to the crime then they will each get 1 year in prison. If one person confesses and the other does not, then the confessor gets no time, but their co-conspirator gets 9 years. If both confess then they both get 6 years.

	Silent	Confess
Silent	-1,-1	-9,0
Confess	0,-9	-6,-6

In this game, confessing is a DSE: it yields greater utility than staying silent no matter what the other player does. A DSE rarely exists in practice when given a game, but it can be useful in the context of mechanism design, where we get to design the rules of the game, potentially in order to induce a DSE. It is the idea underlying e.g. the second-price auction which we will cover later.

Consider some strategy profile $a \in A$. We say that a is a *pure-strategy Nash equilibrium* if for each player i and each alternative action $a_i' \in A_i$:

$$u_i(a) \geq u_i(a_{-i}, a_i'),$$

where, again, a_{-i} denotes all the actions in a except that of i. A DSE is always a pure-strategy Nash equilibrium, but not vice versa. Consider the *Professor's dilemma*,[1] where the professor chooses a row

[1]Example borrowed from Ariel Procaccia's slides

strategy and the students choose a column strategy:

		Students	
		Listen	Sleep
Prof.	Prepare	$10^6, 10^6$	-10,0
	Slack off	0,-10	0,0

In this game there is no DSE, but there's clearly two pure-strategy Nash equilibria: the professor prepares and students listen, or the professor slacks off and students sleep. But these have quite different properties. Thus, if we hope to use PNE as a prescriptive tool for what will happen, then we need to decide on which PNE will be played. This is called the *equilibrium selection problem*, and it is a major issue in general-sum games. There are at least two reasons for this: first, if we want to predict the behavior of players then how do we choose which equilibrium to predict? Second, if we want to prescribe behavior for an individual player, then we cannot necessarily suggest that they play some particular strategy from a Nash equilibrium, because if the other players do not play the same Nash equilibrium then it may be a terrible suggestion (for example, suggesting that the professor plays "Prepare" from the Prepare/Listen equilibrium, when the students are playing the Slack off/Sleep equilibrium would be bad for the professor).

Moreover, pure-strategy equilibria are not even guaranteed to exist, as we saw in the previous lecture with the rock-paper-scissors example.

To fix the existence issue we may consider allowing players to randomize over their choice of strategy (as in rock-paper-scissors where players should randomize uniformly). Let $\sigma_i \in \Delta^{|S_i|}$ denote player i's probability distribution over their strategy, this is called a *mixed strategy.* Let a strategy profile be denoted by $\sigma = (\sigma_1, \dots, \sigma_n)$. By a slight abuse of notation we may rewrite a player's utility function as

$$u_i(\sigma) = \sum_{s \in S} u_i(s) \prod_i \sigma_i(s_i)$$

A (mixed-strategy) Nash equilibrium is a strategy profile σ such that for all pure strategies σ_i' (σ_i' is pure if it puts probability 1 on a single strategy):

$$u_i(\sigma) \geq u_i(\sigma_{-i}, \sigma_i').$$

Now, Nash's theorem says that

Theorem 2. *Any game with a finite set of strategies and a finite set of players has a mixed-strategy Nash equilibrium.*

Since our goal is the prescribe or predict behavior, we would also like to be able to compute a Nash equilibrium. Unfortunately this turns out to be computationally difficult:

Theorem 3. *The problem of computing a Nash equilibrium in general-sum finite games is PPAD-complete.*

We won't go into detail on what the complexity class PPAD is for now, but suffice it to say that it is weaker than the class of NP-complete problems (it is not hard to come up with a MIP for computing a Nash equilibrium, for example), but still believed to take exponential time in the worst case.

As a sidenote, one may make the following observation about why Nash equilibrium does not "fit" in the class of NP-complete problems: typically in NP-completeness we ask questions such as "does there exist a satisfying assignment to this Boolean formula?" But given a particular game, we already know that a Nash equilibrium exists. Thus we cannot ask the type of existence questions typically used in NP-complete problems; we already know the answer. Instead, it is only the task of *finding* one of the solutions that is difficult. This can be a useful notion to keep in mind when encountering other problems that have guaranteed existence. That said, once one asks for additional properties such as "does there exist a Nash equilibrium where the sum of utilities is at least v?" one gets an NP-complete problem.

Given a strategy profile σ, we will often be interested in measuring how "happy" the players are with the outcome of the game under σ. Most commonly, we are interested in the *social welfare* of a strategy profile (and especially for equilibria). The social welfare is the expected value of the sum of the player's utilities:

$$\sum_{i=1}^{n} u_i(\sigma) = \sum_{i=1}^{n} \sum_{s \in S} u_i(s) \prod_{i'=1}^{n} \sigma_{i'}(s_{i'}).$$

We already saw in the Professor's Dilemma that there can be multiple equilibria with wildly different social welfare: when the professor slacks off and the students sleep, the social welfare is zero; when the professor prepares and the students listen, the social welfare is $2 \cdot 10^6$.

2.1.1 Zero-Sum Games

In the special case of a two-player zero-sum game, we have $u_1(s) = -u_2(s) \forall s \in S$. In that case, we can represent our problem as the bilinear saddlepoint problem we saw in the last lecture:

$$\min_{x \in \Delta^n} \max_{y \in \Delta^m} \langle x, Ay \rangle.$$

A first observation one may make is that the minimization problem faced by the x-player is a convex optimization problem, since the max operation is convexity-preserving. This suggests that we should have a lot of algorithmic options to use. This turns out to be true: unlike the general case, we can compute a zero-sum equilibrium in polynomial time using linear programming (LP).

In fact, we have the following stronger statement, which is essentially equivalent to LP duality:

Theorem 4 (von Neumann's minimax theorem)**.** *Every two-player zero-sum game has a unique value v, called the* value of the game*, such that*

$$\min_{x \in \Delta^n} \max_{y \in \Delta^m} \langle x, Ay \rangle = \max_{y \in \Delta^m} \min_{x \in \Delta^n} \langle x, Ay \rangle = v.$$

We will prove a more general version of this theorem when we discuss regret minimization.

Because zero-sum Nash equilibria are min-max solutions, they are the best that a player can do, given a worst-case opponent. This guarantee is the rationale for saying that a given game has been *solved* if a Nash equilibrium has been computed for the game. Some games are trivially solvable, e.g. in rock-paper-scissors we know that the uniform distribution is the only equilibrium. However, this notion has also been applied to quite large games. For example heads-up limit Texas hold'em, one of the smallest poker variants played by humans (which is still a huge game). In 2015, that game was *essentially solved.* The idea of *essentially solving* a game is as follows: we want to compute a strategy that is statistically indistinguishable from a Nash equilibrium in a lifetime of human-speed play. The statistical notion was necessary because the solution was computed using iterative methods that only converge to an equilibrium in the limit (but in practice get quite close very rapidly). The same argument is also used in constructing AIs for even larger two-player zero-sum poker games where we can only try to approximate the equilibrium.

Note that this min-max guarantee of Nash equilibria does not hold in general-sum games. In general-sum games, we have no payoff guarantees if our opponent does not play their part of the same Nash equilibrium that we play. Interestingly, the AI and optimization methods developed for two-player zero-sum poker turned out to still outperform top-tier human players in 6-player no-limit Texas hold'em poker, in spite of these equilibrium selection issues. An AI based on these methods ended up beating professional human players, in spite of the methods having no guarantees on performance, nor even of converging to a general-sum Nash equilibrium.

Here is another interesting property of zero-sum Nash equilibrium: it is interchangeable. Meaning that if you take an equilibrium (x, y) and another equilibrium (x', y') then (x, y') and (x', y) are also equilibria. This is easy to see from the minimax formulation.

2.2 Historical Notes

Daskalakis et al. [47] were the first to show that solving general games is a PPAD-complete problem. Their initial result was for four-player games. Chen et al. [35] showed that the result holds even for two-player general-sum games. NP-completeness of finding Nash equilibria with various properties was shown by Gilboa and Zemel [70] and Conitzer and Sandholm [44].

The result where Heads-up limit Texas hold'em was *essentially solved* was by Bowling et al. [16]. That paper also introduced the notion of "essentially solved." The strong performance against top-tier humans in 6-player poker was shown by Brown and Sandholm [22].

Chapter 3

Auctions and Mechanism Design Intro

3.1 Introduction

In this lecture note we will study the problem of how to aggregate a set of agent preferences into an outcome, ideally in a way that achieves some desirable outcome. Desiderata we might care about include *social welfare*, which is just the sum of the agent's utilities derived from the outcome, or revenue in the context of auctions.

Suppose that we have a car, and we wish to give it to one of n people, with the goal of giving it to the person that would get the most utility out of the car. One thing we could do is ask each person to tell us how much utility they would get out of receiving the car, expressed as some positive number. This, obviously, leads to the "who can name the highest number?" game, since no person will want to tell us how much value they actually place on the car, but will instead try to name as large of a number as possible.

The above, rather silly, example shows that in general we need to be careful about how we design the rules that map the stated preferences by the agents of a mechanism into an outcome. The general field concerned with the design of rules, or *mechanisms* for designing rules that ask agents about their preferences and use that to choose an outcome is called *mechanism design*.

3.2 Auctions

We will mostly focus on the most classical mechanism-design setting: auctions. We will start by considering single-item auctions: there is a single good for sale, and there is a set of n buyers, with each buyer having some value v_i for the good. The goal will be to sell the item via a *sealed-bid* auction, which works as follows:

1. Each bidder i submits a bid $b_i \geq 0$, without seeing the bids of anyone else.
2. The seller decides who gets the good based on the submitted bids.
3. Each buyer i is charged a price p_i which is a function of the bid vector b.

A few things in our setup may seem strange. First, most people would not think of sealed bids when envisioning an auction. Instead, they typically envision what's called the *English auction.* In the English auction, bidders repeatedly call out increasing bids, until the bidding stops, at which point the highest bidder wins and pays their last bid. This auction can be conceptualized as having a price that starts at zero, and then rises continuously, with bidders dropping out as they become priced out. Once only one bidder is left, the increasing price stops and the item is sold to the last bidder at that price. This auction format turns out to be equivalent to the *second-price* sealed-bid auction which we will cover below. Another auction format is the *Dutch auction*, which is less prevalent in practice. It starts the price very high such that nobody is interested, and then continuously drops the price until some bidder says they are interested, at which point they win the item at that price. The Dutch auction is likewise equivalent to the *first-price* sealed-bid auction, which we cover below.

Secondly, it would seem natural to always give the item to the highest bid in step 2, but this is not always done (though we will focus on that rule). Thirdly, the pricing step allows us to potentially charge more bidders than only the winner. This is again done in some reasonable auction designs, though we will mostly focus on auction formats where $p_i = 0$ if i does not win.

When thinking about how buyers are going to behave in an auction, we need to first clarify what each buyer knows about the other bidders. Perhaps the most standard setting is one where each buyer i has some distribution F_i from which they draw their bid is drawn, independently from the distribution for every other buyer. This is known as the *independent private values* (IPV) model. In this model, every buyer knows the distribution of every other buyer, but they only get to observe their own value $v_i \sim F_i$ before choosing their bid b_i. For this model, we need a new game-theoretic equilibrium notion called a *Bayes Nash equilibrium* (BNE). A BNE is a set of mappings $\{\sigma_i\}_{i=1}^n$, where $\sigma_i(v_i)$ specifies the bid that buyer i submits when they have value i, such that for all values v_i and alternative bids b_i, $\sigma_i(v_i)$ achieves at least as much utility as b_i in a Bayesian sense:

$$\mathbb{E}_{v_{-i}\sim F_{-i}}[u_i(\sigma_i(v_i), \sigma_{-i}(v_{-i}))|v_i] \geq \mathbb{E}_{v_{-i}\sim F_{-i}}[u_i(b_i, \sigma_{-i}(v_{-i}))|v_i].$$

In the auction context, $u_i(b_i, \sigma_{-i}(v_{-i}))$ is utility that buyer i derives given the allocation and payment rule. The idea of a BNE works more generally for a game setup where u_i is some arbitrary utility function.

We will now introduce some useful mechanism-design terminology. We will introduce it in this single-item auction context, but it applies more broadly.

Efficiency. An outcome of a single-item auction is *efficient* if the item ends up allocated to the buyer that values it the most. In general mechanism design problems, an efficient outcome is typically taken to be one that maximizes the sum of the agent utilities, which is also known as maximizing the *social welfare.* Alternatively, efficiency is sometimes taken to mean that we get a Pareto-optimal outcome, which is a weaker notion of efficiency than social welfare maximization (convince yourself of this with a small example.)

Revenue. The revenue of a single-item auction is simply the sum of payments made by the bidders.

Truthfulness, strategyproofness, and incentive compatibility. Informally, we say that an auction is *truthful* or *incentive compatible* (IC) if buyers maximize their utility by bidding their true value (i.e. $b_i = v_i$). More formally, an auction is *dominant strategy incentive compatible* (DSIC) if a buyer maximizes their utility by bidding their value, *no matter what everyone else does.* Saying that an auction (or more generally a mechanism) is "truthful" or "strategyproof" typically means that it is DSIC. DSIC auctions are very attractive because it means that buyers do not need to worry about what the other buyers will do: no matter what happens, they should just bid their value. This also means that, as auction designers, we can reasonably expect that buyers will bid their true value (or at least try to, if they are not perfectly capable of estimating it themselves). This makes it much easier to reason about aspects such as efficiency or revenue.

A slightly weaker degree of truthfulness is that of *Bayes-Nash incentive compatibility*: an auction is Bayes-Nash IC if there exists a BNE where every buyer bids their value. It is clear why this notion is less appealing: Now buyers need to worry about whether other buyers are going to bid truthfully. If they think that they will, then we might expect them to bid their value as well. However, if the system starts out in some other state, we might worry that buyers will adapt their bidding over time in a way that pushes them into some other non-truthful equilibrium.

3.2.1 First-price auctions

First-price auctions are perhaps what most people imagine when we say that we are selling our good via a sealed-bid auctions. In first-price auctions, each buyer submits some bid $b_i \geq 0$, and then we allocate the item to the buyer i^* with the highest bid and charge that buyer b_{i^*}. This is also sometimes referred to as *pay-your-bid.*

Let's briefly try to reason about what might happen in a first-price auction. Clearly, no buyer should bid their true value for the good under this mechanism; in that case they receive no utility even when they win. Instead, buyers should *shade* their bids, so that they sometimes win while also receiving strictly positive utility. The problem is that buyers must strategize about how other buyers will bid, in order to figure out how much to shade by.

This issue of shading and guessing what other buyers will bid happened in early Internet ad auctions, where first-price auctions were initially adopted. *Overture* was an early pioneer in selling Internet sponsored search ads via auction. They initially ran first-price auctions, and provided services to MSN and Yahoo (which were popular search engines at the time). Bidding and pricing turned out to be very inefficient, because buyers were constantly changing their bids in order to best respond to each other. Plots of the price history show a clear "sawtooth pattern," where a pair of bidders will take turns increasing their bid by 1 cent each, in order to beat the other bidder. Finally, one of the bidders reaches their valuation, at which point they drop their bid much lower in order to win something else instead. Then, the winner realizes that they should bid much lower, in order to decrease the price they pay. At that point, the bidder that dropped out starts bidding 1 cent more again, and the pattern repeats. This leads to huge price fluctuations, and inefficient allocations, since about half the time the item goes to the bidder with the lower valuation.

All that said, it turns out that there does exist at least one interesting characterization of how bidding should work in a single-item first-price auction (the Overture example technically consists of many "independent" first-price auctions; though that independence does not truly hold as we shall see later).

For this characterization, we assume the following symmetric model: we have n buyers as before, and buyer i assigns value $v_i \in [0, \overline{v}]$ for the good. Each v_i is sampled IID from an increasing distribution function F. F is assumed to have a continuous density f and full support. Each bidder knows their own value v_i, but only knows that the value of each other buyer is sampled according to F.

Given a bid b_i, buyer i earns utility $v_i - b_i$ if they win, and utility 0 otherwise. If there are multiple bids tied for highest then we assume that a winner is picked uniformly at random among the winning bids, and only the winning bidder pays.

It turns out that there exists a *symmetric equilibrium* in this setting, where each bidder bids according to the function

$$\beta(v_i) = \mathbb{E}[Y_1 | Y_1 < v_i],$$

where Y_1 is the random variable denoting the maximum over $n-1$ independently-drawn values from F.

Theorem 5. *If every bidder in a first-price auction bids according to β then the resulting strategy profile is a Bayes-Nash equilibrium.*

Proof. Let $G(y) = F(y)^{n-1}$ denote the distribution function for Y_1.

Suppose all bidders except i bids according to β. The function β is continuous and monotonically increasing: a higher value for v_i simply adds additional values to the highest end of the distribution. As a consequence, the highest bid other than that of bidder i is $\beta(Y_1)$. It follows that bidder i should never bid more than $\beta(\overline{v})$, since that is the highest possible other bid. Now consider bidding $b_i \le \beta(\overline{v})$. Letting z be such that $\beta(z) = b_i$, the expected value that bidder i obtains from bidding b_i is:

$$\begin{aligned}
u_i(b_i, v_i) =& G(z)[v_i - \beta(z)] \\
=& G(z)v_i - G(z)\mathbb{E}[Y_1 | Y_1 < z] && \text{by definition of } \beta(z) \\
=& G(z)v_i - \int_0^z y g(y) dy && \text{by definition of expectation} \\
=& G(z)v_i - G(z)z + \int_0^z G(y) dy && \text{integration by parts} \\
=& G(z)(v_i - z) + \int_0^z G(y) dy
\end{aligned}$$

Now we can compare the values from bidding $\beta(v_i)$ and b_i:

$$\begin{aligned}
\Pi(\beta(v_i), v_i) - \Pi(b_i, v_i) &= G(v_i)(v_i - v_i) + \int_0^{v_i} G(y)dy - G(z)(v_i - z) - \int_0^z G(y)dy \\
&= G(z)(z - v_i) - \int_{v_i}^z G(y)dy
\end{aligned}$$

If $z \ge v_i$ then this is clearly positive since $G(z) \ge G(y)$ for all $y \in [v_i, z]$. If $z \le v_i$, then $G(z) \le G(y)$, and so we have a negative number and subtract a more negative number. ☐

A nice property that follows from the monotonicity of β is that the item is always allocated to the bidder with the highest valuation, and thus the symmetric equilibrium is efficient.

3.2.2 Second-price auctions

Now we look at another pricing rule: the *second-price auction.* The *second-price auction* turns out to simply allow buyers to submit their true value as their bid. In a second-price auction, the winning bidder i^* is charged the *second-highest bid.* It's easy to see that a bidder should simply bid their valuation in this auction format. There are four cases to consider for a non-truthful bid $b_i \neq v_i$:

1. $b_i > v_i \geq b_2$ where b_2 is the second-highest bid. In that case buyer i would have gotten the same utility from bidding v_i.
2. $b_i > b_2 > v_i$ where b_2 is the second-highest bid. In that case buyer i wins, but gets utility $v_i - b_2 < 0$, and they would have been better off bidding their valuation.
3. $b_i < b_2 < v_i$ where b_2 is the second-highest bid. In that case buyer i does not win, but they could have won and gotten strictly positive utility if they had bid their valuation.
4. $b_2 < b_i < v_i$ where b_2 is the second-highest bid. In that case buyer i wins, but they would have won, and paid the same, if they had bid their true value.

It follows that the second-price auction is DSIC, because an agent should report their true valuation no matter what everybody else does. The second-price auction is also efficient, in the sense that it maximizes social welfare (since the item goes to the buyer with the highest value). Finally, it is *computable*, in the sense that it is easy to find the allocation and payments.

Note that the first-price auction is also computable, and under the symmetric equilibrium given in Theorem 5 it is also efficient. But it is not truthful, and it is not hard to come up with a simple discrete setting where there is not even an equilibrium.

3.3 Mechanism Design

More generally, in mechanism design:

- There's a set of outcomes O, and the job of the mechanism is to choose some outcome $o \in O$
- Each agent i has a private *type* $\theta_i \in \Theta_i$, that they draw from some publicly-known distribution F_i
- Each agent i has some publicly-known valuation function $v_i(o|\theta_i)$ that specifies a utility for each outcome, given their type
- The goal of the center is to design a mechanism that maximizes some objective, e.g. social welfare $\sum_i u_i(o|\theta_i)$

A mechanism takes as input a vector of reported types θ from the players, and outputs an outcome, formally it is a function $f : \times_i \Theta_i \to O$ that specifies the outcome that results from every possible set of reported types. In mechanism design with money, we also have a *payment function* $g : \times_i \Theta_i \to \mathbb{R}^n$ that specifies how much each agent pays under the outcome. In less formal terms, a mechanism merely specifies what happens, given the reported types from the agents. In first and second-price auctions the outcome function was the same (allocate to the highest bidder), but the payment function was different. We could potentially also allow randomized mechanism $f : \times_i \Theta_i \to \Delta(O)$ that map to a probability distribution over the outcome space.

How do we analyze what happens in a given mechanism? The ideal answer is that every agent is best off reporting their true type, no matter what everybody else does, i.e. the mechanism should be DSIC. Formally, that would mean that for every agent i, type $\theta_i \in \Theta_i$, any type vector θ_{-i} of the remaining agents, and misreported type $\theta_i' \in \Theta_i$:

$$\mathbb{E}\left[v_i(f(\theta_i, \theta_{-i}))\right] \geq \mathbb{E}\left[v_i(f(\theta_i', \theta_{-i}))\right],$$

where the expectation is over the potential randomness of the mechanism. If there is also a payment function g and agents have *quasi-linear utilities* then the inequality is

$$\mathbb{E}\left[v_i(f(\theta_i, \theta_{-i})) - g(\theta_i, \theta_{-i})\right] \geq \mathbb{E}\left[v_i(f(\theta_i', \theta_{-i})) - g(\theta_i', \theta_{-i})\right],$$

A less satisfying answer is that there exists a Bayes-Nash equilibrium of the game induced by the mechanism, in which every agent reports their true type. Formally, that would mean that for every agent i, type $\theta_i \in \Theta_i$, and misreported type $\theta_i' \in \Theta_i$:

$$\mathbb{E}_{\theta_{-i}}\left[v_i(f(\theta_i, \theta_{-i}))\right] \geq \mathbb{E}_{\theta_{-i}}\left[v_i(f(\theta_i', \theta_{-i}))\right],$$

where the expectation is over the types θ_{-i} of the other agents, and the potential randomness of the mechanism (note the difference to DSIC, where we did not take the expectation over the types of other agents). This constraint just says that reporting their true type should maximize their expected utility, given that everybody else is truthfully reporting. This can likewise be generalized for a payment function g.

In the setting where we can charge money, the *Vickrey-Clarke-Groves* (VCG) mechanism is DSIC and maximizes social welfare. In VCG, after receiving the type vector θ, we pick the outcome o that maximizes the reported welfare. Formally, VCG selects an outcome in the set $\arg\max_{o \in O} \sum_i v_i(o|\theta_i)$. Of course, an agent i can effectively "choose" the allocation by reporting a very high value for a given outcome. The key to making VCG incentive compatible is that we charge each agent their *externality*, which is the amount that their presence in the markets harms the sum of utilities over the remaining agents. Formally, the externality, and thus payment, of agent i is:

$$\max_{o' \in O} \sum_{i' \neq i} v_{i'}(o'|\theta_{i'}) - \sum_{i' \neq i} v_{i'}(o|\theta_{i'}).$$

The first term is the maximum social welfare achievable when ignoring the utility of agent i (i.e. how well the remaining agents would have done if i left the market), and the second term is the actual sum of utilities achieved by the remaining agents with agent i present. The utility for agent i under a given outcome is their value for the outcome minus their payment:

$$v_i(o) - \max_{o' \in O} \sum_{i' \neq i} v_{i'}(o'|\theta_{i'}) + \sum_{i' \neq i} v_{i'}(o|\theta_{i'}) = \sum_{i'} v_{i'}(o|\theta_{i'}) - \max_{o' \in O} \sum_{i' \neq i} v_{i'}(o'|\theta_{i'})$$

On the right-hand side, we collect all agent values for the outcome into a single summation, and we see that this is exactly the social welfare under the outcome o. The second term cannot be affected by agent i; their reported type does not factor into the maximization. Thus, the only thing that agent i can do is try to maximize social welfare, which is achieved by reporting their true type θ_i.

3.4 Historical Notes

The issues with first-price in the context of Overture's sponsored search auctions is described in Edelman and Ostrovsky [52], which also shows plots from real data exhibiting the sawtooth pattern. The derivation of the symmetric equilibrium of the first-price auction follows the proof from Krishna [84]. Interestingly, first-price auctions have experienced a resurgence in the context of display advertising, where many independent ad exchanges moved to first price in 2018, and Google followed suit and moved their Ad Manager to first price in 2019[1].

The second-price auction is sometimes referred to as the *Vickrey auction* after its inventor [130]. The generalized second-price auction was described by Edelman et al. [53], though it had been in use in the Internet ad industry for a while at that point. The VCG mechanism was described in a series of papers by Vickrey [130], Clarke [40], and Groves [74]. A full description of a slightly more general VCG mechanism, and proof of correctness, can be found in Nisan et al. [103, Chapter 9]

[1]see `https://www.blog.google/products/admanager/update-first-price-auctions-google-ad-manager/`

Chapter 4

Regret Minimization and Sion's Minimax Theorem

So far we have mostly discussed the *existence* of game-theoretic equilibria such as Nash equilibrium. Now we will get started on how to *compute* Nash equilibria, specifically in two-player zero-sum games. The fastest methods for computing large-scale zero-sum Nash equilibrium are based on what's called *regret minimization.* Regret minimization is a form of single-agent decision making, where the decision maker repeatedly chooses decision from a set of possible choices, and each time they make a decision, they are then given some *loss vector* specifying how much loss they incurred through their decision. It may seem counterintuitive that we move on to a single-agent problem after discussing game-theoretic problems with two or more players, but we shall see that regret minimization can be used to *learn* how to play a game. We will also use it to prove a fairly general version of von Neumann's minimax theorem, a variant that is known as *Sion's minimax theorem.*

4.1 Regret Minimization

In the simplest regret-minimization setting we imagine that we are faced with the task of repeatedly choosing among a finite set of n actions. At each time step, we choose an action, and then a loss $g_{ti} \in [0, 1]$ is revealed for each action i. The loss is how *unhappy* we are with having chosen action i, and the goal is to minimize losses over time. This scenario is then repeated iteratively. The key is that the losses may be adversarially chosen after we make our choice, and we would like to come up with a decision-making procedure that does at least as well as the single best action in hindsight. We will be allowed to choose a distribution over actions, rather than a single action, at each decision point. Classical example applications would be picking stocks, picking which route to take to work in a routing problem, or weather forecasting. To be concrete, imagine that we have n weather-forecasting models that we will use to forecast the weather each day. We would like to decide which model is best to use, but we're not sure how to pick the best one. In that case, we may run a regret-minimization algorithm, where our "action" is to pick a model, or a probability distribution over models, to forecast the weather with. If we spend enough days forecasting, then we will show that it is possible for our *average* prediction to be as good as the best single model in hindsight. As can be seen from the above examples, regret minimization methods are widely applicable beyond equilibrium computation and a useful tool to know about.

4.1.1 Setting

Formally, we are faced with the following problem: at each time step $t = 1, \dots, T$:

1. We must choose a decision $x_t \in \Delta^n$

2. Afterwards, a loss vector $g_t \in [0, 1]^n$ is revealed to us, and we pay the loss $\langle g_t, x_t \rangle$

Our goal is to develop an algorithm that recommends good decisions. A natural goal would be to do as well as the best sequence of actions in hindsight. But this turns out to be too ambitious, as the following example shows

Example 1. *We have 2 actions a_1, a_2. At timestep t, if our algorithm puts probability greater than $\frac{1}{2}$ on action a_1, then we set the loss to $(1, 0)$, and vice versa we set it to $(0, 1)$ if we put less than $\frac{1}{2}$ on a_1. Now we face a loss of at least $\frac{T}{2}$, whereas the best sequence in hindsight has a loss of* 0.

Instead, our goal will be to minimize *regret*. The regret at time t is how much worse our sequence of actions is, compared to the best single action in hindsight:

$$R_t = \sum_{\tau=1}^{t} \langle g_\tau, x_\tau \rangle - \min_{x \in \Delta^n} \sum_{\tau=1}^{t} \langle g_\tau, x \rangle$$

We say that an algorithm is a *no-regret algorithm* if for every $\epsilon > 0$, there exists a sufficiently-large time horizon T such that $\frac{R_T}{T} \leq \epsilon$.

Let's see an example showing that randomization is necessary. Consider the following natural algorithm: at time t, choose the action that minimizes the loss seen so far, where e_i is the vector of all zeroes except index i is 1:

$$x_{t+1} = \operatorname*{argmin}_{x \in \{e_1, \ldots, e_n\}} \sum_{\tau=1}^{t} \langle g_\tau, x \rangle. \tag{FTL}$$

This algorithm is called *follow the leader* (FTL). Note that it always chooses a deterministic action. The following example shows that FTL, as well as any other deterministic algorithm, cannot be a no-regret algorithm

Example 2. *At time t, say that we recommend action i. Since the adversary gets to choose the loss vector after our recommendation, let them choose the loss vector be such that $g_i = 1$, $g_j = 0 \forall j \neq i$. Then our deterministic algorithm has loss T at time T, whereas the cost of the best action in hindsight is at most $\frac{T}{n}$.*

It is also possible to derive a lower bound showing that any algorithm must have regret at least $O(\sqrt{T})$ in the worst case, see e.g. [113] Example 17.5.

4.1.2 The Hedge Algorithm

We now show that, while it is not possible to achieve no-regret with deterministic algorithms, it is possible with randomized ones. We will consider the *Hedge* algorithm. It works as follows:

- At $t = 1$, initialize a weight vector w^1 with $w_i^1 = 1$ for all actions i
- At time t, choose actions according to the probability distribution $x_{t,i} = \frac{w_i^t}{\sum_j w_j^t}$
- After observing g_t, set $w_i^{t+1} = w_i^t \cdot e^{-\eta g_{t,i}}$, where η is a stepsize parameter

The stepsize η controls how aggressively we respond to new information. If $g_{t,i}$ is large then we decrease the weight w_i more aggressively.

Theorem 6. *Consider running Hedge for T timesteps. Hedge satisfies*

$$R_T \leq \frac{\eta T}{2} + \frac{\log n}{\eta}$$

Proof. Let g_t^2 denote the vector of squared losses. Let $Z_t = \sum_j w_j^t$ be the sum of weights at time t. We have

$$\begin{aligned} Z_{t+1} &= \sum_{i=1}^n w_i^t e^{-\eta g_{t,i}} \\ &= Z_t \sum_{i=1}^n x_{t,i} e^{-\eta g_{t,i}} \\ &\le Z_t \sum_{i=1}^n x_{t,i}(1 - \eta g_{t,i} + \frac{\eta^2}{2} g_{t,i}^2) \\ &= Z_t(1 - \eta\langle x_t, g_t\rangle + \frac{\eta^2}{2}\langle x_t, g_t^2\rangle) \\ &\le Z_t e^{-\eta\langle x_t, g_t\rangle + \frac{\eta^2}{2}\langle x_t, g_t^2\rangle} \end{aligned}$$

where the first inequality uses the second-order Taylor expansion $e^{-x} \le 1 - x + \frac{x^2}{2}$ and the second inequality uses $1 + x \le e^x$.

Telescoping and using $Z_1 = n$, we get

$$Z_{T+1} \le n \prod_{t=1}^T e^{-\eta\langle x_t, g_t\rangle + \frac{\eta^2}{2}\langle x_t, g_t^2\rangle} = n e^{-\eta \sum_{t=1}^T \langle x_t, g_t\rangle + \frac{\eta^2}{2}\sum_{t=1}^T \langle x_t, g_t^2\rangle}$$

Now consider the best action in hindsight i^*. We have

$$e^{-\eta \sum_{t=1}^T g_{t,i^*}} = w_{i^*}^{T+1} \le Z_{T+1} \le n e^{-\eta \sum_{t=1}^T \langle x_t, g_t\rangle + \frac{\eta^2}{2}\sum_{t=1}^T \langle x_t, g_t^2\rangle}$$

Taking logs gives

$$-\eta \sum_{t=1}^T g_{t,i^*} \le \log n - \eta \sum_{t=1}^T \langle x_t, g_t\rangle + \frac{\eta^2}{2}\sum_{t=1}^T \langle x_t, g_t^2\rangle.$$

Now we rearrange to get

$$R_T \le \frac{\log n}{\eta} + \frac{\eta}{2}\sum_{t=1}^T \langle x_t, g_t^2\rangle \le \frac{\log n}{\eta} + \frac{\eta T}{2},$$

where the last inequality follows from $x_t \in \Delta^n$ and $g_t \in [0,1]^n$. □

If we know T in advance we can now set $\eta = \frac{1}{\sqrt{T}}$ to get that Hedge is a no-regret algorithm.

4.2 Online Convex Optimization

In OCO, we are faced with a similar, but more general, setting than in the regret-minimization setup from last time. In the OCO setting, we are making decisions from some compact convex set $X \in \mathbb{R}^n$ (analogous to the fact that we were previously choosing probability distributions from Δ^n). After choosing a decision x_t, we suffer a convex loss $f_t(x_t)$. We will assume that f_t is differentiable for convenience, but this assumption is not necessary.

As before, we would like to minimize the regret:

$$R_T = \sum_{t=1}^T f_t(x_t) - \min_{x \in X} \sum_{t=1}^T f_t(x)$$

We saw in the last lecture that the follow-the-leader (FTL) algorithm, which always picks the action that minimizes the sum of losses seen so far, does not work. That same argument carries over to the OCO setting. The basic problem with FTL is that it is too unstable: If we consider a setting with $X = [-1,1]$ and

$f_1(x) = \frac{1}{2}x$ and f_t alternates between $-x$ and x then we get that FTL flip-flops between -1 and 1, since they become alternately optimal, and always end up being the wrong choice for the next loss.

This motivates the need for a more stable algorithm. What we will do is to smooth out the decision made at each point in time. In order to describe how this smoothing out works we need to take a detour into *distance-generating functions*.

4.3 Distance-Generating Functions

A distance-generating function (DGF) is a function $d : X \to \mathbb{R}$ which is continuously differentiable on the interior of X, and strongly convex with modulus 1 with respect to a given norm $\|\cdot\|$, meaning

$$d(x) + \langle \nabla d(x), x' - x\rangle + \frac{1}{2}\|x' - x\|^2 \leq d(x') \forall x, x' \in X$$

If d is twice differentiable on $\operatorname{int} X$ then the following definition is equivalent:

$$\langle h, \nabla^2 d(x) h\rangle \geq \|h\|^2, \ \forall x \in X, h \in \mathbb{R}^n$$

Intuitively, strong convexity says that the gap between d and its first-order approximation should grow at a rate of at least $\|x - x'\|^2$. Graphically, we can visualize the 1-dimensional version of this as follows:

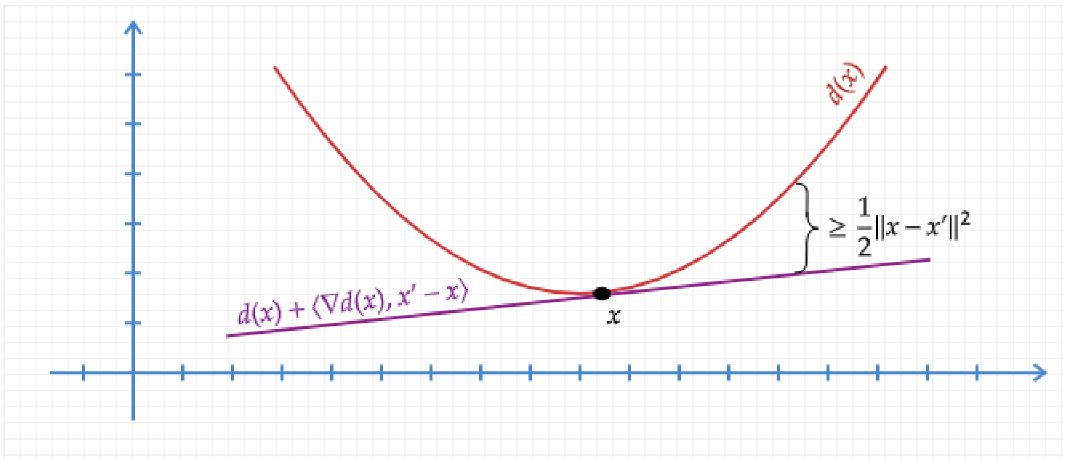

$d(x)$
$\geq \frac{1}{2}\|x - x'\|^2$
$d(x) + \langle \nabla d(x), x' - x\rangle$
x

Figure 4.1: Strong convexity illustrated. The gap between the distance function and its first-order approximation should grow at least as $\|x - x'\|^2$.

We will use this gap to construct a distance function. In particular, we say that the *Bregman divergence* associated with a DGF d is the function:

$$D(x'\|x) = d(x') - d(x) - \langle \nabla d(x), x' - x\rangle.$$

Intuitively, we are measuring the distance going from x to x'. Note that this is not symmetric, the distance from x' to x may be different, and so it is not a true distance metric.

Given d and our choice of norm $\|\cdot\|$, the performance of our algorithms will depend on the *set width* of X with respect to d:

$$\Omega_d = \max_{x, x' \in X} d(x) - d(x'),$$

and the dual norm of $\|\cdot\|$:

$$\|g\|_* = \max_{\|x\| \leq 1} \langle g, x\rangle.$$

In particular, we will care about the largest possible loss vector g that we will see, as measured by the dual norm $\|g\|_*$.

Norms and their dual norm satisfy a useful inequality that is often called the Generalized Cauchy-Schwarz inequality:

$$\langle g, x\rangle = \|x\| \left\langle g, \frac{x}{\|x\|} \right\rangle \leq \|x\| \max_{\|x'\|\leq 1} \langle g, x'\rangle \leq \|x\|\|g\|_*$$

What's the point of these DGFs, norms, and dual norms? The point is that we get to choose all of these in a way that fits the "geometry" of our set X. This will become important later when we will derive convergence rates that depend on Ω and L, where L is an upper bound on the dual norm $\|g\|_{X,*}$ of all loss vectors.

Consider the following two DGFs for the probability simplex $\Delta^n = \{x : \sum_i x_i = 1, x \geq 0\}$:

$$d_1(x) = \sum_i x_i \log(x_i), \quad d_2(x) = \frac{1}{2}\sum_i x_i^2.$$

The first is the *entropy DGF*, the second is the *Euclidean DGF*. First let us check that they are both strongly convex on Δ^n. The Euclidean DGF is clearly strongly convex wrt. the ℓ_2 norm. It turns out that the entropy DGF is strongly-convex wrt. the ℓ_1 norm. Using the second-order definition of strong convexity and any $h \in \mathbb{R}^n$:

$$\begin{aligned}
\|h\|_1^2 &= \left(\sum_i |h_i|\right)^2 \\
&= \left(\sum_i \sqrt{x_i}\frac{|h_i|}{\sqrt{x_i}}\right)^2 \\
&\leq \left(\sum_i x_i\right)\left(\sum_i \frac{|h_i|^2}{x_i}\right) && \text{by the Cauchy-Schwarz inequality} \\
&= \left(\sum_i \frac{|h_i|^2}{x_i}\right) && \text{because } x \in \Delta^n \\
&= \langle h, \nabla^2 d_1(x) h\rangle
\end{aligned}$$

But now imagine that our losses are in $[0,1]^n$. The maximum dual norm for the Euclidean DGF is then

$$\max_{\|x\|_2\leq 1} \langle \vec{1}, x\rangle = \left\langle \vec{1}, \frac{\vec{1}}{\sqrt{n}} \right\rangle = \sqrt{n},$$

while $\Omega_{d_2} = 1$.

In contrast, the maximum dual norm for the ℓ_1 norm is

$$\max_{\|x\|_1\leq 1} \langle \vec{1}, x\rangle = \|\vec{1}\|_\infty = 1.$$

and the set width of the entropy DGF is $\Omega_{d_1} = \log n$.

Thus if our convergence rate is of the form $O\left(\frac{\Omega L}{\sqrt{T}}\right)$, then the entropy DGF gives us a $\log n$ dependence on the dimension n of the simplex, whereas the Euclidean DGF leads to a $\sqrt{n}$ dependence. This shows the well-known fact that the entropy DGF is the "right" DGF for the simplex (from a theoretical standpoint, things turn out to be quite different in numerical performance as we shall see later in the course).

We will need the following inequality on a given norm and its dual norm:

$$\langle g, x\rangle \leq \frac{1}{2}\|g\|_*^2 + \frac{1}{2}\|x\|^2. \tag{4.1}$$

which follows from

$$\langle g, x\rangle - \frac{1}{2}\|x\|^2 \leq \|g\|_*\|x\| - \frac{1}{2}\|x\|^2 \leq \frac{1}{2}\|g\|_*^2$$

where the first step is by the generalized Cauchy-Schwarz inequality and the second step is by maximizing over x.

We will also need the following result concerning Bregman divergences.

Lemma 1 (Three-point lemma)**.** *For any three points x, u, z, we have*

$$D(u\|x) - D(u\|z) - D(z\|x) = \langle \nabla d(z) - \nabla d(x), u - z\rangle$$

The proof is direct from expanding definitions and canceling terms. The left-hand side is analogous to the triangle inequality. The right-hand side characterizes the difference between the two sides of the "triangle inequality." Unlike the triangle inequality, here the right-hand side is not guaranteed to be negative. The right-hand side can be seen as an adjustment to the first-order approximation of d at z to u: we subtract out the first-order approximation at x and add in the first-order approximation at z.

4.4 Online Mirror Descent

We now cover one of the canonical OCO algorithms: *Online Mirror Descent* (OMD). In this algorithm, we smooth out the choice of x_{t+1} in FTL by penalizing our choice by the Bregman divergence $D(x\|x_t)$ from x_t. This has the effect of stabilizing the algorithm, where the stability is essentially due to the strong convexity of d. We pick our iterates as follows:

$$x_{t+1} = \operatorname*{argmin}_{x\in X} \langle \eta\nabla f_t(x), x\rangle + D(x\|x_t).$$

where $\eta > 0$ is the stepsize.

For this algorithm to be well-defined we also need one of the following assumptions:

$$\lim_{x\to\partial X} \|\nabla d(x)\| = +\infty \tag{4.2}$$

$$\tag{4.3}$$

or d should be continuously differentiable on all of X.

To ease notation a bit, we let $g_t = \nabla f_t(x_t)$ throughout the section.

We first prove what is sometimes called a *descent lemma* or *fundamental inequality* for OMD[1].

Theorem 7. *For all $x \in X$, we have*

$$\eta(f_t(x_t) - f_t(x)) \le \eta\langle g_t, x_t - x\rangle \le D(x\|x_t) - D(x\|x_{t+1}) + \frac{\eta^2}{2}\|g_t\|_*^2$$

Proof. The first inequality in the theorem is direct from convexity of f_t. Thus we only need to prove the second inequality.

By first-order optimality of x_{t+1} we have

$$\langle \eta g_t + \nabla d(x_{t+1}) - \nabla d(x_t), x - x_{t+1}\rangle \ge 0, \forall x \in X \tag{4.4}$$

Now pick some arbitrary $x \in X$. By rearranging terms and adding and subtracting $\langle \nabla d(x_{t+1}) - \nabla d(x_t), x - x_{t+1}\rangle$ we have

$$\begin{aligned}
\langle \eta g_t, x_t - x\rangle =& \langle \nabla d(x_t) - \nabla d(x_{t+1}) - \eta g_t, x - x_{t+1}\rangle + \langle \nabla d(x_{t+1}) - \nabla d(x_t), x - x_{t+1}\rangle \\
&+ \langle \eta g_t, x_t - x_{t+1}\rangle \\
\le& \langle \nabla d(x_{t+1}) - \nabla d(x_t), x - x_{t+1}\rangle + \langle \eta g_t, x_t - x_{t+1}\rangle; \quad \text{by (4.4)} \\
=& D(x\|x_t) - D(x\|x_{t+1}) - D(x_{t+1}\|x_t) + \langle \eta g_t, x_t - x_{t+1}\rangle; \quad \text{by three-points lemma} \\
\le& D(x\|x_t) - D(x\|x_{t+1}) - D(x_{t+1}\|x_t) + \frac{\eta^2}{2}\|g_t\|_*^2 + \frac{1}{2}\|x_t - x_{t+1}\|^2; \quad \text{by (4.1)} \\
\le& D(x\|x_t) - D(x\|x_{t+1}) + \frac{\eta^2}{2}\|g_t\|_*^2; \quad \text{by strong convexity of } d,
\end{aligned}$$

which proves the theorem. □

[1]Our proof follows the one from the excellent lecture notes of Orabona [105]. See also Beck [8] for a proof of the offline variant of mirror descent.

The descent lemma gives us a one-step upper bound on how much better x is than x_t. Based on the descent lemma, a bound on the regret of OMD can be derived. The idea is to apply the descent lemma at each time step, and then showing that when we sum across the resulting inequalities, a sequence of useful cancellations occur.

Theorem 8. *The OMD algorithm with DGF d achieves the following bound on regret:*

$$R_T \leq \frac{D(x\|x_1)}{\eta} + \frac{\eta}{2}\sum_{t=1}^{T} \|g_t\|_*^2$$

Proof. Consider any $x \in X$. Now we apply the inequality from Theorem 7 separately to each time step $t = 1, \dots, T$, divide through by η, and then summing from $t = 1, \dots, T$ we get

$$\begin{aligned}\sum_{t=1}^{T} \langle g_t, x - x_t\rangle &\leq \sum_{t=1}^{T} \frac{1}{\eta}\left(D(x\|x_t) - D(x\|x_{t+1}) + \frac{\eta^2}{2}\|g_t\|_*^2 \right) \\ &\leq \frac{D(x\|x_1) - D(x\|x_{T+1})}{\eta} + \sum_{t=1}^{T} \frac{\eta}{2}\|g_t\|_*^2 \\ &\leq \frac{D(x\|x_1)}{\eta} + \sum_{t=1}^{T} \frac{\eta}{2}\|g_t\|_*^2\end{aligned}$$

where the second inequality is by noting that the term $D(x\|x_t)$ appears with a positive sign at the t'th part of the sum, and negative sign at the $t-1$'th part of the sum. □

Suppose that each f_t is Lipschitz in the sense that $\|g_t\|_* \leq L$, using our bound Ω on DGF differences, and supposing we initialize x_1 at the minimizer of d, then we can set $\eta = \frac{\sqrt{2\Omega}}{L\sqrt{T}}$ to get

$$R_T \leq \frac{\Omega}{\eta} + \frac{\eta T L^2}{2} \leq \sqrt{2\Omega T}L$$

A related algorithm is the *follow-the-regularizer-leader* algorithm. It works as follows:

$$x_{t+1} = \operatorname*{argmin}_{x\in X} \eta\langle \sum_{\tau=1}^{t} g_t, x\rangle + d(x).$$

Note that it is more directly related to FTL: it uses the FTL update, but with a single smoothing term $d(x)$, whereas OMD re-centers a Bregman divergence at $D(\cdot\|x_t)$ at every iteration. FTRL can be analyzed similarly to OMD. It gives the same theoretical properties for our purposes, but we will see some experimental performance from both algorithms later where the performance differs quite a bit. For a convergence proof see Orabona [105].

4.5 Minimax theorems via OCO

In the first and second chapters we saw von Neumann's minimax theorem, which was:

Theorem 9 (von Neumann's minimax theorem). *Every two-player zero-sum game has a unique value v, called the* value of the game*, such that*

$$\min_{x\in\Delta^n} \max_{y\in\Delta^m} \langle x, Ay\rangle = \max_{y\in\Delta^m} \min_{x\in\Delta^n} \langle x, Ay\rangle = v.$$

We will now prove a generalization of this theorem.

Theorem 10 (Generalized minimax theorem)**.** *Let $X \in \mathbb{R}^n, Y \in \mathbb{R}^m$ be compact convex sets. Let $f(x,y)$ be continuous, convex in x for a fixed y, and concave in y for a fixed x, with some upper bound L on the partial subgradients with respect to x and y. Then there exists a value v such that*

$$\min_{x\in X}\max_{y\in Y} f(x,y) = \max_{y\in Y}\min_{x\in X} f(x,y) = v.$$

Proof. We will view this is a game between a player choosing the minimizer and a player choosing the maximizer. Let y^* be the y chosen when y is chosen first. When y is chosen second, the maximizer over y can, in the worst case, pick at least y^* every time. Thus we get

$$\max_{y\in Y}\min_{x\in X} f(x,y) \le \min_{x\in X}\max_{y\in Y} f(x,y)$$

For the other direction we will use our OCO results. We run a repeated game where the players choose a strategy x_t, y_t at each iteration t. The x player chooses x_t according to a no-regret algorithm (say OMD), while y_t is always chosen as $\text{argmax}_{y\in Y} f(x_t, y)$. Let the average strategies be

$$\bar{x} = \frac{1}{T}\sum_{t=1}^T x_t, \quad \bar{y} = \frac{1}{T}\sum_{t=1}^T y_t.$$

Using OMD with the Euclidean DGF (since X is compact this is well-defined), we get the following bound:

$$R_T = \sum_{t=1}^T f(x_t, y_t) - \min_{x\in X}\sum_{t=1}^T f(x, y_t) \le O\left(\sqrt{\Omega T} L\right) \tag{4.5}$$

Now we bound the value of the min-max problem as

$$\min_{x\in X}\max_{y\in Y}\ f(x,y) \le \max_{y\in Y} f(\bar{x}, y) \le \frac{1}{T}\max_{y\in Y}\sum_{t=1}^T f(x_t, y) \le \frac{1}{T}\sum_{t=1}^T f(x_t, y_t),$$

where the first inequality follows because $\bar{x}$ is a valid choice in the minimization over X, the second inequality follows by convexity, and the third inequality follows because y_t is chosen to maximize $f(x_t, y_t)$. Now we can use the regret bound (4.5) for OMD to get

$$\begin{aligned}\min_{x\in X}\max_{y\in Y}\ f(x,y) &\le \frac{1}{T}\min_{x\in X}\sum_{t=1}^T f(x, y_t) + O\left(\frac{\sqrt{\Omega}L}{\sqrt{T}}\right)\\ &\le \min_{x\in X} f(x, \bar{y}) + O\left(\frac{\sqrt{\Omega}L}{\sqrt{T}}\right)\\ &\le \max_{y\in Y}\min_{x\in X} f(x,y) + O\left(\frac{\sqrt{\Omega}L}{\sqrt{T}}\right)\end{aligned}$$

Now taking the limit $T \to \infty$ we get

$$\min_{x\in X}\max_{y\in Y} f(x,y) \le \max_{y\in Y}\min_{x\in X} f(x,y)$$

which concludes the proof. □

For simplicity we assumed continuity of f. The argument did not really need continuity, though. The same proof works for f which is lower/upper semicontinuous in x and y respectively.

4.6 Historical notes

When applied to the offline setting where $f_t = f \ \forall t$, OMD is equivalent to the *mirror descent* algorithm which was introduced by Nemirovsky and Yudin [98], with the more modern variant introduced by Beck and Teboulle [9]. There's a functional-analytic interpretation of OMD and mirror descent where one views d as a *mirror map* that allows us to think of f and x in terms of the dual space of linear forms. This was the original motivation for mirror descent, and allows one to apply the algorithm in broader settings, e.g. Banach spaces. This is described in several textbooks and lecture notes e.g. Orabona [105] or Bubeck et al. [25]. The FTRL algorithm run on an offline setting with $f_t = f$ becomes equivalent to Nesterov's *dual averaging* algorithm [101].

The minimax theorems in Theorem 9 and Theorem 10 were developed by John von Neumann in [131]. The term "von Neumann's minimax theorem" is often used to refer to the specific version in Theorem 9. In his original 1928 paper, von Neumann actually proved a more general result for continuous quasi-convex-quasi-concave functions f, which captures the form given in Theorem 10. See Kjeldsen [81] for a discussion of the history of von Neumann's development and conceptualization of the minimax theorem, including a discussion of the quasi-convex-quasi-concave generalization. The more general Theorem 10, as well as even more general versions that allow quasi-concavity and quasi-convexity and abstract topological decision spaces, are often referred to as *Sion's minimax theorem*[2], sometimes even in cases that fall under von Neumann's generalization beyond the bilinear case. For example, in his 1958 paper [118], Sion claims that von Neumann's theorem is only concerned with bilinear functions, whereas it is actually substantially more general. This misconception that von Neumann only dealt with the bilinear case may have arisen because that is by far the most important case from a game-theoretic perspective (since it enables solutions to two-player zero-sum games). Moreover, von Neumann's original 1928 paper was written in German, and an English translation did not appear until 1958 [132].

[2]A quite general version of what's usually referred to as Sion's minimax theorem can be found on Wikipedia at `https://en.wikipedia.org/wiki/Sion%27s_minimax_theorem`.

Chapter 5

Blackwell Approachability and Regret Matching*

In this lecture we are going to introduce a new type of online-learning problem concerned with *vector-valued games*. This framework will eventually be shown to lead to one of the fastest algorithms for game solving in practice.

5.1 Blackwell Approachability

In two-player zero-sum games we saw that there exists a value for the game v such that the row player can choose a strategy x assuring that the payoff will be in the set $(-\infty, v]$ no matter what the column player does, and vice versa the column player can assure that the payoff lies in the set $[v, \infty)$, no matter what the row player does.

In Blackwell approachability we ask whether there is a way to generalize the notion of forcing the payoffs to lie in a particular set to *vector-valued games.*

We consider the following setup:

- The row and column players choose strategies from compact convex sets X and Y respectively.
- There is a bilinear vector-valued payoff function $f(x, y) \in \mathbb{R}^m$.
- There is a closed convex *target set* C.
- We will assume that $f(x, y) \in B(0, 1), C \subseteq B(0, 1)$, where $B(0, 1) = \{g : \|g\|_2 \leq 1\}$.

The goal for the row player is to get payoffs $f(x, y)$ to lie inside C. The case of a single-shot game is trivially analyzed: it is generally only possible to do this if there exists x such that $f(x, y) \in C$, $\forall y \in Y$. So in general this won't be possible. However, it turns out that much more interesting things can be said about a variant where the two players are playing a repeated game. In particular, the players choose actions x_t, y_t at each timestep t. The goal for the row player is to have the average payoff vector $\bar{f}_t = \frac{1}{t}\sum_{i=1}^{t} f(x_t, y_t)$ approach C, while the goal of the column player is to keep $\bar{f}_t$ from approaching C. We will measure the distance as $d(\bar{f}_t, C) = \min_{z \in C} \|z - \bar{f}_t\|_2$

We will say that

Definition 1. *A target set C is* approachable *if there exists an algorithm for picking x_t based on $x_1, \dots, x_{t-1}, y_1, \dots, y_{t-1}$ such that $d(\bar{f}_t, C) \to 0$ as t goes to infinity.*

A stronger notion is that

Definition 2. *A target set C is* forceable *if there exists x such that $f(x, y) \in C$ for all $y \in Y$.*

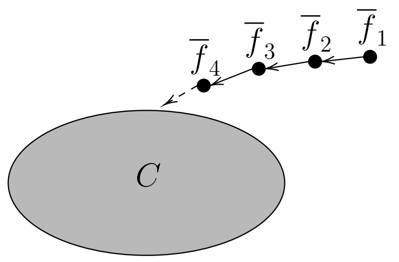


Figure 5.1: Blackwell approachability requires that the sequence $\{\bar{f}_t\}_{t=1}$ approaches C no matter the choices of the y player.

5.1.1 Scalar Approachability

In the special case where $m = 1$ we get a scalar approachability game. As discussed at the beginning of this section, this can be analyzed via minimax theorems. In particular, for the scalar case target sets are intervals, and we may analyze only intervals of the form $(-\infty, \lambda]$ without loss of generality. Clearly, an interval $(-\infty, \lambda]$ is approachable if $\lambda \geq v$, where v is the value of the game associated to the bilinear function f in Sion's minimax theorem. This follows because if the row player plays any strategy x such that they are guaranteed at least v, then $f(x, y_t) \in (-\infty, \lambda]$ for all t no matter the y_t. Conversely, if $\lambda < v$, then by Sion's theorem the column player may play a strategy y such that no matter the x_t, $f(x_t, y) \geq v > \lambda$. We thus have the lemma

Lemma 2. *In scalar approachability games, the following three statements are equivalent:*

- *A target set $(-\infty, \lambda]$ is approachable.*
- *A target set $(-\infty, \lambda]$ is forceable.*
- *$\lambda \geq v$, where v is the value of the game associated to f, X, Y in Sion's minimax theorem.*

5.1.2 Halfspace Approachability

We first analyze the special case where the target set is a halfspace $H = \{h : \langle h, a \rangle \leq b\}$. Halfspaces turn out to have the nice property that forceability is equivalent to approachability:

Lemma 3. *A halfspace H is approachable if and only if it is forceable.*

Proof. The proof consists in reducing halfspace approachability to a scalar approachability game. To do that, let $\hat{f}(x, y) = \langle a, f(x, y) \rangle$. Now we clearly have that forcing H wrt. f is equivalent to forcing $(-\infty, b]$ wrt. $\hat{f}$. Say x^* forces $(-\infty, b]$, then

$$b \geq \hat{f}(x^*, y) = \langle a, f(x^*, y) \rangle, \ \forall y \in Y,$$

and so x^* also forces H, and vice versa.

For approachability we have that the distance from $\bar{f}_t$ to H satisfies

$$d(\bar{f}_t, H) = d\left(\langle a, \bar{f}_t \rangle, (-\infty, b]\right) = d\left(\frac{1}{t}\sum_{i=1}^{t} \langle a, f_i \rangle, (-\infty, b]\right).$$

Thus approachability of H is equivalent to approachability of $(-\infty, b]$.

From Lemma 2 we have that approachability and forceability are equivalent for $(-\infty, b]$, so they must be equivalent for H. $\square$

5.1.3 Blackwell's Approachability Theorem

Now we are ready to analyze the general case of when a convex closed set C is approachable. Blackwell proved the following:

Theorem 11. *A convex closed set C is approachable if and only if every halfspace $H \supseteq C$ is forceable. If every halfspace is forceable then C can be approached at a rate of $\frac{2}{\sqrt{T}}$.*

Blackwell's proof is constructive. It is based on the following algorithm for approaching C when all halfspaces containing C are forceable: At every timestep t, do the following:

- If $\bar{f}_t \in C$, play any x_t.
- Else consider the projection ϕ_t of $\bar{f}_t$ onto C. We construct a halfspace H with normal vector $a_t = \phi_t - \bar{f}_t$, and constant $b_t = \langle a_t, \phi_t \rangle$. Play any x_t forcing H.

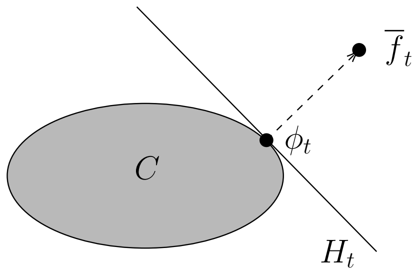


Figure 5.2: The tangent halfspace forced in Blackwell's theorem.

The algorithm repeatedly takes the halfspace tangent to the projection of $\bar{f}_t$, and forces it. We now prove Blackwell's theorem.

Proof. Say that C is approachable. Then we may play any algorithm guaranteed to approach C, and we will then be guaranteed to approach every $H \supseteq C$.

Now assume that all $H \supseteq C$ are approachable, and play Blackwell's algorithm. First note that since ϕ_t is the projection of $\bar{f}_t$ onto a convex set H (this follows from how we constructed H) we have from first-order optimality:

$$\langle \phi_t - \bar{f}_t, z - \phi_t \rangle \geq 0, \ \forall z \in H. \tag{5.1}$$

Let $f_{t+1} = f(x_{t+1}, y_{t+1})$. We have

$$\begin{aligned}
d(\bar{f}_{t+1}, C)^2 &= \min_{z \in C} \|\bar{f}_{t+1} - z\|_2^2 \\
&\leq \|\bar{f}_{t+1} - \phi_t\|_2^2 \\
&= \left\| \frac{t}{t+1}\bar{f}_t + \frac{1}{t+1} f_{t+1} - \phi_t \right\|_2^2 ; \quad \text{by definition of } \bar{f}_{t+1} \\
&= \left\| \frac{t}{t+1}(\bar{f}_t - \phi_t) + \frac{1}{t+1}(f_{t+1} - \phi_t) \right\|_2^2 \\
&= \frac{1}{(t+1)^2}\left(t^2 \|(\bar{f}_t - \phi_t)\|_2^2 + \|(f_{t+1} - \phi_t)\|_2^2 + 2t\langle \bar{f}_t - \phi_t, f_{t+1} - \phi_t \rangle \right) \\
&\leq \frac{1}{(t+1)^2}\left(t^2 \|(\bar{f}_t - \phi_t)\|_2^2 + \|(f_{t+1} - \phi_t)\|_2^2 \right); \quad \text{by (5.1)} \\
&= \frac{1}{(t+1)^2}\left(t^2 d(\bar{f}_t, C)^2 + \|(f_{t+1} - \phi_t)\|_2^2 \right)
\end{aligned}$$

Telescoping this inequality we have

$$d(\bar{f}_{t+1}, C)^2 \leq \frac{1}{(t+1)^2} \sum_{i=1}^{t} \|f_{t+1} - \phi_t\|_2^2 \leq \frac{4t}{(t+1)^2} \leq \frac{4}{t+1},$$

where the second inequality is from the fact that we assumed payoffs to lie in the norm-ball $B(0,1)$. Taking the square root of both sides gives the theorem. □

5.2 Regret Matching

Blackwell's constructive result can easily be converted to a regret minimization algorithm for linear losses over a simplex Δ^n. For each pure action i we say that $r_{t,i} = \langle g_t, x_t \rangle - g_{t,i}$ is the regret from not playing action i rather than x_t, and we let r_t be the vector of all n regrets. We will use $\frac{r}{\sqrt{n}}$ as our vector-valued payoff. Note that the regret is now $R_T = \max_i \sum_{t=1}^{T} r_{t,i}$, and having regret grow sublinearly is equivalent to $\bar{r}_t = \frac{1}{t}\sum_{k=1}^{t} r_k$ approaching the non-positive orthant as t tends to infinity. Thus our target set is $C = \mathbb{R}^n_-$.

By Blackwell's theorem having $\bar{r}_t$ approach $\mathbb{R}^n_-$ can be done by repeatedly forcing tangent halfspaces. To do so, let ϕ_t be the projection of $\bar{r}_t$ onto $\mathbb{R}^n_-$. Note that the normal vector $a_t = \bar{r}_t - \phi_t$ simply thresholds $\bar{r}_t$ at zero, setting all negative entries to zero. Now, we will force r_{t+1} to be in the halfspace with normal vector a_t by ensuring $\langle a_t, r_{t+1} \rangle = 0$. To do so, first consider the square matrix of pairwise regrets B, where B_{ij} is the regret from playing i rather than j under g_{t+1}. We have that $B_{ij} = -B_{ji}$, so B is skew-symmetric, which means that $\langle q, Bq \rangle = 0$ for all q. We can choose $x_{t+1} = \frac{a_t}{\|a_t\|_1}$, in which case we get that the next regret is $r_{t+1} = Bx_{t+1} = B\frac{a_t}{\|a_t\|_1}$, and now it satisfies $\langle a_t, r_{t+1} \rangle = 0$, and thus we forced the desired halfspace.

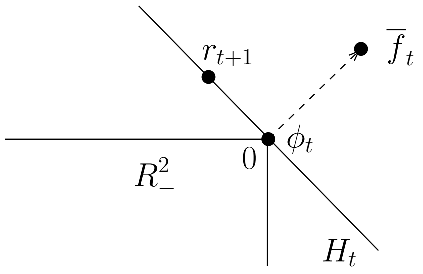


Figure 5.3: The next regret vector r_{t+1} lies in the halfspace forced in Blackwell's theorem.

Summarizing what we did in terms of our standard regret minimization framework, we have an algorithm that works as follows:

- Play arbitrary x_1
- Keep a sum $\hat{r}_t = \sum_{k=1}^{t} r_k$ of regret vectors
- At time $t+1$ set $x_{t+1,i} = \frac{[\hat{r}_{t,i}]^+}{\sum_{k=1}^{n}[\hat{r}_{t,k}]^+}$ ($[\cdot]^+$ denotes thresholding at 0)
- If no regrets are positive, play uniform strategy

This algorithm is called *regret matching*, and by Blackwell's theorem regret matching has regret that grows on the order of $O\left(\sqrt{T}\right)$, assuming $g_t \in B(0,1)$ for all t (if this does not hold we may simply normalize the payoffs).

5.3 Regret Matching$^+$

Finally, we present a variation on regret matching, which turns out to be immensely useful in practice. In regret matching, remember that we took the sum of the regret vectors and thresholded it at zero when generating x_{t+1}. In *regret matching*$^+$ (RM$^+$), we only keep track of positive regrets. Formally, we have the following algorithm:

- initialize $Q_1 = 0$ and play x_1 arbitrarily
- After seeing r_t, set $Q_t = \left[\frac{t-1}{t} Q_{t-1} + \frac{1}{t} r_t\right]^+$
- At time $t+1$, play $x_{t+1,i} = \frac{Q_{t,i}}{\|Q_t\|_1}$

The important observation for RM$^+$ is that we are constructing a sequence that upper-bounds regret, i.e. $Q_t \geq \bar{r}_t$. This is easy to see, as we are only dropping negative terms in the summation that makes up $\bar{r}_t$.

Visually, we may think of it as moving along a face of $\mathbb{R}^n_-$, while maintaining the same distance d to $\mathbb{R}^n_-$ while moving towards 0. See Figure 5.4

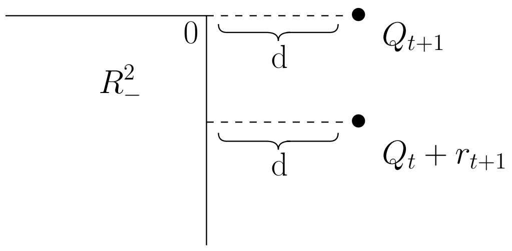


Figure 5.4: The thresholding used in constructing Q_{t+1}.

Theorem 12. RM$^+$ *approaches* $C = \mathbb{R}^n_-$ *at a rate of* $\frac{2}{\sqrt{T+1}}$

Proof. Let Q_t^* be the projection of Q_t onto C. Let H be the halfspace $\{q : \langle Q_t, q\rangle \leq 0\}$ corresponding to forcing in Blackwell's theorem (since $Q_t^* = 0$). We have

$$\begin{aligned}
d(Q_{t+1}, C)^2 &= \min_{z\in C} \|Q_{t+1} - z\|^2 \\
&\leq \|Q_{t+1} - Q_t^*\|^2 \\
&= \|Q_{t+1}\|^2; \text{ since } Q_t^* = 0 \\
&= \left\| \left[\frac{t}{t+1} Q_t + \frac{1}{t+1} r_{t+1}\right]^+ \right\|^2 \\
&\leq \left\| \frac{t}{t+1} Q_t + \frac{1}{t+1} r_{t+1} \right\|^2 \text{; since thresholding can only decrease the norm} \\
&= \frac{1}{(t+1)^2} \left(t^2 \|Q_t\|^2 + \|r_{t+1}\|^2 + 2t\langle Q_t, r_{t+1}\rangle\right) \\
&= \frac{1}{(t+1)^2} \left(t^2 \|Q_t\|^2 + \|r_{t+1}\|^2\right) \text{; by forcing } r_{t+1} \in H.
\end{aligned}$$

By telescoping we now get

$$\begin{aligned}
d(Q_{t+1}, C)^2 &\leq \frac{1}{(t+1)^2} \left(t^2 d(Q_t, C) + \|r_{t+1}\|^2\right) \\
&\leq \frac{1}{(t+1)^2} \sum_{k=1}^{t} \|r_{k+1}\|^2 \\
&\leq \frac{1}{(t+1)^2} 4t \\
&\leq \frac{4}{(t+1)}
\end{aligned}$$

Taking square roots concludes the theorem. □

5.4 Overview of Regret Minimizers

At this point we have covered quite a few regret minimizers. In the coming lectures we will start to look at how they can be used to solve zero-sum games, both matrix games and extensive-form games. For now, let us quickly recap and compare our options. Say that we want to minimize linear losses from $[0,1]^n$ over a simplex Δ^n (note that this covers convex losses with bounded dual norm of the gradients). In that case we have covered 5 algorithms with two different types of regret bounds:

- Regret bound: $O\left(\sqrt{T \log n}\right)$: **Hedge** and **OMD (entropy)**
- Regret bound: $O\left(\sqrt{nT}\right)$: **OMD (Euclidean)**, **Regret Matching**, and **Regret Matching**$^+$.

5.5 Historical Notes and Further Reading

Blackwell approachability was introduced in [14]. Regret matching was introduced by [75]. The RM$^+$ algorithm was introduced in [123] and proven correct by [124]. The proof of RM$^+$ via modified Blackwell approachability is, I believe, new. It was developed together with Gabriele Farina when working on the papers Farina et al. [59, 60].

There aren't many places to find coverage of Blackwell approachability, and furthermore all the sources I know of cover it in quite different ways and levels of generality. Lecture notes 13 and 14 of Ramesh Johari [78] cover the finite-action space case as well as regret matching and the relationship to calibration. Another nice presentation for that same case is the one given by Young [138]. The more general proof of Blackwell's theorem given here largely follows the one given in a blog post by Farina at `http://www.cs.cmu.edu/~gfarina/2016/approachability/`.

Chapter 6

Self-Play via Regret Minimization

6.1 Recap

We have covered a slew of no-regret algorithms: hedge, online mirror descent (OMD), regret matching (RM), and RM$^+$. All of these algorithms can be used for the case of solving two-player zero-sum matrix games of the form

$$\min_{x\in\Delta^n}\max_{y\in\Delta^m}\langle x, Ay\rangle.$$

Matrix games are a special case of the more general saddle-point problem

$$\min_{x\in X}\max_{y\in Y} f(x,y)$$

where f is convex-concave, meaning that $f(\cdot, y)$ is convex for all fixed y, and $f(x,\cdot)$ is concave for all fixed x, and lower/upper semicontinuous. In this chapter we will cover how to solve this more general class of saddle-point problems by using regret minimization for each "player" and having the regret minimizers perform what is usually called *self play*. The name self play comes from the fact that we usually use the same regret-minimization algorithm for each player, and so in a sense this approach towards computing equilibria lets the chosen regret-minimization algorithm play against itself. After covering the self play setup, we will look at some experiments on practical performance for the matrix-game case. We will also compare to an algorithm that has stronger theoretical guarantees.

6.2 From Regret to Nash Equilibrium

In order to use regret-minimization algorithms for computing Nash equilibrium, we will run a repeated game between the x and y players. We will assume that each player has access to some regret-minimizing algorithm RM_x and RM_y (we will be a bit loose with notation here and implicitly assume that RM_x and RM_y keep a state that may depend on the sequence of losses and decisions). The game is as follows:

- Initialize $x_1 \in X, y_1 \in Y$ to be some pair of strategies in the relative interior (in matrix games we usually start with the uniform strategy)
- At time t, let x_t be the recommendation from RM_x and y_t be the recommendation from RM_y
- Let RM_x and RM_y observe losses $g_t = f(\cdot, y_t), \ell_t = f(x_t, \cdot)$ respectively

For a strategy pair $\bar{x}, \bar{y}$, we will measure proximity to Nash equilibrium via the *saddle-point residual* (SPR):

$$\xi(\bar{x},\bar{y}) := \left[\max_{y\in Y} f(\bar{x},y) - f(\bar{x},\bar{y})\right] + \left[f(\bar{x},\bar{y}) - \min_{x\in X} f(x,\bar{y})\right] = \max_{y\in Y} f(\bar{x},y) - \min_{x\in X} f(x,\bar{y}).$$

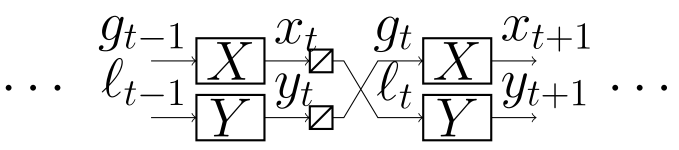


Figure 6.1: The flow of strategies and losses in regret minimization for games.

Each bracketed term represents how much each player can improve by deviating from $\bar{y}$ or $\bar{x}$ respectively, given the strategy profile $(\bar{x}, \bar{y})$. In game-theoretic terms the brackets are how much each player improves by best responding.

Now, suppose that the regret-minimizing algorithms guarantee regret bounds of the form

$$\begin{aligned} \max_{y \in Y} \sum_{t=1}^{T} f(x_t, y) - \sum_{t=1}^{T} f(x_t, y_t) \le \epsilon_y \\ \sum_{t=1}^{T} f(x_t, y_t) - \min_{x \in X} \sum_{t=1}^{T} f(x, y_t) \le \epsilon_x, \end{aligned} \tag{6.1}$$

then the following folk theorem holds

Theorem 13. *Suppose* (6.1) *holds, then for the average strategies* $\bar{x} = \frac{1}{T}\sum_{t=1}^{T} x_t, \bar{y} = \frac{1}{T}\sum_{t=1}^{T} y_t$ *the SPR is bounded by*

$$\xi(\bar{x}, \bar{y}) \le \frac{(\epsilon_x + \epsilon_y)}{T}.$$

Proof. Summing the two inequalities in (6.1) we get

$$\begin{aligned} \epsilon_x + \epsilon_y \ge & \max_{y \in Y} \sum_{t=1}^{T} f(x_t, y) - \sum_{t=1}^{T} f(x_t, y_t) + \sum_{t=1}^{T} f(x_t, y_t) - \min_{x \in X} \sum_{t=1}^{T} f(x, y_t) \\ = & \max_{y \in Y} \sum_{t=1}^{T} f(x_t, y) - \min_{x \in X} \sum_{t=1}^{T} f(x, y_t) \\ = & T \max_{y \in Y} \sum_{t=1}^{T} \frac{1}{T} f(x_t, y) - T \min_{x \in X} \sum_{t=1}^{T} \frac{1}{T} f(x, y_t) \\ \ge & T \left[\max_{y \in Y} f(\bar{x}, y) - \min_{x \in X} f(x, \bar{y}) \right], \end{aligned}$$

where the second inequality is by f being convex-concave.

□

So now we know how to compute a Nash equilibrium: simply run the above repeated game with each player using a regret-minimizing algorithm, and the uniform average of the strategies will converge to a Nash equilibrium.

Figure 6.2 shows the performance of the various regret-minimization algorithms covered so far in the book, when used to compute a Nash equilibrium of a zero-sum matrix game via Theorem 13. Performance is shown on 3 randomized matrix game classes where entries in A are sampled according to: 100-by-100 uniform $[0, 1]$, 500-by-100 standard Gaussian, and 100-by-100 standard Gaussian. All plots are averaged across 50 game samples per setup. We show one addition algorithm for reference: the *mirror prox* algorithm, which is an offline optimization algorithm that converges to a Nash equilibrium at a rate of $O\left(\frac{1}{T}\right)$. It's an accelerated variant of mirror descent, and it similarly relies on a distance-generating function d. The plot shows mirror prox with the Euclidean distance.

As we see in Figure 6.2, mirror prox indeed performs better than all the $O\left(\frac{1}{\sqrt{T}}\right)$ regret minimizers using the setup for Theorem 13. On the other hand, the entropy-based variant of OMD, which has a $\log n$ dependence on the dimension n, performs much worse than the algorithms with $\sqrt{n}$ dependence.

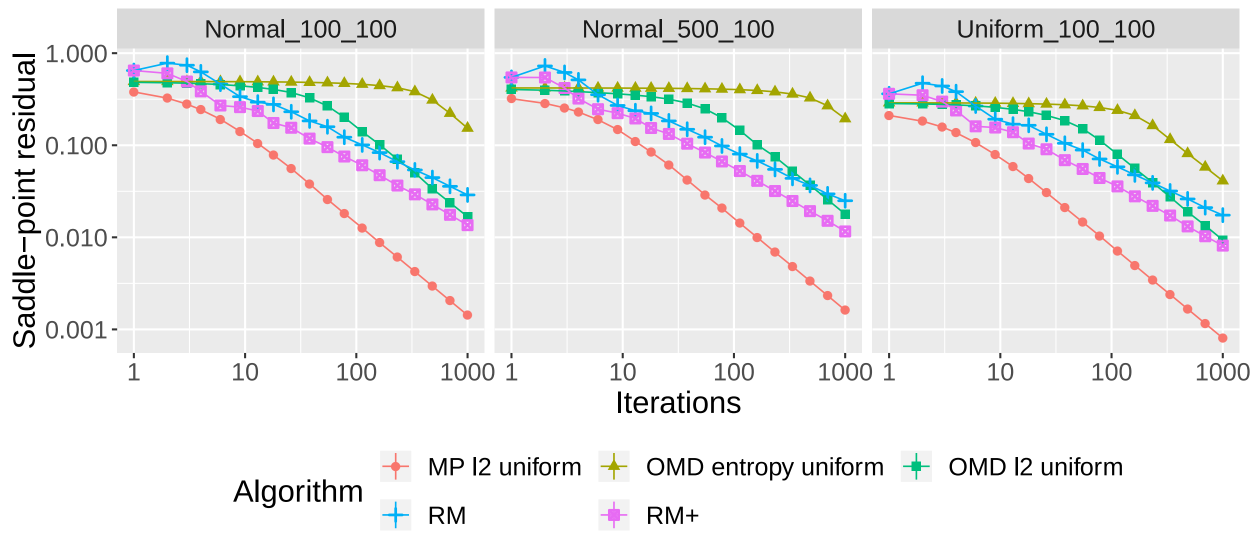


Figure 6.2: Plots showing the performance of four different regret-minimization algorithms for computing Nash equilibrium, all using Theorem 13. Mirror prox with uniform averaging is also shown as a reference point.

6.3 Alternation

Let's try making a small tweak now; the idea of *alternation*. In alternation, the players are no longer symmetric: one player sees the loss based on the previous strategy of the other player as before, but the second player sees the loss associated to the current strategy.

- Initialize x_1, y_1 to be uniform distributions over actions
- At time t, let x_t be the recommendation from RM_x
- The y player observes loss $f(x_t, \cdot)$
- y_t is the recommendation from RM_y after observing $f(x_t, \cdot)$
- The x player observes loss $f(\cdot, y_t)$

Suppose that the regret-minimizing algorithms guarantee regret bounds of the form

$$\begin{aligned} \max_{y\in Y} \sum_{t=1}^{T} f(x_{t+1}, y) - \sum_{t=1}^{T} f(x_{t+1}, y_t) &\le \epsilon_y \\ \sum_{t=1}^{T} f(x_t, y_t) - \min_{x\in X} \sum_{t=1}^{T} f(x, y_t) &\le \epsilon_x. \end{aligned} \tag{6.2}$$

Theorem 14. *Suppose we run two regret minimizers with alternation and they give the guarantees in (6.2). Then the average strategies* $\bar{x} = \frac{1}{T}\sum_{t=1}^{T} x_{t+1}, \bar{y} = \frac{1}{T}\sum_{t=1}^{T} y_t$ *satisfy*

$$\xi(\bar{x}, \bar{y}) \le \frac{\epsilon_x + \epsilon_y + \sum_{t=1}^{T} \left(f(x_{t+1}, y_t) - f(x_t, y_t)\right)}{T}$$

Proof. As before we sum the regret bounds to get

$$\begin{aligned}
\epsilon_x + \epsilon_y \geq& \max_{y \in Y} \sum_{t=1}^{T} f(x_{t+1}, y) - \sum_{t=1}^{T} f(x_{t+1}, y_t) + \sum_{t=1}^{T} f(x_t, y_t) - \min_{x \in X} \sum_{t=1}^{T} f(x, y_t) \\
=& \max_{y \in Y} \sum_{t=1}^{T} f(x_{t+1}, y) - \min_{x \in X} \sum_{t=1}^{T} f(x, y_t) - \sum_{t=1}^{T} [f(x_{t+1}, y_t) - f(x_t, y_t)] \\
\geq& T \left[\max_{y \in Y} f(\bar{x}, y) - \min_{x \in X} f(x, \bar{y}) \right] - \sum_{t=1}^{T} [f(x_{t+1}, y_t) - f(x_t, y_t)]
\end{aligned}$$

□

Theorem 14 shows that if $f(x_{t+1}, y_t) - f(x_t, y_t) \leq 0$ for all t, then the bound for alternation is weakly better than the bound in Theorem 13. But what does this condition mean? If we examine it from the regret minimization perspective, it is saying that x_{t+1} does better than x_t against y_t. Intuitively, we would expect this to hold: x_t is chosen right before observing $f(\cdot, y_t)$, whereas x_{t+1} is chosen immediately after observing $f(\cdot, y_t)$, and generally we would expect that any time we make a new observation, we should move somewhat in the direction of improvement against that observation. Indeed, it turns out to be relatively straightforward to show that this holds for all the regret minimizers we saw so far (As an exercise, show that this holds for a few regret minimizers; it is easiest for OMD).

Figure 6.3 shows the performance of the same set of regret-minimization algorithms but now using the setup from Theorem 14. Mirror prox is shown exactly as before.

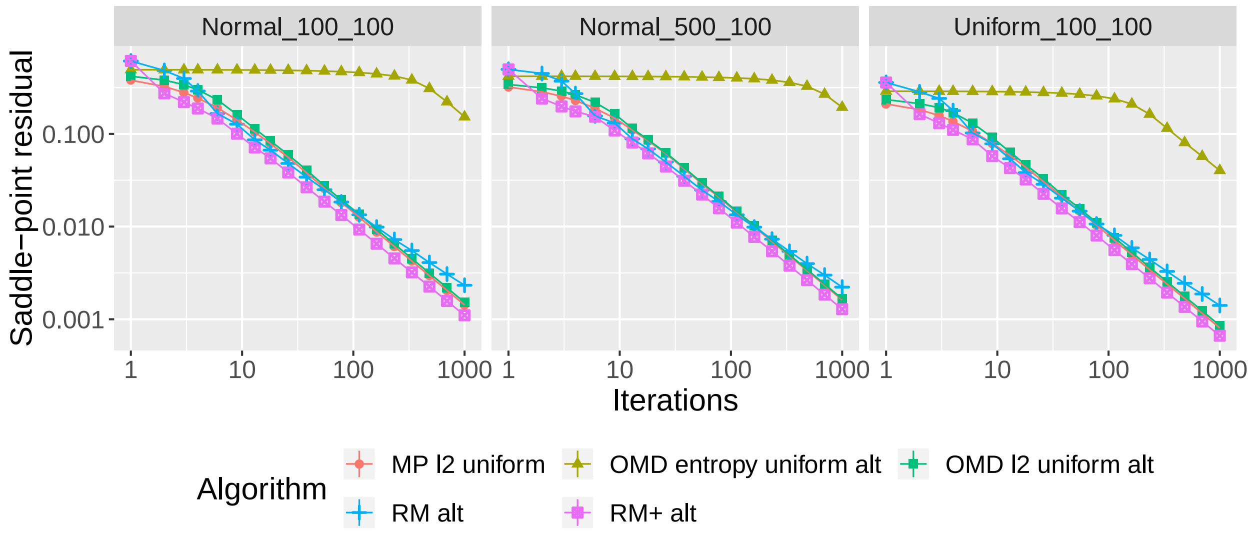


Figure 6.3: Plots showing the performance of four different regret-minimization algorithms for computing Nash equilibrium, all using Theorem 14. Mirror prox with uniform averaging is also shown as a reference point.

Amazingly, Figure 6.3 shows that with alternation, OMD with the Euclidean DGF, regret matching, and RM$^+$ all perform about on par with mirror prox.

6.4 Increasing Iterate Averaging

Now we will look at one final trick. In Theorems 13 and 14 we generated a solution by uniformly averaging iterates. We will now consider polynomial averaging schemes of the form

$$\bar{x} = \frac{1}{\sum_{t=1}^{T} t^q} \sum_{t=1}^{T} t^q x_t, \quad \bar{y} = \frac{1}{\sum_{t=1}^{T} t^q} \sum_{t=1}^{T} t^q y_t.$$

Figure 6.4 shows the performance of the same set of regret-minimization algorithms but now using the setup from Theorem 14 and linear averaging in all algorithms, including mirror prox. The fastest algorithm

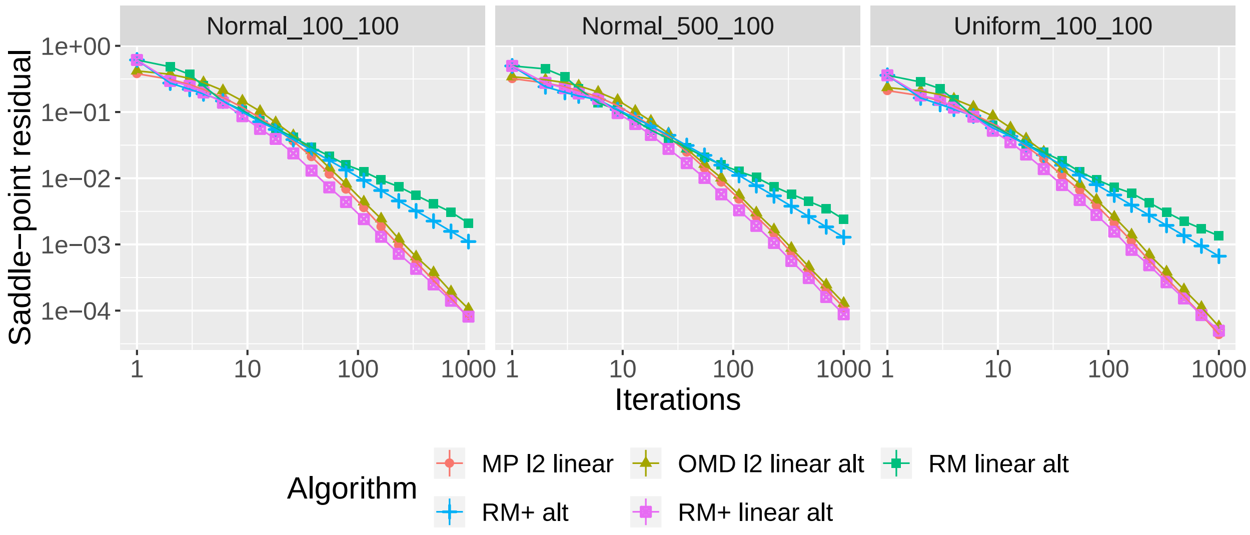


Figure 6.4: Plots showing the performance of four different regret-minimization algorithms for computing Nash equilibrium, all using Theorem 14. All algorithms use linear averaging. RM$^+$ with uniform averaging is shown as a reference point.

with uniform averaging, RM$^+$ with alternation, is shown for reference. OMD with Euclidean DGF and RM$^+$ with alternation both gain another order of magnitude in performance by introducing linear averaging.

It can be shown that RM$^+$, online mirror descent, and mirror prox, all work with polynomial averaging schemes.

6.5 Historical Notes and Further Reading

The derivation of a folk theorem for alternation in matrix games was by Burch et al. [29], after Farina et al. [60] pointed out that the original folk theorem does not apply when using alternation. The general convex-concave case is new, although easily derived from the existing results.

The fact that instantiating OMD with the Euclidean distance seems to perform better than entropy when solving matrix games in practice has been observed in a few different algorithms both first-order methods [33, 68] and regret-minimization algorithms [61]. The fact that OMD with Euclidean distance performs much better after adding alternation has not been observed before.

Results for polynomial averaging schemes were shown by Tammelin et al. [124] and Brown and Sandholm [21] for RM$^+$, in Nemirovski's lecture notes[1] for mirror descent and mirror prox, and for several other primal-dual first-order methods by Gao et al. [68].

[1]`https://www2.isye.gatech.edu/~nemirovs/LMCO_LN2019NoSolutions.pdf`

Chapter 7

Optimism and Fast Convergence of Self Play

7.1 Predictive Online Learning

Suppose that we are in an online learning setting as in Chapter 4: we must repeatedly choose actions $x_t \in X \subset \mathbb{R}^n$ for some convex and compact decision set X, and then we receive (linear) losses $g_t \in \mathbb{R}^n$. But now suppose we receive some additional information about the loss function g_t before we have to make a prediction. In particular, we will suppose that at each time t, we are given some *prediction* $m_t \in [0,1]^n$ of the loss g_t. Formally, we now have the following learning protocol: at each time step $t = 1, \ldots, T$:

1. We are given a prediction $m_t \in [0,1]^n$

2. We must choose a decision $x_t \in X$

3. Afterwards, a loss vector $g_t \in [0,1]^n$ is revealed to us, and we pay the loss $\langle g_t, x_t \rangle$

The question is to what extent we can use the prediction to do better than in the standard online learning setting. It is immediately clear that we have to be careful about what we want. Suppose that the prediction are perfect, i.e. $m_t = g_t$, then we can simply best respond to m_t, i.e. select $x_t = \arg\min_{x \in \Delta^n} \langle m_t, x \rangle$ and we will do as well as the best sequence of decisions in hindsight, and generally have significant *negative* regret against the single best action in hindsight. On the other hand, if m_t turns out to be inaccurate, then best responding to m_t could yield linear regret. Ideally, we would like to guarantee $\sqrt{T}$ regret even when m_t is inaccurate, while still doing "well" when m_t is a reasonable prediction. Next we will show that this is indeed possible, with variations on the OMD and FTRL algorithms introduced in Chapter 4.

7.1.1 Online Mirror Descent with Predictions

First we consider OMD with predictions. OMD with predictions is usually called *optimistic online mirror descent* (OOMD). There are two ways to incorporate the prediction m_t into the OMD algorithm. The first is what we will call single-step OOMD:

$$x_{t+1} = \arg\min_{x \in X} \langle g_t + m_{t+1} - m_t, x \rangle + \frac{1}{\eta} D(x \| x_t).$$

As a base case, let $x_0 = \arg\min_{x \in X} d(x)$ and $m_0 = 0$. Intuitively, we can think of $g_t - m_t$ as "undoing" the previous move in the direction of m_t and instead moving in the direction of g_t. Then, we additionally "optimistically" assume that m_{t+1} is a good prediction, and furthermore move in the direction of m_{t+1}.

We now show that single-step OOMD satisfies a regret bound that lets us get compelling guarantees whether the predictions are accurate or not.

Theorem 15. *Assume that $m_1 = 0$ and d is 1-strongly convex. The regret of single-step OOMD with respect to a sequence of losses $g_1, \dots, g_T$ and predictions $m_1, \dots, m_T$ is bounded by*

$$R_T \leq \frac{D(x\|x_1)}{\eta} + \eta \sum_{t=1}^{T} \|g_t - m_t\|_*^2 - \frac{1}{4\eta} \sum_{t=1}^{T} \|x_{t+1} - x_t\|^2.$$

Proof. By first-order optimality, we have for each $t \in \{1, \dots, T\}$ that

$$\begin{aligned} &\langle m_{t+1} + g_t - m_t + (1/\eta)\nabla d(x_{t+1}) - (1/\eta)\nabla d(x_t), x - x_{t+1} \rangle \geq 0 \\ &\qquad\Leftrightarrow \langle m_{t+1} + g_t - m_t, x_{t+1} - x \rangle \leq \frac{1}{\eta} \langle \nabla d(x_{t+1}) - \nabla d(x_t), x - x_{t+1} \rangle. \end{aligned}$$

Applying the three-point lemma (Lemma 1) we get

$$\langle m_{t+1} + g_t - m_t, x_{t+1} - x \rangle \left(\frac{1}{\eta} \leq D(x\|x_t) - D(x\|x_{t+1}) - D(x_{t+1}\|x_t) \right). \tag{7.1}$$

Summing Eq. (7.1) over $t = 1, \dots, T$ and removing telescoping terms on both sides, we get

$$\sum_{t=1}^{T} \langle g_t, x_{t+1} - x \rangle + \sum_{t=1}^{T} \langle m_{t+1} - m_t, x_{t+1} \rangle + \langle m_1 - m_{T+1}, x \rangle \leq \frac{1}{\eta} \left(D(x\|x_1) - D(x\|x_{T+1}) - \sum_{t=1}^{T} D(x_{t+1}\|x_t) \right). \tag{7.2}$$

Now we simplify the left-hand side of Eq. (7.2).

$$\begin{aligned} &\sum_{t=1}^{T} \langle g_t, x_{t+1} - x \rangle + \sum_{t=1}^{T} \langle m_{t+1} - m_t, x_{t+1} \rangle + \langle m_1 - m_{T+1}, x \rangle \\ =& \sum_{t=1}^{T} \langle g_t, x_{t+1} - x \rangle + \sum_{t=1}^{T} \langle m_{t+1} - m_t, x_{t+1} \rangle \\ =& \sum_{t=1}^{T} \langle g_t, x_t - x \rangle + \sum_{t=1}^{T} \langle g_t - m_t, x_{t+1} - x_t \rangle + \sum_{t=1}^{T} \langle m_{t+1}, x_{t+1} \rangle - \sum_{t=1}^{T} \langle m_t, x_t \rangle \\ =& \sum_{t=1}^{T} \langle g_t, x_t - x \rangle + \sum_{t=1}^{T} \langle g_t - m_t, x_{t+1} - x_t \rangle + \langle m_{T+1}, x_{T+1} \rangle - \langle m_1, x_1 \rangle \\ =& \sum_{t=1}^{T} \langle g_t, x_t - x \rangle + \sum_{t=1}^{T} \langle g_t - m_t, x_{t+1} - x_t \rangle \end{aligned} \tag{7.3}$$

The first step is by noting that we set $m_1 = 0$, and we can assume $m_{T+1} = 0$ without changing the regret up to time T. The second step is by adding and subtracting $\langle g_t, x_t \rangle + \langle m_t, x_t \rangle$ for each t. The third step is by telescoping terms. The fourth step is again by noting that we set $m_1 = 0$ and $m_{T+1} = 0$.

Combining Eq. (7.2) and Eq. (7.3), we get

$$\sum_{t=1}^{T} \langle g_t, x_t - x \rangle \leq \sum_{t=1}^{T} \langle g_t - m_t, x_t - x_{t+1} \rangle + \frac{1}{\eta} \left(D(x\|x_1) - D(x\|x_{T+1}) - \sum_{t=1}^{T} D(x_{t+1}\|x_t) \right) \tag{7.4}$$

Notice that the left-hand side is the regret up to time T. Next we simplify the first term on the right-hand side via the Cauchy-Schwarz and Young inequalities.

$$\begin{aligned} \langle g_t - m_t, x_t - x_{t+1} \rangle &\leq \|g_t - m_t\|_* \|x_t - x_{t+1}\| \\ &\leq \eta \|g_t - m_t\|_*^2 + \frac{1}{4\eta} \|x_t - x_{t+1}\|^2. \end{aligned}$$

Plugging this upper bound into Eq. (7.4) and using $D(x_{t+1}\|x_t) \geq \frac{1}{2}\|x_{t+1}-x_t\|^2$ we get the desired result.

$$\begin{aligned}\sum_{t=1}^T \langle g_t, x_t - x\rangle &\leq \sum_{t=1}^T \left(\eta\|g_t - m_t\|_*^2 + \frac{1}{4\eta}\|x_t - x_{t+1}\|^2 - \frac{1}{2\eta}\|x_{t+1}-x_t\|^2\right) + \frac{1}{\eta}\left(D(x\|x_1) - D(x\|x_{T+1})\right)\\ &\leq \frac{1}{\eta}D(x\|x_1) + \sum_{t=1}^T\left(\eta\|g_t - m_t\|_*^2 - \frac{1}{4\eta}\|x_{t+1}-x_t\|^2\right)\end{aligned}$$

□

Two-step OOMD The second way to incorporate predictions in OMD is the *two-step* OOMD. In Two-step OOMD, we maintain two separate sequences of decisions:

$$\begin{aligned}x_{t+1} &= \arg\min_{x\in X}\langle m_{t+1}, x\rangle + \frac{1}{\eta}D(x\|z_t)\\ z_{t+1} &= \arg\min_{x\in X}\langle g_t, x\rangle + \frac{1}{\eta}D(x\|z_t).\end{aligned}$$

Intuitively, we can think of z_t as the sequence of iterates generated by always moving in the direction of improvement against the losses $g_1, \ldots g_t$, while each x_t is generated by taking one step in the direction of m_t from the previous iterate z_{t-1}. Because the steps in the direction of m_t are never incorporated into the sequence z_t, there is no need to "undo" moves as in single-step OOMD. Two-step OOMD is arguably less attractive than single-step OOMD, because it requires an additional proximal step. Two-step OOMD has the same regret guarantee as single-step OOMD.

The two-step OOMD procedure was the first to be introduced in the literature, and it was historically referred to simply as OOMD. In the rest of the book, when I refer to OOMD, it can be thought of as either the single-step or two-step procedure. For theoretical purposes, there is usually no difference. In practice single-step OOMD may be preferable, since it avoids the need for an additional proximal step.

7.2 Optimism and RVU Bounds

Next we study a particular form of prediction: we will use the *previous* loss as the prediction of the next loss. In particular, this means that we set $m_t = g_{t-1}$. Now, we are effectively saying that our predictions will be good if losses are not changing too rapidly over time. This leads to the notion of *Regret bounded by Variation in Utilities* (RVU):

Definition 3. *An online learning algorithm satisfies the* Regret bounded by Variation in Utilities *(RVU) property with parameters $\alpha > 0, 0 < \beta \leq \gamma$ and a pair of primal-dual norm $\|\cdot\|, \|\cdot\|_*$ if its regret on a sequence of losses $g_1, \ldots, g_T$ is bounded by*

$$R_T \leq \alpha + \beta\sum_{t=1}^T \|g_t - g_{t-1}\|_*^2 - \gamma\sum_{t=1}^T \|x_t - x_{t-1}\|^2.$$

If we instantiate OOMD with $m_t = g_{t-1}$, then Theorem 15 shows that OOMD satisfies the RVU property with parameters $\alpha = \max_{x\in X}(D(x\|x_1)/\eta$, $\beta = \eta$, and $\gamma = 1/(4\eta)$. Note that the sum over $\|x_{t+1}-x_t\|^2$ in Theorem 15 (known as the *path lentgh*) does not include $\|x_1 - x_0\|^2$, but this value is zero, since $m_1 = g_0 = 0$.

7.3 Fast Convergence in Zero-Sum Games

Next we show that the RVU bounds can be used to obtain fast convergence in two-player zero-sum games. In particular, suppose that we have a game $\min_{x\in X}\max_{y\in Y}\langle x, Ay\rangle$ where X, Y are convex and compact, and A has operator norm $\|A\| \leq L$ with respect to the norms $\|\cdot\|_x, \|\cdot\|_y$. Suppose also that we have distance-generating functions d_x, d_y that are each 1-strongly convex with respect to $\|\cdot\|_x$, and $\|\cdot\|_y$.

Before we start studying the repeated game setup, it will be useful to derive a few inequalities that will allow us to relate A to the variation in the dual norm of losses $\|A(y_t - y_{t-1})\|_{x,*}$ and $\| - A^\top(x_t - x_{t-1})\|_{y,*}$. By definition of the operator norm, we have

$$\|A\| = \max_{\|x\|_x=1} \|Ax\|_{y,*} = \max_{\|x\|_x=1} \max_{\|y\|_y=1} \langle Ay, x\rangle,$$

$$\|A\| = \max_{\|x\|_x=1} \|Ax\|_{y,*} = \max_{x} \frac{1}{\|x\|_x} \|Ax\|_{y,*} \geq \frac{1}{\|x'\|_x} \|Ax'\|_{y,*} \ \forall x' \in X, \tag{7.5}$$

$$\|A\| = \max_{\|y\|_y=1} \|A^\top y\|_{x,*} = \max_{y} \frac{1}{\|y\|_y} \|A^\top y\|_{x,*} \geq \frac{1}{\|y'\|_y} \|A^\top y'\|_{x,*} \ \forall y \in Y. \tag{7.6}$$

The repeated game is as follows:

- Initialize $x_0 \in X, y_0 \in Y$ to be some pair of strategies in the relative interior (in matrix games we usually start with the uniform strategy)
- Provide a recommendation $m_t^x = Ay_{t-1}$ to RM_x and $m_t^y = -A^\top x_{t-1}$ to RM_y
- At time t, let x_t be the recommendation from RM_x and y_t be the recommendation from RM_y
- Let RM_x and RM_y observe losses $g_t = Ay_t, \ell_t = -A^\top x_t$ respectively

In this setup, we see that OOMD satisfies the RVU property with parameters $\alpha = (\max_{x\in X} D(x\|x_1)/\eta$, $\beta = \eta$, and $\gamma = 1/(4\eta)$, as described in the previous section.

Theorem 16. *Suppose that $x_1, \ldots, x_T$ and $y_1, \ldots, y_T$ are generated by regret minimizers satisfying the RVU property with parameters $\alpha_x, \beta_x, \gamma_x, \alpha_y, \beta_y, \gamma_y$ such that $\beta_x\|A\|^2 \leq \gamma_y$ and $\beta_y\|A\|^2 \leq \gamma_x$, then we have the following convergence rate for the pair of average strategies $\bar{x} = \frac{1}{T}\sum_{t=1}^T x_t$ and $\bar{y} = \frac{1}{T}\sum_{t=1}^T y_t$:*

$$\xi(\bar{x}, \bar{y}) \leq \frac{\alpha_x + \alpha_y}{T}$$

Proof. We have

$$\begin{aligned}
T\xi(\bar{x}, \bar{y}) &= T\left(\max_y \langle Ay, \bar{x}\rangle - \min_x \langle A\bar{y}, x\rangle\right) \\
&= \max_y \sum_{t=1}^T \langle Ay, x_t\rangle - \min_x \sum_{t=1}^T \langle Ay_t, x\rangle \\
&= \max_y \sum_{t=1}^T \langle Ay, x_t\rangle - \sum_{t=1}^T \langle Ay_t, x_t\rangle + \sum_{t=1}^T \langle Ay_t, x_t\rangle - \min_x \sum_{t=1}^T \langle Ay_t, x\rangle \\
&\leq \alpha_y + \beta_y \sum_{t=1}^T \|A(x_t - x_{t-1})\|_*^2 - \gamma_y \sum_{t=1}^T \|y_t - y_{t-1}\|^2 \\
&\quad \alpha_x + \beta_x \sum_{t=1}^T \|A(y_t - y_{t-1})\|_*^2 - \gamma_x \sum_{t=1}^T \|x_t - x_{t-1}\|^2
\end{aligned} \tag{7.7}$$

The second equality is by expanding the average strategies. The inequality follows by noting that we have the sum of the player regrets, and then applying the RVU bound. Now we can upper bound Eq. (7.7) by using Eqs. (7.5) and (7.6) to get

$$\begin{aligned}
Eq.\ (7.7) &\leq \alpha_y + \beta_y \|A\|^2 \sum_{t=1}^{T} \|x_t - x_{t-1}\|^2 - \gamma_y \sum_{t=1}^{T} \|y_t - y_{t-1}\|^2 \\
&\alpha_x + \beta_x \|A\|^2 \sum_{t=1}^{T} \|y_t - y_{t-1}\|^2 - \gamma_x \sum_{t=1}^{T} \|x_t - x_{t-1}\|^2 \\
&= \alpha_y + \alpha_x + (\beta_y \|A\|^2 - \gamma_x) \sum_{t=1}^{T} \|x_t - x_{t-1}\|_*^2 + (\beta_x \|A\|^2 - \gamma_y) \sum_{t=1}^{T} \|y_t - y_{t-1}\|_*^2 \\
&\leq \alpha_y + \alpha_x
\end{aligned}$$

Dividing everything by T yields the results. □

Corollary 1. *Suppose that $x_1, \ldots, x_T$ and $y_1, \ldots, y_T$ are generated by OOMD with stepsizes $\eta_x \leq 1/(2\|A\|), \eta_y \leq 1/(2\|A\|^2)$ with the previous loss as the prediction, then we have the following convergence rate for the pair of average strategies $\bar{x} = \frac{1}{T}\sum_{t=1}^{T} x_t$ and $\bar{y} = \frac{1}{T}\sum_{t=1}^{T} y_t$:*

$$\xi(\bar{x}, \bar{y}) \leq \frac{\max_{x \in X} D(x\|x_1)}{\eta_x T} + \frac{\max_{y \in Y} D(y\|y_1)}{\eta_y T}$$

7.4 Small Individual Regrets in General-Sum Games

Next we show that the RVU bounds can be used to obtain small individual regrets in general-sum games. This will rely on each algorithm having a *stability* property, meaning that the algorithm's recommendation does not change too much between each time step.

Lemma 4. *The decisions of OOMD are* stable *in the sense that $\|x_{t+1} - x_t\| \leq \eta \|m_{t+1} + g_{t-1} - m_t\|_*$.*
Suppose $m_t = g_{t-1}$, then we have $\|x_{t+1} - x_t\| \leq \eta \|2g_t - g_{t-1}\|_*$.

Proof. Since $D(\cdot\|x)$ is 1-strongly convex for any x, we have that its convex conjugate is 1-Lipschitz with respect to its gradient. The iterates x_t and x_{t+1} are respectively equal to the gradients of the convex conjugate $D^*(\cdot\|x_t)$ at 0 and at $g_t + m_{t+1} - m_t$. Thus, we have $\|x_{t+1} - x_t\| \leq \eta \|g_t + m_{t+1} - m_t\|_*$. □

Consider a general-sum game where we have n players, decision spaces X_i, and each player has a concave utility function $u_i(x)$ which is Lipschitz in the sense that $\|\nabla u_i(x) - \nabla u_i(x')\|_* \leq L_i \sum_{j=1}^{n} \|x_j - x'_j\|$ for some $L_i > 0$. This is satisfied e.g. if u_i is multilinear, as in the case of normal-form games and extensive-form games. The repeated game is as follows:

- Initialize $x_0^i \in X_i$ to be a strategy in the relative interior for each player i (in normal-form games we usually start with the uniform strategy)
- Provide a recommendation $m_t^i = \nabla u_i(x_{t-1})$ to the regret minimizer for each player i
- At time t, let x_t^i be the recommendation for player i and x_t be the collection of recommendations for all players (i.e. the strategy profile at time t).
- Let player i observe the loss $g_t^i = \nabla u_i(x_t)$

As in the previous section, OOMD satisfies the RVU property with parameters $\alpha = (D(x\|x_1)/\eta$, $\beta = \eta$, and $\gamma = 1/(4\eta)$.

Theorem 17. *Suppose that each player's decisions i in a general-sum game are* stable *in the sense that $\|x_t^i - x_{t-1}^i\| \leq \kappa$ for all t, and each player uses a regret minimizer with RVU guarantees $\alpha_i, \beta_i, \gamma_i$. Then each player's regret is bounded as follows*

$$R_T^i \leq \alpha_i + \beta_i T L_i^2 n^2 \kappa^2$$

Proof. First note that from Lipschitzness of the game, we have

$$\sum_{t=1}^{T} \|g_t^i - g_{t-1}^i\|_*^2 \le \sum_{t=1}^{T} nL_i^2 \sum_{j=1}^{n} \|x_{i,t} - x_{i,t-1}\|^2 \le TL_i^2 n^2 \kappa^2$$

Combining this with the RVU property, we have

$$\begin{aligned} R_T^i &\le \alpha_i + \beta_i \sum_{t=1}^{T} \|g_t^i - g_{t-1}^i\|_*^2 - \gamma_i \sum_{t=1}^{T} \|x_t^i - x_{t-1}^i\|^2 \\ &\le \alpha_i + \beta_i T L_i^2 n^2 \kappa^2 \end{aligned}$$

□

Now we immediately get a better than $\sqrt{T}$ regret bound for OOMD by setting the stepsize the right way.

Corollary 2. *Suppose that each player's decisions are generated by OOMD with stepsizes* $\eta_i = \Omega_i^{1/4}/(T^{1/4}L_i^{1/2}n^{1/2})$, *then each player's regret is bounded as follows*

$$R_T^i \le 2\Omega_i^{3/4} T^{1/4} L_i^{1/2} n^{1/2}$$

Proof. Instantiating the regret bound with OOMD gives

$$R_T^i \le \frac{\Omega_i}{\eta} + \eta^3 T L_i^2 n^2 \le \Omega_i^{3/4} T^{1/4} L_i^{1/2} n^{1/2} + \Omega_i^{3/4} T^{1/4} L_i^{1/2} n^{1/2}$$

□

7.5 Historical Notes and Further Reading

The idea of predictive online learning leading to fast convergence in zero-sum games was shown by Rakhlin and Sridharan [110]. The formulation of RVU bounds was given by Syrgkanis et al. [120], where they showed that the bounds can be used to obtain fast convergence in two-player zero-sum games, and improved regret bounds in general-sum games. Earlier, Daskalakis et al. [48] (note that the conference version appeared in 20111) had showed that it is possible to achieve $O(\ln T/T)$ convergence in two-player zero-sum games via self-play with no-regret learning dynamics, but their result relied on a somewhat intricate learning dynamic based on a decentralized implementation of the EGT algorithm for saddle-point problems [99].

The idea of optimism and fast convergence in two-player zero-sum games is also related to earlier works in the first-order methods literature, where some form of *extrapolation* leads to a $O(1/T)$ rate of convergence for convex-concave saddle-point problems. For example, the mirror prox method by Nemirovski [97] achieves this rate, and as pointed out by Rakhlin and Sridharan [110], optimistic OMD in self play can be seen as achieving a similar idea as mirror prox. Moreover, in the case of using the Euclidean DGF in optimistic OMD for solving a two-player zero-sum game, the algorithm is equivalent to an algorithm given by Popov [109], though the $O(1/T)$ rate was not known at the time. Prior to the $O(1/T)$ rate result by Nemirovski [97], Nesterov was, to the best of my knowledge, the first to show that such rates are attainable via first-order methods. Nesterov's approach used what's now known as *Nesterov smoothing* [100], where a smooth approximation to the nonsmooth problem is constructed, and then this approximation is solved via accelerated first-order methods. Though the Nesterov smoothing paper appeared in a journal in 2005 and the Nemirovski paper appeared in 2004, the Nesterov paper predates the Nemirovski paper; it was made available online in 2003. In fact, Nemirovski explicitly credits Nesterov's work as an inspiration in his paper. The inversion of dates is due to the tardiness of the journal publication process. Concurrently with Nemirovski's mirror prox result, Nesterov also developed the *excessive gap technique* (EGT), another method that achieves $O(1/T)$ via first-order updates [99].

Optimism in EFGs was first studied by Farina et al. [61], where they use dilated distance-generating functions (DGFs) such as those we studied in Section 8.5. However, the numerical performance turned out to be worse than that of CFR$^+$ algorithms. Lee et al. [91] showed last-iterate convergence results for optimistic algorithms in two-player zero-sum EFGs that use dilated DGFs, though with the assumption of a unique Nash equilibrium in the case of dilated entropy-based DGFs.

Based on the strong practical performance of CFR$^+$ compared to optimistic methods in EFG solving, it was a natural question whether "optimistic learning" in CFR$^+$ is possible. Farina et al. [63] and Flaspohler et al. [66] concurrently showed how to design predictive variants of RM$^+$. Farina et al. [63] introduced predictive CFR$^+$ which combines CFR and predictive RM$^+$. They show that predictive CFR$^+$ leads to very strong practical performance in many games. Interestingly, they found that non-predictive CFR$^+$ is faster for poker games, whereas predictive CFR$^+$ is *much* faster for various non-poker EFG benchmark games. However, no theoretical improvement over non-predictive CFR$^+$ or RM$^+$ is achieved by these algorithms, in terms of dependence on the number of iterations T when used in self play in two-player zero-sum games. Unlike for OMD, OOMD, and various FTRL variants, it was recently shown that the RM+ algorithm is not stable [64]. This is a key reason why the predictive variant of RM$^+$ does not achieve a $1/T$ convergence rate in zero-sum games, since it means that the previous loss it not always a good prediction of the next loss. Farina et al. [64] also show numerical examples where predictive RM$^+$ converges at a rate of $1/\sqrt{T}$.

Chapter 8

Extensive-Form Games

8.1 Introduction

In this lecture we will cover *extensive-form games* (EFGs). Extensive-form games are a richer game description that explicitly models sequential interaction. EFGs are played on a game tree. Each node in the game tree belongs to some player, whom gets to choose the branch to traverse.

8.2 Perfect-Information EFGs

We start by considering *perfect-information* EFGs. The term perfect information refers to the fact that in these games, every player always knows the exact state of the game. A perfect-information EFG is a game played on a tree, where each internal node belongs to some player. The actions for the player at a given node is the set of branches, and by selecting a branch the game proceeds to the following node. An example is shown in Figure 8.1 on the left. That game has four nodes where players take actions, two belong to player 1 (labelled P1) and two belonging to player 2 (labelled P2). Additionally, the game tree has 6 leaf nodes. At each leaf node, each player receives some payoff. In this particular game, it is a zero-sum game, and the value at a leaf denotes the value that player 1 receives.

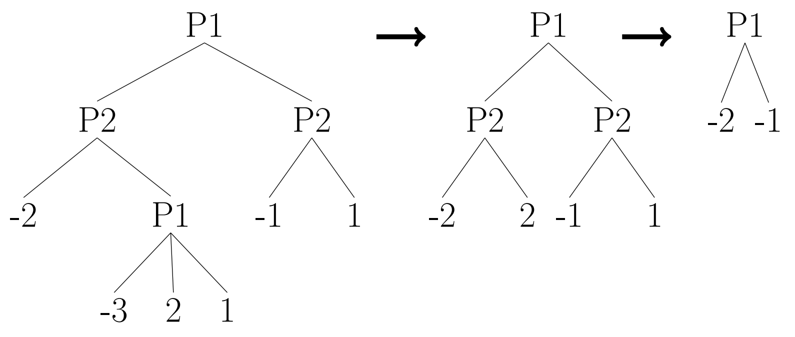


Figure 8.1: A simple perfect-information EFG. Three versions of the game are shown, where each stage corresponds to removing one layer of the game via backward induction.

Perfect-information EFGs are trivially solvable (at least if we are able to traverse the whole game tree at least once). The way to solve them is via *backward induction.* Backward induction works by starting at some bottom decision node of the game tree, which only leads to leaf nodes after each action is taken (such a node always exists). Then, the optimal action for the player at the node is selected, and the node is replaced with the corresponding leaf node. Now we get a new perfect-information EFG with one less internal node. Backward induction then repeats this process until there's no internal nodes left, at which point we have computed a Nash equilibrium. Thus perfect-information EFGs always have pure-strategy Nash equilibria.

While backward induction yields a linear-time algorithm for solving perfect-information games, in practice, many games of interest are way too large to solve with it nonetheless. For example, chess and go both have enormous game trees, with estimates of $\sim 10^{45}$ and $\sim 10^{172}$ nodes respectively.

Next let us see how converting to normal form works. The way converting to normal form works is that for each player, we create an action corresponding to every possible way of assigning an action at every decision point. So, if a player has d decision points with A actions each, then there are A^d actions in the normal form representation of the EFG. This reduction to normal form works for both perfect and imperfect-information games.

Let's consider an instructive example. Here we will model the Cuban Missile Crisis. The USSR has moved a bunch of nuclear weapons to Cuba, and the US has to decide how to respond. If they do nothing, then the USSR wins a political victory, and gets to keep nuclear missiles within firing distance of major US cities. If the US responds, then it could result in a series of escalations that would eventually lead to nuclear war, or the USSR will eventually compromise and remove the missiles.

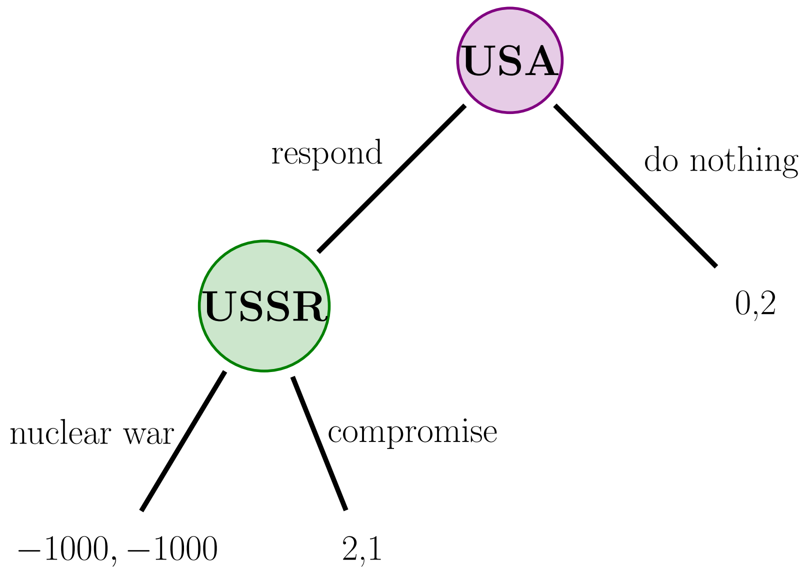


Figure 8.2: A perfect-information EFG modeling the Cuban missile crisis.

If we convert this game to normal form, we get the following game:

		USSR	
		Nuclear war	Compromise
USA	Respond	$-1000, -1000$	2,1
	Do Nothing	0,2	0,2

It is straightforward to see from this representation that the Cuban Missile Crisis game has two PNE: (do nothing, nuclear war) and (respond, compromise). However, the first PNE is in a sense not compelling: what if the USA just responded? The USSR probably would not be willing to follow through on taking the action "nuclear war" since it has such low utility for them as well. This leads to the notion of *subgame-perfect equilibria*, which are equilibria that remain equilibria if we take any *subgame* consisting of picking some node in the tree and starting the game there.

8.3 Imperfect-Information EFGs

Next we study *imperfect-information EFGs*. As the name implies, these are games where players may not have perfect knowledge about the state of the game. From a game-theoretic perspective, this class of games is richer, and will rely more directly on equilibrium concepts for talking about solutions (in contrast to perfect-information EFGs, where solutions are straightforwardly obtained from backward induction). An example is shown in Figure 8.3.

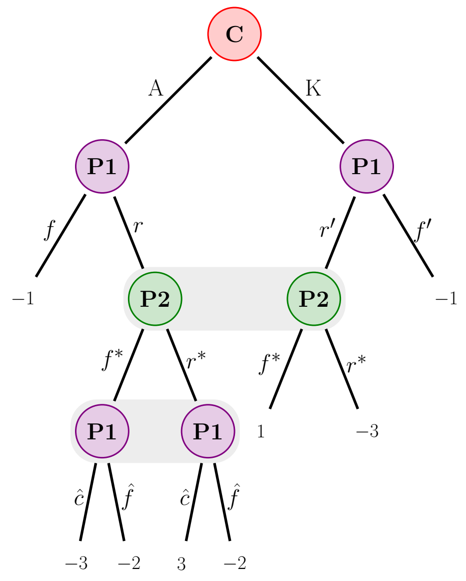


Figure 8.3: A (rather weird) poker game where P1 is dealt Ace or King with equal probability. "r," "f," and "c" stands for raise, fold, and check respectively. Leaf values denote P1 payoffs. The shaded area denotes an information set: P2 does not know which of these nodes they are at, and must thus use the same strategy in both. Note that in the case where they are dealt an ace, P1 does not observe the action taken by P2.

An EFG has the following:

- Information sets: for each player, the nodes belonging to that player are partitioned into *information sets* $I \in \mathcal{I}_i$. information sets represent imperfect information: a player does not know which node in an information set they are at, and thus they muse utilize the same strategy at each node in that information set. In Figure 8.3 P2 has only 1 information set, which contains both their nodes, whereas P1 has four information sets, each one a singleton node. For player i we will also let $\mathcal{J}_i$ be an index set of information sets with generic element j.
- Each information set I with index j has a set of actions that the corresponding player may take, which is denoted by A_j.
- Leaf nodes Z: the set of terminal states. Player i gains utility $u_i(z)$ if leaf node z is reached. Z is the set of all leaf nodes.
- Chance nodes where Chance or Nature moves with a fixed probability distribution. In Figure 8.3 chance deals A or K with equal probability.

We will assume throughout that the game has *perfect recall*, which means that no player ever forgets something they knew in the past. More formally, it means that for every information set $I \in \mathcal{I}_i$, there is a single last information-set action pair I', a' belonging to i that was the last information set and action taken by that player for every node in I.

The last action taken by player i before reaching an information set with index j is denoted p_j. This is well-defined due to perfect recall.

We spent a lot of time learning how one may compute a Nash equilibrium in a two-player zero-sum game by finding a saddle point of a min-max problem over convex compact polytopes. This model looked as follows (we also learned how to handle convex-concave objectives, here we restrict our attention to bilinear saddle-point problems)

$$\min_{x \in X} \max_{y \in Y} \langle x, Ay \rangle. \tag{8.1}$$

Now we would like to find a way to represent EFG zero-sum Nash equilibrium this way. This turns out to be possible, and the key is to find the right way to represent strategies such that we get a bilinear objective. The next section will describe this representation.

First, let us see why the most natural formulation of the strategy spaces won't work. The natural formulation would be to have a player specify a probability distribution over actions at each of their information sets. Let σ be a strategy profile, where σ_a is the probability of taking action a (from now on we assume that every action is distinct so that for any a there is only one corresponding I where the action can be played). The expected value over leaf nodes is

$$\sum_{z \in Z} u_2(z)\mathbb{P}(z|\sigma)$$

The problem with this formulation is that if a player has more than one action on the path to any leaf, then the probability $\mathbb{P}(z|\sigma)$ of reaching z is non-convex in that player's own strategy, since we have to multiply each of the probabilities belonging to that player on the path to z. Thus we cannot get the bilinear form in (8.1).

8.4 Sequence Form

In this section we will describe how we can derive a bilinear representation X of the strategy space for player 1. Everything is analogous for Y.

In order to get a bilinear formulation of the expected value we do not write our strategy in terms of the probability σ_a of playing an action a. Instead, we associate to each information-set-action pair I, a a variable x_a denoting the probability of playing the *sequence* of actions belonging to player 1 on the path to I, including the probability of a at I. For example, in the poker game in Figure 8.3, there would be a variable $x_{\hat{c}}$ denoting the product of probabilities player 1 puts on playing actions r and then $\hat{c}$. To be concrete, say that we have a behavioral strategy σ^1 for player 1, then the corresponding sequence-form probability on the action $\hat{c}$ would be $x_{\hat{c}} = \sigma^1_r \cdot \sigma^1_{\hat{c}}$. Similarly there would be a variable $x_{\hat{f}} = \sigma^1_r \cdot \sigma^1_{\hat{f}}$ denoting the product of probabilities on r and $\hat{f}$. Clearly, for this to define a valid strategy we must have $x_{\hat{c}} + x_{\hat{f}} = x_r$.

More generally, X is defined as the set of all $x \in \mathbb{R}^n, x \geq 0$ such that

$$x_{p_j} = \sum_{a \in A_j} x_a, \forall j \in \mathcal{J}_1, \tag{8.2}$$

where $n = \sum_{I \in \mathcal{I}_i} |A|$, and $p(I)$ is the parent sequence leading to I.

One way to visually think of the set of sequence-form strategies is given in Figure 8.4. This representation

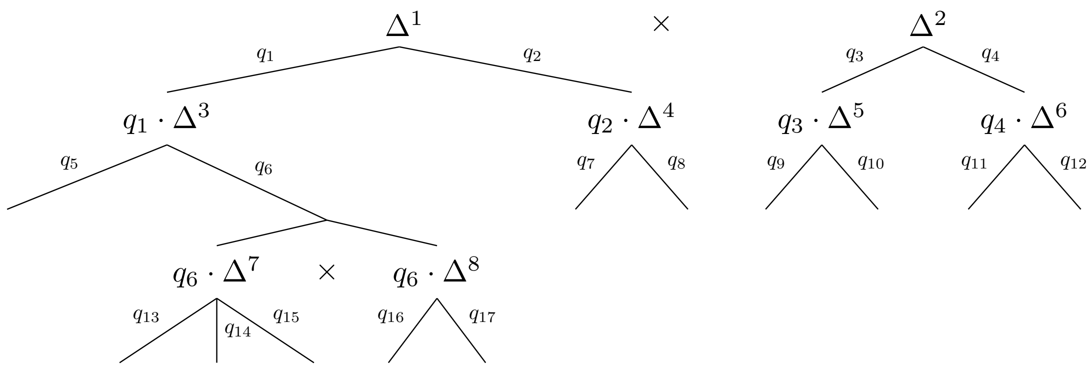


Figure 8.4: An example treeplex constructed from 9 simplices.

is called a *treeplex*. Each information set is represented as a simplex, which is scaled by the parent sequence leading to that information set (by perfect recall there is a unique parent sequence). After taking a particular action it is possible that a player may arrive at several next possible simplexes depending on what the other player or nature does. This is represented by the $\times$ symbol.

It's important to understand that the sequence form specifies probabilities on sequences of actions *for a single player*. Thus they are not the same as paths in the game tree; indeed, the sequence r^* for player 2 appears in two separate paths of the game tree, as player 2 has two nodes in the corresponding information set.

Say we have a set of probability distributions over actions at each information set, with σ_a denoting the probability of playing action a. We may construct a corresponding sequence-form strategy by applying the following equation in top-down fashion (so that x_{p_j} is always assigned before x_a):

$$x_a = x_{p_j}\sigma_a, \forall j \in \mathcal{J}, a \in A_j. \tag{8.3}$$

The payoff matrix A associated with the sequence-form setup is a sparse matrix, with each row corresponding to a sequence of the x player and each column corresponding to a sequence of the y player. Each leaf has a cell in A at the pair of sequences that are last visited by each player before reaching that leaf, and the value in the cell is the payoff to the maximizing player in the bilinear min-max formulation. Cells corresponding to pairs of sequences that are never the last pair of sequences visited before a leaf have a zero.

With this setup we now have an algorithm for computing a Nash equilibrium in a zero-sum EFG: Run online mirror descent (OMD) for each player, using either of our folk-theorem setups from the previous chapter. However, this has one issue, recall the update for OMD (also known as a *prox mapping*):

$$x_{t+1} = \operatorname*{argmin}_{x\in X} \langle \gamma g_t, x\rangle + D(x\|x_t),$$

where $D(x\|x_t) = d(x) - d(x_t) - \langle \nabla d(x_t), x - x_t\rangle$ is the Bregman divergence from x_t to x. In order to run OMD, we need to be able to compute this prox mapping. The question of whether the prox mapping is easy to compute is easily answered when X is a simplex, where updates for the entropy DGF are closed-form, and updates for the Euclidean DGF can be computed in $n \log n$ time, where n is the number of actions. For treeplexes this question becomes more complicated.

In principle we could use the standard Euclidean distance for d. In that case the update can be rewritten as

$$x_{t+1} = \operatorname*{argmin}_{x\in X} \|x - (x_t - \gamma g_t)\|_2^2,$$

which means that the update requires us to project onto a treeplex. This can be done in $n \cdot d \cdot \log n$ time, where n is the number of sequences and d is the depth of the decision space of the player. While this is acceptable, it turns out there are smarter ways to compute these updates which take linear time in n.

8.5 Dilated Distance-Generating Functions

We will see two ways to construct regret minimizers for treeplexes. The first is based on choosing an appropriate distance-generating function (DGF) for the treeplex, such that prox mappings are easy to compute. To that end, we now introduce what are called *dilated DGFs*. In dilated DGFs we assume that we have a DGF d_j for each information set $j \in \mathcal{J}$. For the polytope X we construct the DGF

$$d(x) = \sum_{j\in\mathcal{J}_1} \beta_j x_{p_j} d_j\left(\frac{x^j}{x_{p_j}}\right),$$

where $\beta_j > 0$ is the weight on information set j.

Dilated DGFs have the nice property that the proximal update can be computed recursively as long as we know how to compute the simplex update for each j. Let x^j, g_t^j etc denote the slice of a given vector corresponding to sequences belonging to information set j. The update is

$$\begin{aligned}
&\operatorname*{argmin}_{x\in X}\langle g_t, x\rangle + D(x\|x_t)\\
=&\operatorname*{argmin}_{x\in X}\langle g_t, x\rangle + d(x) - d(x_t) - \langle \nabla d(x_t), x - x_t\rangle\\
=&\operatorname*{argmin}_{x\in X}\langle g_t - \nabla d(x_t), x\rangle + d(x)\\
=&\operatorname*{argmin}_{x\in X}\sum_{j\in\mathcal{J}}\left(\langle g_t^j - \nabla d(x_t)^j, x^j\rangle + \beta_j x_{p_j} d_j(x^j/x_{p_j})\right)\\
=&\operatorname*{argmin}_{x\in X}\sum_{j\in\mathcal{J}} x_{p_j}\left(\langle g_t^j - \nabla d(x_t)^j, x^j/x_{p_j}\rangle + \beta_j d_j(x^j/x_{p_j})\right)
\end{aligned}$$

Now we may consider some information set j with no descendant information sets. Since x_{p_j} is on the outside of the parentheses, we can compute the update at j as if it were a simplex update, and the value at the information set can be added to the coefficient on x_{p_j}. That logic can then be applied recursively. Thus we can traverse the treeplex in bottom-up order, and at each information set we can compute the value for x^j_{t+1} in however long it takes to compute an update for a simplex with DGF d_j.

If we use the entropy DGF for each $j \in \mathcal{J}$ and set the weight $\beta_j = 2 + \max_{a\in A_j}\sum_{j'\in\mathcal{C}_j^a} 2\beta_{j'}$, then we get a DGF for X that is strongly convex modulus $\frac{1}{M}$ where $M = \max_{x\in X}\|x\|_1$. If we scale this DGF by M we get that it is strongly convex modulus 1. If we instantiate the mirror prox algorithm with this DGF for X and Y we get an algorithm that converges at a rate of

$$O\left(\frac{\max_{ij} A_{ij}\max_{I\in\mathcal{I}}\log(|A_I|)\sqrt{M_x^2 2^d + M_y^2 2^d}}{T}\right),$$

where M_x, M_y are the maximum ℓ_1 norms on X and Y, and d is an upper bound on the depth of both treeplexes. This gives the fastest theoretical rate of convergence among gradient-based methods. However, this only works for OMD. All our other algorithms (RM, RM$^+$) were for simplex domains exclusively. Next we derive a way to use these locally at each information set. It turns out that faster practical performance can be obtained this way.

8.6 Counterfactual Regret Minimization

The framework we will cover is the *counterfactual regret minimization* (CFR) framework for constructing regret minimizers for EFGs.

CFR is based on deriving an upper bound on regret, which allows decomposition into local regret minimization at each information set.

We are interested in minimizing the standard regret notion over the sequence form:

$$R_T = \sum_{t=1}^T\langle g_t, x_t\rangle - \min_{x\in X}\sum_{t=1}^T\langle g_t, x\rangle.$$

To get the decomposition, we will define a local notion of regret which is defined with respect to behavioral strategies $\sigma \in \times_j \Delta^j =: \Sigma$ (here we just derive the decomposition for a single player, say player 1. Everything is analogous for player 2).

We saw in the previous lecture note that it is always possible to go from behavioral form to sequence form using the following recurrence, where assignment is performed in top-down order.

$$x_a = x_{p_j}\sigma_a, \forall j \in \mathcal{J}, a \in A_j. \tag{8.4}$$

It is also possible to go the other direction (though this direction is not a unique mapping, as one has a choice of how to assign behavioral probabilities at information sets j such that $x_{p_j} = 0$). These procedures produce payoff-equivalent strategies for EFGs.

For a behavioral strategy vector σ (or loss vector g_t) we say that σ^j is the slice of σ corresponding to information set j. $\sigma^{j\downarrow}$ is the slice corresponding to j, *and every information set below* j. Similarly, $\Sigma^{j\downarrow}$ is the set of all behavioral strategy assignments for the subset of simplexes that are in the tree of simplexes rooted at j.

We let $\mathcal{C}_{j,a}$ be the set of next information sets belonging to player 1 that can be reached from j when taking action a. In other words, the set of information sets whose parent sequence is a.

Now, let the *value function* at time t for an information set j belonging to player 1 be defined as

$$V_t^j(\sigma) = \langle g_t^j, \sigma^j \rangle + \sum_{a \in A_j} \sum_{j' \in \mathcal{C}_{j,a}} \sigma_a V_t^{j'}(\sigma^{j'\downarrow}).$$

where $\sigma \in \Sigma^{j\downarrow}$. Intuitively, this value function represents the value that player 1 derives from information set j, assuming that i played to reach it, i.e. if we counterfactually set $x_{p_j} = 1$.

The *subtree regret* at a given information set j is

$$R_T^{j\downarrow} = \sum_{t=1}^T V_t^j(\sigma_t^{j\downarrow}) - \min_{\sigma \in \Sigma^{j\downarrow}} \sum_{t=1}^T V_t^j(\sigma),$$

Note that this regret is with respect to the behavioral form.

The local loss that we will eventually minimize is defined as

$$\hat{g}_{t,a}^j = g_{t,a} + \sum_{j' \in \mathcal{C}_{j,a}} V_t^{j'}(\sigma_t^{j'\downarrow}).$$

Note that for each j, the loss depends linearly on σ^j; σ^j does not affect information sets below j, since we use σ_t in the value function for child information sets j'.

Now we show that the subtree regret decomposes in terms of local losses and subtree regrets.

Theorem 18. *For any $j \in \mathcal{J}$, the subtree regret at time T satisfies*

$$R_T^{j\downarrow} = \sum_{t=1}^T \langle \hat{g}_t^j, \sigma_t^j \rangle - \min_{\sigma \in \Delta^j} \left(\sum_{t=1}^T \langle \hat{g}_t^j, \sigma \rangle - \sum_{a \in A_j, j' \in \mathcal{C}_{j,a}} \sigma_a R_T^{j'\downarrow} \right).$$

Proof. Using the definition of subtree regret we get

$$\begin{aligned} R_t^{j\downarrow} &= \sum_{t=1}^T V_t^j(\sigma_t^{j\downarrow}) - \min_{\sigma \in \Sigma^{j\downarrow}} \left(\sum_{t=1}^T \langle g_t^j, \sigma^j \rangle + \sum_{a \in A_j, j' \in \mathcal{C}_{j,a}} \sigma_a V_t^{j'}(\sigma^{j'\downarrow}) \right) && \text{by expanding } V_t^j(\sigma^{j\downarrow}) \\ &= \sum_{t=1}^T V_t^j(\sigma_t^{j\downarrow}) - \min_{\sigma \in \Delta^j} \left(\sum_{t=1}^T \langle g_t^j, \sigma \rangle + \sum_{a \in A_j, j' \in \mathcal{C}_{j,a}} \sigma_a \min_{\hat{\sigma} \in \Sigma^{j'\downarrow}} V_t^{j'}(\hat{\sigma}^{j'\downarrow}) \right) && \text{by sequential min} \\ &= \sum_{t=1}^T V_t^j(\sigma_t^{j\downarrow}) - \min_{\sigma \in \Delta^j} \left(\sum_{t=1}^T \langle \hat{g}_t^j, \sigma \rangle - \sum_{a \in A_j, j' \in \mathcal{C}_{j,a}} \sigma_a R_T^{j'\downarrow} \right) && \text{by definition of } \hat{g}_t \text{ and } R_T^{j'\downarrow}. \end{aligned}$$

The theorem follows, since $V_t^j(\sigma_t^{j\downarrow}) = \langle \hat{g}_t^j, \sigma_t^j \rangle$. □

The local regret that we will be minimizing is the following

$$\hat{R}_T^j := \sum_{t=1}^T \langle \hat{g}_t^j, \sigma_t^j \rangle - \min_{\sigma \in \Delta^j} \sum_{t=1}^T \langle \hat{g}_t^j, \sigma \rangle.$$

Note that this regret is in the behavioral form, and it corresponds exactly to the regret associated to locally minimizing $\hat{g}_t^j$ at each simplex j.

The CFR framework is based on the following theorem, which says that the sequence-form regret can be upper-bounded by the behavioral-form local regrets.

Theorem 19. *The regret at time T satisfies*

$$R_T = R_T^{root\downarrow} \le \max_{x \in X} \sum_{j \in \mathcal{J}} x_{p_j} \hat{R}_T^j,$$

where root is the root information set.

Proof. For the equality, consider the regret R_T over the sequence form polytope X. Since each sequence-form strategy has a payoff equivalent behavioral strategy in Σ and vice versa, we get that the regret R_T is equal to $R_T^{root\downarrow}$ for the root information set *root* (we may assume WLOG. that there is a root information set since if not then we can add a dummy root information set with a single action).

By Theorem 18 we have for any $j \in \mathcal{J}$

$$\begin{aligned} R_T^{j\downarrow} &= \sum_{t=1}^T \langle \hat{g}_t^j, \sigma_t^j \rangle - \min_{\sigma \in \Delta^j} \left(\sum_{t=1}^T \langle \hat{g}_t^j, \sigma \rangle - \sum_{a \in A_j, j' \in \mathcal{C}_{j,a}} \sigma_a R_T^{j'\downarrow} \right) \\ &\le \sum_{t=1}^T \langle \hat{g}_t^j, \sigma_t^j \rangle - \min_{\sigma \in \Delta^j} \sum_{t=1}^T \langle \hat{g}_t^j, \sigma \rangle + \max_{\sigma \in \Delta^j} \sum_{a \in A_j, j' \in \mathcal{C}_{j,a}} \sigma_a R_T^{j'\downarrow}, \end{aligned} \tag{8.5}$$

where the inequality is by the fact that independently minimizing the terms $\sum_{t=1}^T \langle \hat{g}_t^j, \sigma \rangle$ and $-\sum_{a \in A_j, j' \in \mathcal{C}_{j,a}} \sigma_a R_T^{j'\downarrow}$ is smaller than jointly minimizing them.

Now we may apply (8.5) recursively in top-down fashion starting at *root* to get the theorem. □

A direct corollary of Theorem 19 is that if the counterfactual regret at each information set grows sublinearly then overall regret grows sublinearly. This is the foundation of the *counterfactual regret minimization* (CFR) framework for minimizing regret over treeplexes. The CFR framework can succinctly be described as

1. Instantiate a local regret minimizer for each information set simplex Δ^j.

2. At iteration t, for each $j \in \mathcal{J}$, feed the local regret minimizer the counterfactual regret $\hat{g}_t^j$.

3. Generate x_{t+1} as follows: ask for the next recommendation from each local regret minimizer. This yields a set of simplex strategies, one for each information set. Construct x_{t+1} via (8.4).

Thus we get an algorithm for minimizing regret on treeplexes based on minimizing counterfactual regrets. In order to construct an algorithm for computing a Nash equilibrium based on a CFR setup, we may invoke the folk theorems from the previous lectures using the sequence-form strategies generated by CFR. Doing this yields an algorithm that converges to a Nash equilibrium of an EFG at a rate on the order of $O\left(\frac{1}{\sqrt{T}}\right)$

While CFR is technically a framework for constructing local regret minimizers, the term "CFR" is often overloaded to mean the algorithm that comes from using the folk theorem with uniform averages, and using regret matching as the local regret minimizer at each information set. CFR$^+$ is the algorithm resulting from using the alternation setup, taking linear averages of strategies, and using RM$^+$ as the local regret minimizer at each information set.

We now show pseudocode for implementing the CFR algorithm with the RM$^+$ regret minimizer. In order to compute Nash equilibria with this method one would use CFR as the regret minimizer in one of the folk-theorem setups from the previous lecture.

Algorithm 1: CFR(RM$^+$)($\mathcal{J}, X$)

```
Data: $\mathcal{J}$ set of infosets
      $X$ sequence-form strategy space
function Setup()
    $\mathbf{Q} \leftarrow$ 0-initialized vector over sequences
    $t \leftarrow 1$
function NextStrategy()
    $x \leftarrow 0 \in \mathbb{R}^{|X|}$
    $x_\emptyset \leftarrow 1$
    for $j \in \mathcal{J}$ in top-down order do
        $s \leftarrow \sum_{a \in A_j} \mathbf{Q}_a$
        if $s = 0$ then
            for $a \in A_j$ do
                $x_a \leftarrow x_{p_j} / |A_j|$
        else
            for $a \in A_j$ do
                $x_a \leftarrow x_{p_j} \times \mathbf{Q}_a / s$
    return $x$
function ObserveLoss($g_t \in \mathbb{R}^{|X|}$)
    for $j \in \mathcal{J}$ in bottom-up order do
        $s \leftarrow \sum_{a \in A_j} \mathbf{Q}_a$
        $v \leftarrow 0$    // the value of information set $j$
        if $s = 0$ then
            $v \leftarrow \sum_{a \in A_j} g_{t,a} / |A_j|$
        else
            $v \leftarrow \sum_{a \in A_j} \langle g_{t,a}, \mathbf{Q}_a / s \rangle$
        $g_{t,p_j} \leftarrow g_{t,p_j} + v$    // construct local loss $\hat{g}_t$
        for $a \in A_j$ do
            $\mathbf{Q}_a \leftarrow [\mathbf{Q}_a + (v - g_{t,a})]^+$    // $g_{t,a} = \hat{g}_{t,a}$ since all $j' \in \mathcal{C}_{j,a}$ were already traversed
    $t \leftarrow t + 1$
```

NextStrategy simply implements the top-down recursion (8.4) while computing the update corresponding to RM$^+$ at each j. ObserveLoss uses bottom-up recursion to keep track of the regret-like sequence Q_a, which is based on $\hat{g}_{t,a}$ in CFR.

A technical note here is that we assume that there is some dummy sequence $\emptyset$ at the root of the treeplex with no corresponding j (this corresponds to a single-action dummy information set at the root, but leaving out that dummy information set in the index set $\mathcal{J}$). This makes code much cleaner because there is no need to worry about the special case where a given j has no parent sequence, at the low cost of increasing the length of the sequence-form vectors by 1.

8.7 Numerical Comparison of CFR methods and OMD-like methods

Figure 8.5 shows the performance of three different variations of CFR, as well as the *excessive gap technique* (EGT), a first-order method that converges at a rate of $O(1/T)$ using the dilated entropy DGF from last lecture (EGT is equivalent to the mirror prox algorithm that was shown previously, in terms of theoretical convergence rate). The plots show performance on four EFGs: *Leduc poker*, a simplified poker game that is standard in EFG solving (three different deck sizes are shown), and *search*, a game played on a graph where an attacker attempts to reach a particular set of nodes, and the defender tries to capture them (full descriptions can be found in Kroer et al. [88]).

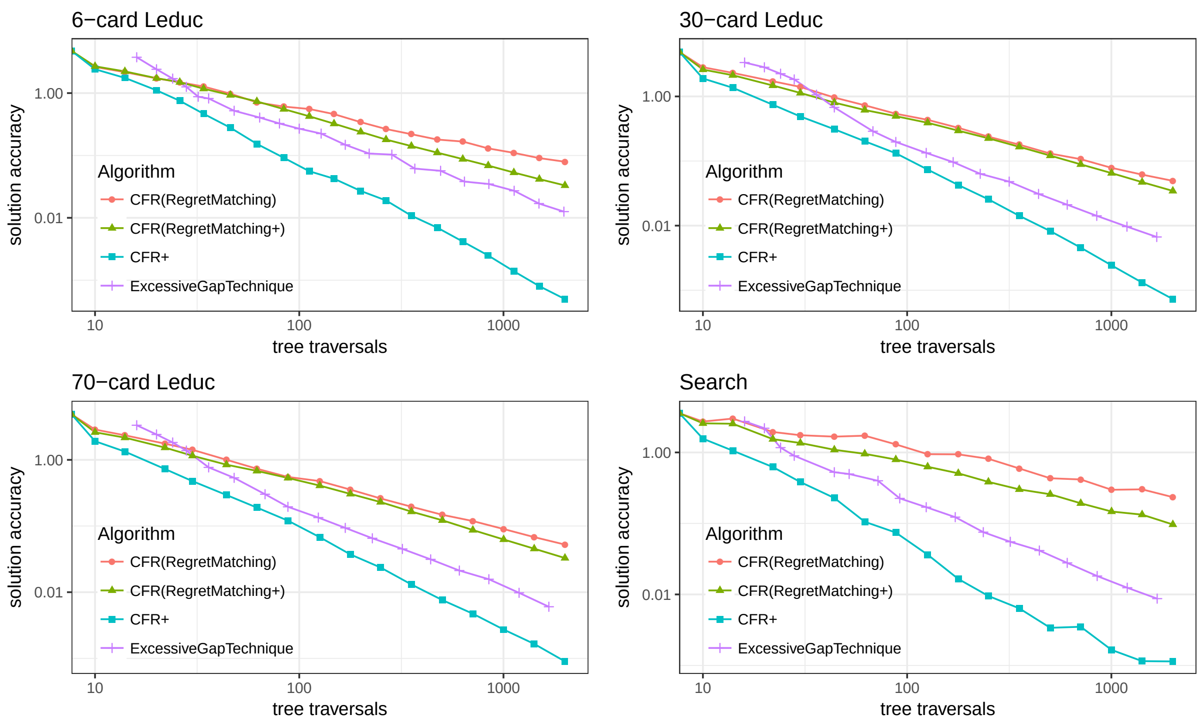


Figure 8.5: Solution accuracy as a function of the number of tree traversals in three different variants of Leduc hold'em and a pursuit evasion game. Results are shown for CFR with regret matching, CFR with regret matching$^+$, CFR$^+$, and EGT. Both axes are shown on a log scale.

8.8 Stochastic Gradient Estimates

So far we have operated under the assumption that we can easily compute the matrix-vector product $g_t = Ay_t$, where A is the payoff matrix of the EFG that we are trying to solve. While g_t can indeed be computed in time linear in the size of the game tree, we may be in a case where the game tree is so large that even one traversal is too much. In that case, we are interested in developing methods that can work with some stochastic gradient estimator $\tilde{g}_t$ of the gradient. Typically, one would consider unbiased gradient estimators, i.e. $\mathbb{E}[\tilde{g}_t] = g_t$.

Assuming that we have a gradient estimator $\tilde{g}_t$ for each time t, a natural approach for attempting to compute a solution would be to apply our previous approach of running a regret minimizer for each player and using the folk theorem, but now using $\tilde{g}_t$ at each iteration, rather than g_t. If our unbiased gradient estimator $\tilde{g}_t$ is reasonably accurate then we might expect that this approach should still yield an algorithm for computing a Nash equilibrium. This turns out to be the case.

Theorem 20. *Assume that each player uses a bounded unbiased gradient estimator for their loss at each iteration. Then for all $p \in (0,1)$, with probability at least $1-2p$*

$$\xi(\bar{x},\bar{y}) \le \frac{\tilde{R}_T^1 + \tilde{R}_T^2}{T} + \left(2\Delta + \tilde{M}_1 + \tilde{M}_2\right)\sqrt{\frac{2}{T}\log\frac{1}{p}},$$

where $\tilde{R}_T^i$ is the regret incurred under the losses $\tilde{g}_t^i$ for player i, $\Delta = \max_{z,z'\in Z} u_2(z) - u_2(z')$ is the payoff range *of the game, and $\tilde{M}_1 \ge \max_{x,x'\in X}\langle \tilde{g}_t, x - x'\rangle, \forall \tilde{g}_t$ is a bound on the "size" of the gradient estimate, with M_2 defined analogously.*

We will not show the proof here, but it follows from introducing the discrete-time stochastic process

$$d_t := g_t(x_t - x)) - \tilde{g}_t(x_t - x),$$

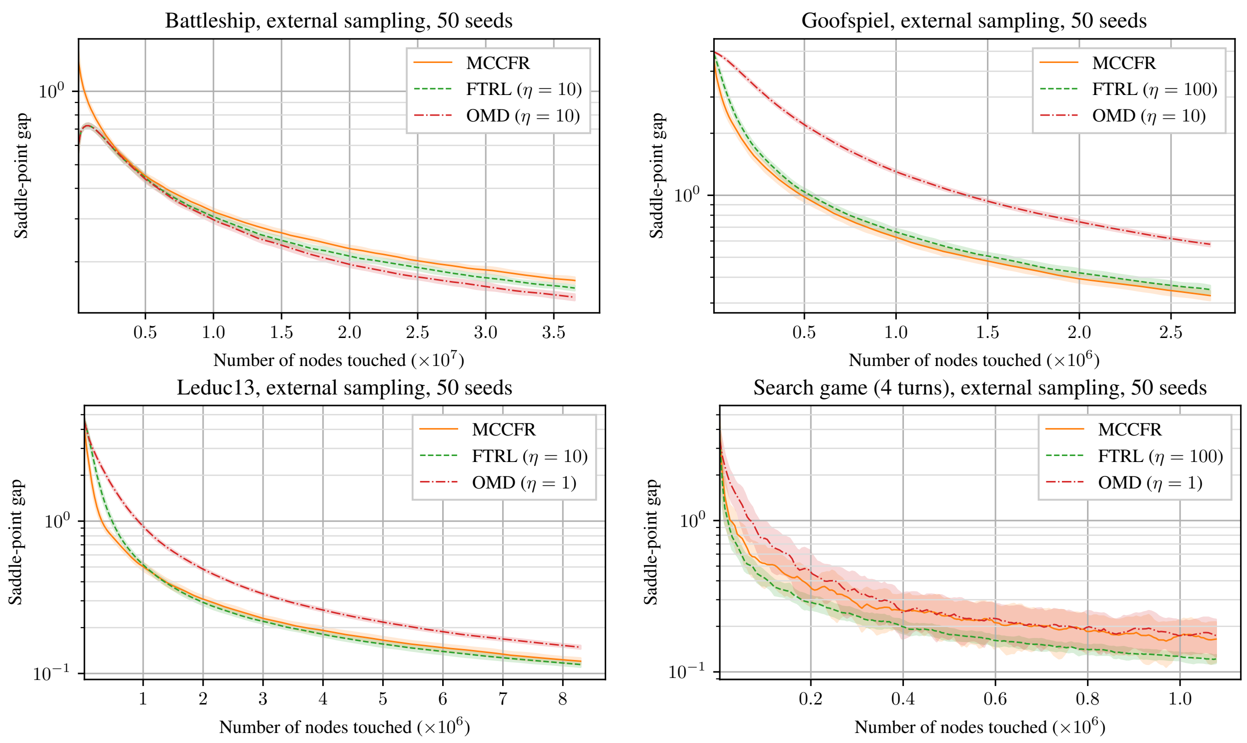


Figure 8.6: Performance of CFR, FTRL, and OMD when using the external sampling gradient estimator.

observing that it is a martingale difference sequence, and applying the Azuma-Hoeffding concentration inequality.

With Theorem 20 in hand, we just need a good way to construct gradient estimates $\tilde{g}_t \approx Ay_t$. Generally, one can construct a wide array of gradient estimators by using the fact that Ay_t can be computed by traversing the EFG game tree: at each leaf node z in the tree, we add $-u_1(z)y_a$ to $g_{t,a'}$, where a is the last sequence taken by the y player, and a' is the last sequence taken by the x player. To construct an estimator, we may choose to sample actions at some subset of nodes in the game tree, and then only traverse the sampled branches, while taking care to normalize the eventual payoff so that we maintain an unbiased estimator. One of the most successful estimators construct this way is the *external sampling* estimator. In external sampling when computing the gradient Ay_t, we sample a single action at every node belonging to the y player or chance, while traversing all branches at nodes belonging to the x player.

Figure 8.6 shows the performance when using external sampling in CFR (CFR with sampling is usually called Monte-Carlo CFR or MCCFR), FTRL, and OMD. Performance is shown on Leduc with a 13-card deck, Goofspiel (another card game), search, and battleship. In the deterministic case we saw that CFR$^+$ was much faster than the theoretically-superior EGT algorithm (and OMD/FTRL would perform much worse than EGT). Here we see that in the stochastic case it varies which algorithm is better.

8.9 Historical Notes

The sequence form was discovered in the USSR in the 60s [112] and later rediscovered independently [134, 82]. Dilated DGFs for EFGs were introduced by Hoda et al. [76] where they proved that any such DGF constructed from simplex DGFs which are strongly convex must also be strongly convex. Kroer et al. [88] showed the strong convexity modulus of the dilated entropy DGF shown here. An explicit bound for the dilated Euclidean DGF can be found in Farina et al. [61], which also explores regret minimization algorithms with dilated DGFs in depth.

CFR-based algorithms were used as the algorithm for computing Nash equilibrium in all the recent milestones where AIs beat human players at various poker games [16, 95, 20, 22].

CFR was introduced by Zinkevich et al. [139]. Many later variations have been developed, for example the stochastic method MCCFR [90], and variations on which local regret minimizer to use in order to speed up practical performance [124, 21]. The proof of CFR given here is a simplified version of the more general theorem developed in [60]. The plots on CFR vs EGT are from Kroer et al. [88].

The bound on error from using a stochastic method in Theorem 20 is from Farina et al. [62], and the plots on stochastic methods are from that same paper. External sampling and several other EFG gradient estimators were introduced by Lanctot et al. [90].

Chapter 9

Search in Extensive-Form Games

9.1 Introduction

We previously saw how to compute a Nash equilibrium of a two-player zero-sum extensive-form game (EFG) by using dilated distance-generating functions or the CFR framework. We also saw that even if computing gradients $g_t = Ay_t$ is too time-consuming we can still run algorithms using gradient estimates constructed via sampling. However, for some real-world games such as two-player no-limit Texas hold'em, this is still not enough. The game tree in this game has roughly 10^{170} nodes, and the strategy space is much too large to even write down strategy iterates. Faced with this situation, we need to make even coarser approximations to our problem.

One major innovation for solving large-scale poker games was the use of *real-time search*. Traditionally, poker AIs were created by precomputing an approximate Nash equilibrium for some extremely coarsened representation of the full game using e.g. CFR$^+$. Then, that offline strategy was simply employed during play. In real-time search, the precomputed Nash equilibrium approximation is refined in real time for subgames encountered during live play. This allows the AI to reason in much more detail, especially towards the end of the game, where the encountered subtree is manageable in size. In order to understand how search works in EFGs, we will first show how it works in the simpler setting of perfect-information EFGs, where there are no information sets, and so players know exactly which node they are currently at. Search in perfect-information EFGs has historically been extremely successful, it was used in AI milestones on Chess and Go.

9.2 Backward Induction

Perfect-information EFGs (meaning that all information sets consist of a single node) can be solved via *backward induction*. Since the game is played on a tree, and a player always knows exactly where in the tree they are, we can reason about the optimal strategy at a given node purely by considering the subgame rooted at the node. We do not need to worry about what happens in any other parts of the game tree. Backward induction exploits this fact by recursively solving every subgame. It starts at leaf nodes, and then at any internal node, the algorithm pick the action that leads to the best subgame for the player acting at the node (breaking ties arbitrarily). An example is shown in Figure 9.1.

9.3 Search in Games

In search, we search for a solution in real-time during play. Say that we are playing chess, which is a perfect-information EFG. Say that some set of moves already happened, resulting in the board state shown below:

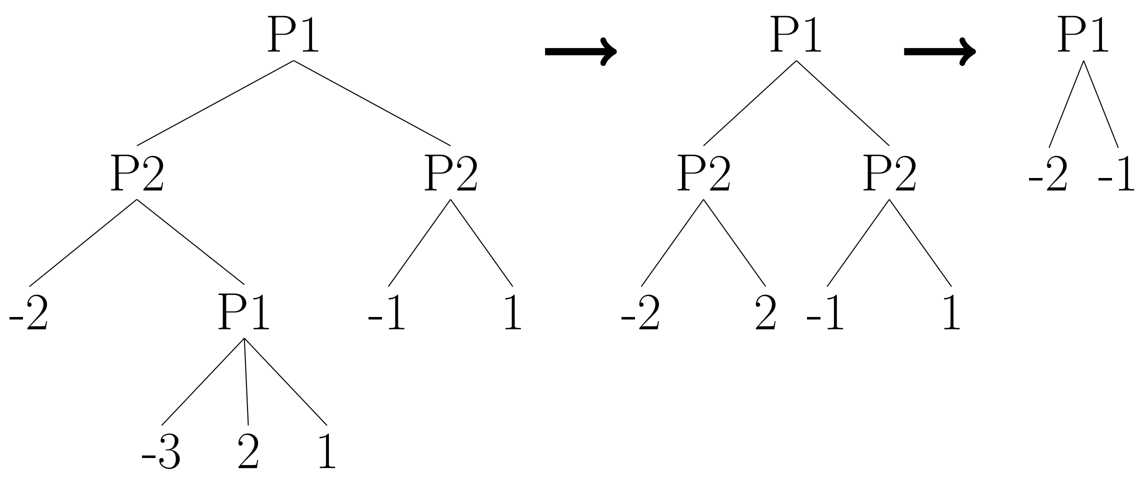
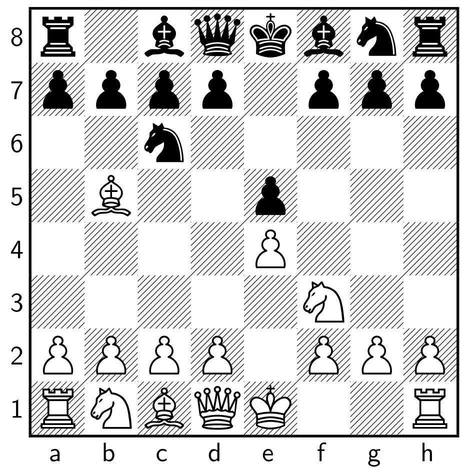


Figure 9.1: A perfect-information EFG solved via backward induction. P1 maximizes leaf node values and P2 minimizes. First P1 picks the best action at the bottom node. Then we replace that P1 node with its value 2. Then P2 picks the best action at each of their bottom nodes, and we replace those nodes with their value. Finally, P1 chooses an optimal action at the root.

1 e4 e5 2 ♘f3 ♘c6 3 ♗b5



In order to decide a next move for black, we can now perform real-time search. We perform backward induction starting at the subgame rooted at the current board state. What this means is that we try all sequences of legal moves starting with the current state, and then we pick the best action based on having solved the subgame via backward induction. However, unless we are close to the end of the game, the size of the subgame usually makes backward induction much too slow. Instead, the search is performed only up to a certain depth, say 10 moves ahead. This generally won't get us to a leaf node of the game, and so instead we replace the nodes at depth 10 with fake leaf nodes that we assign some heuristic estimate of the unique value that would have resulted from backward induction (we will call these fake leaf nodes *subgame leaf nodes*). In order to do that, we need to construct an estimate of what value an internal node would have in the solution. A visualization is shown in Figure 9.2.

In order to estimate the value of some internal node h in the game tree, we assume that we have some *value estimator* $v : H' \to \mathbb{R}$, where H' is the set of nodes in the game tree that are leaf nodes in the subgame. Each subgame leaf node h is then assigned the value $v(h)$ in the subgame. In perfect-information games each node h has some unique value associated to the solution arising from backward induction. In that case, our goal is simply to have $v(h)$ be a good approximation to this unique value. If $v(h)$ provides perfect estimates then backward induction in the subgame recovers the solution to the original game.

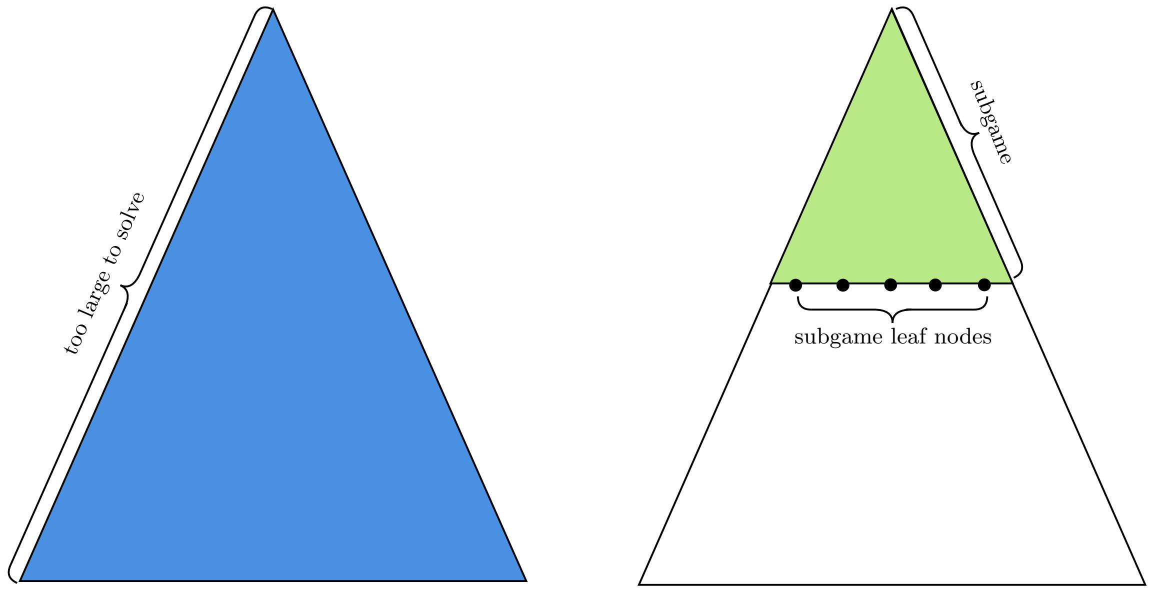


Figure 9.2: A large game truncated to a depth-limited subgame starting at the root.

So how do you get a value estimator? It can be handcrafted based on domain knowledge (this was done for *Deep Blue*, a chess AI which beat Garry Kasparov, at the time considered the best chess player in the world); it can be learned by training on expert human games (this was done by *AlphaGo*, a Go AI which beat Lee Sidol, a top-tier professional Go player); or finally it can be done via self-play (this was done by *AlphaZero*, a generalization of AlphaGo, and *Pluribus*, a poker AI that beat humans at 6-player poker).

For imperfect-information games such as poker, things are more complicated. The primary issue is that backward induction no longer works: The value of a given node cannot be understood purely in terms of the subtree rooted at the node. Instead, we must take into account the rest of the game tree. Further complicating matters is the fact that a node does not have a single well-defined value; the value of a node may change depending on which Nash equilibrium we are considering. Finally, even if we manage to estimate the value of a node in equilibrium, we may end up choosing a strategy where the opponent can best respond in order to exploit us in the truncated part of the game tree. This is easily seen by considering the EFG representation of rock-paper-scissors: At the root node player 1 chooses rock, paper, or scissors. Then, player 2 has a single information set containing all three nodes corresponding to each choice for player 1, and they choose rock, paper, or scissors at that information set. If we truncate the game at depth 1 and assign each player 2 node its value in equilibrium (which is 0), then player 1 ends up with 3 actions, all leading to a payoff of 0. Thus, for the subgame player 1 can choose any pure strategy, e.g. always play rock, and based on the subgame think that they achieve a value of zero. However, once we play in real time, if our opponent knows that we truncated the game and picked rock, they may exploit us by playing paper.

We can resolve this issue as follows: instead of having a function $v(h)$, we can have a function $v(h|p)$ that estimates the value of h, conditional on p. Here p is a probability distribution over the subgame leaf nodes. For the rock-paper-scissors example, $v(h_r|[p_r, p_p, p_s])$ would estimate the value of, say, the rock node h_r conditional on the distribution $[p_r, p_p, p_s]$ over the 3 possible nodes. This is obviously a much more complicated value estimator, since we are now trying to construct a mapping from $H' \times \Delta^{|H'|}$ to $\mathbb{R}$. This is the approach taken in the *DeepStack* poker AI, which beat a group of professional poker players in two-player no-limit Texas hold'em. Values are estimated using a deep neural network that was pretrained by generating random distributions over subgame leaf nodes, and then solving each of the subgames defined by truncating the *top* of the game, and having a chance root node that randomizes over the subgame leaf nodes using the randomly generated distribution.

Once an estimator v has been constructed, real-time search with this setup looks as follows:

1. Define a subgame by looking k moves ahead

2. Solve the subgame using a regret-minimization algorithm for EFGs (e.g. CFR or OMD with dilated DGF)

3. What are the leaf node values that we should use?

4. On each iteration t of the regret-minimization algorithm:

   (a) Set strategies x_t, y_t based on RM algorithms

   (b) x_t, y_t defines a probability distr. $p(Z)$ over subgame leaf nodes

   (c) For each leaf node z, ask value network for estimates $v(z, p(Z))$

   (d) Set loss for the x player to $g_t = A_t y_t$, where A_t is the payoff matrix associated to the subgame with subgame leaf estimates $v(z, p(Z))$. Define the loss for the y player analogously.

If the value network is perfect, then this setup computes a strategy for the subgame that is part of some Nash equilibrium in the full game.

To summarize the approach described here: we train our value network offline, e.g. by generating random distributions over nodes, and solving those subgames. This generates training data. Then, during play we use the already-trained value network to solve subgames as we encounter them.

This still leaves open the question of how to solve the subgames needed to create the value network, since those subgames could be very large themselves (e.g. subgames starting near the root of the game tree). One way to do it is to start by randomly generating shallow games near the bottom of the game, say depth d_1. Once we have a good value network for predicting the value of nodes at deptth d_1, we can move up one level. Next, we randomly generate distributions over nodes at depth d_2, and truncate those games at depth d_1 using our value network that we already constructed for depth d_1. We can then apply this recursively.

So far we have described our methodology and examples as if we are solving a depth-limited subgame starting at the root node of the game tree. However, in practice we would like to solve subgames starting at arbitrary decision points in the game. In perfect-information games this is easily done. We may treat it exactly the same as solving from the root, since every node provides a well-defined subgame with that node as the root. However, in imperfect-information games this is not so.

To construct an imperfect-information EFG subgame, we assume that we have so far been playing according to some *blueprint* strategy which we computed ahead of time (our opponent need not follow the blueprint strategy in practice). Typically this blueprint strategy would be computed using CFR$^+$ on a very coarsened abstraction of the game.

When constructing a non-root subgame in an imperfect-information game, we will in general not know exactly which node we are at, and so instead we would have to start the subgame at the information set that we are currently at. But even taking all of the nodes in the current information set as the root (and applying Bayes' rule to derive a chance node that selects among them), will not be enough. In particular, nodes in subtrees rooted at the information set may be in information sets that contain nodes that are *not* in any of the subtrees. To remedy this, we construct our subgame by starting with all nodes in subtrees rooted at the current information set. Then, we add to our subgame every node that shares an information set with at least one node currently in our subgame. We then repeatedly add nodes in this fashion, until we reach the point where there are *no* nodes outside the subgame which share an information set with any node in our (now much larger) subgame. Finally, in order to finish our construction we need a probability distribution over all the nodes that are at the top level (i.e. same level as the information set we wanted to create a subgame for) of this new subgame. The most naive approach would be to make a single chance node as the root, and use the conditional distribution over the set of top-level nodes given our blueprint strategy. This approach is typically called *unsafe subgame solving.* The reason it is called unsafe is that we are generally not guaranteed that we will be weakly better off by applying subgame solving, as compared to our blueprint strategy. By not considering the rest of the game, it turns out that we might open ourselves up to exploitation. Nonetheless, unsafe subgame solving is often used in practice.

There are various methods for performing "safe" subgame solving. These typically require adding additional gadgets to the unsafe subgame construction, either by enforcing that the opponent achieves a certain level of utility in the subgame (this prevents us from overfitting to the subgame), or replacing the initial chance node with a number of opponent nodes, where they can reject the subgame unless they achieve a certain utility level.

9.4 Historical Notes

Search was used in several poker AIs that beat human poker players of various degrees of expertise, both in two-player poker [95, 20] and 6-player poker [22].

Endgame solving, where we solve the remainder of the game, was studied in the unsafe version by Ganzfried and Sandholm [67]. Safe endgame solving was studied by Burch et al. [28], Moravcik et al. [94] and Brown and Sandholm [19]. The more general version of subgame solving where we do not have to solve to the end of the game was studied by Moravčík et al. [95], Brown et al. [24, 23].

Chapter 10

Intro to Fair Division and Market Equilibrium

10.1 Fair Division Intro

In this chapter we start the study of fair allocation of resources to a set of individuals. We start by focusing on the *fair division* setting. In fair division, we have one or more items that we wish to allocate to a set of agents, under the assumption that the items are infinitely-divisible, meaning that we can perform fractional allocation. In the next chapter we will study the setting with discrete items. The goal will be to allocate the items in a manner that is efficient, while attempting to satisfy various notions of fairness towards each individual agent. Fair allocation has many applications such as assigning course seats to students, pilot-to-plane assignment for airlines, dividing estates, chores, or rent, and fair recommender systems.

In this note we study fair division problems with the following setup: we have a set of m infinitely-divisible items that we wish to divide among n agents. Without loss of generality we may assume that each item has supply 1. We will denote the bundle of items given to agent i as $x_i \in \mathbb{R}_+^m$, where x_{ij} is the amount of item j that is allocated to agent i. Each agent has some utility function $u_i(x_i) \in \mathbb{R}_+$ denoting how much they like the bundle x_i. We shall use $x \in \mathbb{R}_+^{n \times m}$ to denote an assignment of items to agents, where x_{ij} is how much agent i gets of item j. We will later study the indivisible setting.

Given all of the above, we would like to choose a "good" assignment x of items to agents. However, "good" turns out to be very complicated in the setting of fair division, as there are many possible desiderata we may wish to account for.

First, we would like the allocation to somehow be efficient, meaning that it should lead to high utilities for the agents. One option would be to try to maximize the *social welfare* $\sum_i u_i(x_i)$, the sum of agent utilities. However, this turns out to be incompatible with the fairness notions that we will introduce later. An easy criticism of social welfare in the context of fair division is that it favors *welfare monsters*: agents with much greater capacity for utility are given more items[1]. Instead, we shall focus on the much less satisfying notion of *Pareto optimality*: we wish to find an allocation x such that for *every* other allocation x', if some agent i' is better off under x', then some other agent is strictly worse off. In other words, x should be such that no other allocation weakly improves all agent's utilities, unless all utilities stay the same.

We will consider the following measures of how fair an allocation x is:

- *No envy*: x has no envy if for every pair of agents i, i', $u_i(x_i) \geq u_i(x_{i'})$. In other words, every agent likes their own bundle at least as much as that of anyone else
- *Proportionality*: x satisfies proportionality if $u_i(x_i) \geq u_i\left(\vec{1} \cdot \frac{1}{n}\right)$. That is, every agent likes their bundle x_i at least as well as the bundle where they receive $\frac{1}{n}$ of every item. This property is also sometimes known as the *fair shares* property, because, absent any other information or valuations, the most natural way to divide items would be to simply give every agent an equal share of each item. Proportionality ensures that agents are at least as happy as under such an allocation.

[1]See also `https://existentialcomics.com/comic/8`

We begin our study of fair division mechanisms with a classic: *competitive equilibrium from equal incomes* (CEEI). In CEEI, we construct a mechanism for fair division by giving each agent a unit budget of fake currency (or *funny money*), computing what is called a competitive equilibrium (also known as *Walrasian equilibrium* or *market equilibrium*; we will use the latter terminology) under this new market, and using the corresponding allocation as our fair division. The fake currency is then thrown away, since it had no purpose except to define a market.

To understand this mechanism, we first introduce *market equilibrium.* In a market equilibrium, we wish to find a set of prices $p \in \mathbb{R}_+^m$ for each of the m items, as well as an allocation x of items to agents such that everybody is assigned an optimal allocation given the prices and their budget. Formally, the *demand set* of an agent i with budget B_i is

$$D(p) = \text{argmax}_{x_i \geq 0} u_i(x_i) \text{ s.t. } \langle p, x_i \rangle \leq B_i$$

A market equilibrium is an allocation-price pair (x, p) s.t. $x_i \in D(p)$ for all agents i, and $\sum_i x_{ij} = 1$.

CEEI is a perfect solution to our desiderata that we asked for. It is Pareto optimal (every market equilibrium is Pareto optimal by the *first welfare theorem*). It has no envy: since each agent has the same budget $B_i = 1$ in CEEI and every agent is buying something in their demand set, no envy must be satisfied, since they can afford the bundle of any other agent. Finally, proportionality is satisfied, since each agent can afford the bundle where they get $\frac{1}{n}$ of each item (convince yourself why).

Market-equilibrium-based allocation for divisible items has applications in large-scale Internet markets. First, it can be applied in *fair recommender systems.* As an example, consider a job recommendations site. It's a two-sided market. On one side are the users, whom view job ads. On the other side are the companies creating job ads. Naively, a system might try to simply maximize the number of job ads that users click on, or apply to. This can lead to extremely imbalanced allocations, where a few job ads get a huge number of views and applicants, which is bad both for users and the companies. Instead, the system may wish to fairly distribute user views across the many different job ads. In that case, CEEI can be used. In this setting the agents are the job ads, and the items are slots in the ranked list of job ads shown to the user. Secondly, there are strong connections between market equilibrium and the allocation of ads in large-scale Internet ad markets. This connection will be explored in detail in a later note.

Motivated by the application to fair division, we will now cover market equilibrium, both when they exist and how to find one when they do.

10.2 Fisher Market

We first study market equilibrium in the *Fisher market* setting. We have a set of m infinitely-divisible items that we wish to divide among n buyers. Without loss of generality we may assume that each item has supply 1. We will denote the bundle of items given to buyer i as x_i, where x_{ij} is the amount of item j that is allocated to buyer i. Each buyer has some utility function $u_i(x_i) \in \mathbb{R}_+$ denoting how much they like the bundle x_i. We shall use x to denote an assignment of items to buyers. Each buyer is endowed with a budget B_i of currency.

10.2.1 Linear Utilities

We start by studying the simplest setting, where the utility of each buyer is linear. This means that every buyer i has some valuation vector $v_i \in \mathbb{R}^m$, and $u_i(x_i) = \langle v_i, x_i \rangle$.

Amazingly, there is a nice convex program for computing a market equilibrium. Before giving the convex program, let's consider some properties that we would like. First, if we are going to find a feasible allocation, we would obviously like the supply constraints to be respected, i.e.

$$\sum_i x_{ij} \leq 1, \forall j.$$

Secondly, since a buyer's demand does not change even if we rescale their valuation by a constant, we would like the optimal solution to our convex program to also remain unchanged. Similarly, splitting the budget of a buyer into two separate buyers with the same valuation function should leave the allocation unchanged. These conditions are satisfied by the budget-weighted geometric mean of the utilities:

$$\left(\prod_i u_i(x_i)^{B_i}\right)^{1/\sum_i B_i}.$$

Since taking roots does not affect optimality, and taking the log of the whole expression, this is equivalent to the following convex program, known as the *Eisenberg-Gale* (EG) convex program:

$$\begin{array}{lll|l} \max\limits_{x\geq 0} & \sum\limits_i B_i \log\langle v_i, x_i\rangle & & \text{Dual variables} \\ s.t. & \sum\limits_i x_{ij} \leq 1, & \forall j = 1,\ldots,m, & p_j \end{array} \tag{EG}$$

On the right are the dual variables associated to each constraint. It is easy to see that this is a convex program. First, the feasible set is defined by linear inequalities. Second, we are taking a max of a sum of concave functions composed with linear maps. Since taking a sum and composing with a linear map both preserve concavity we get that the objective is concave.

The solution to the primal problem x along with the vector of dual variables p yields a market equilibrium. Here we assume that for every item j there exists i such that $v_{ij} > 0$, and every buyer values at least one item above 0.

Theorem 21. *The pair of allocations x and dual variables p from EG forms a market equilibrium.*

Proof. To see this, we need to look at the KKT conditions of the primal and dual variables. Writing the Lagrangian relaxation and applying Sion's minimax theorem (the most general version of Sion's minimax theorem only requires compactness in one of the variables, which we have due gives

$$\begin{aligned} &\min_{p\geq 0}\max_{x\geq 0}\sum_i B_i \log\left(\langle v_i, x_i\rangle\right) + \sum_j p_j\left(1-\sum_i x_{ij}\right) \\ =&\min_{p\geq 0}\max_{x\geq 0}\sum_i \left[B_i \log\left(\langle v_i, x_i\rangle\right) - \langle p, x_i\rangle\right] + \sum_j p_j \end{aligned} \tag{10.1}$$

Looking at optimality conditions we get

1. For all items j: $p_j > 0 \Rightarrow \sum_i x_{ij} = 1$
2. For all buyers i: $\frac{B_i}{\langle v_i, x_i\rangle} \leq \frac{p_j}{v_{ij}}$
3. For all pairs i, j: $x_{ij} > 0 \Rightarrow \frac{B_i}{\langle v_i, x_i\rangle} = \frac{p_j}{v_{ij}}$

The first condition shows that every item is fully allocated, since for every j there is some buyer i with non-zero value and by the second condition $p_j \geq \frac{v_{ij}B_i}{\langle v_i, x_i\rangle} > 0$.

The second condition for market equilibrium is that every buyer is assigned a bundle from their demand set. We will use $\beta_i = \frac{B_i}{\langle v_i, x_i\rangle} = \frac{B_i}{u_i(x_i)}$ to denote the *utility price* that buyer i pays. First off, by the second condition we have that the utility price that buyer i gets satisfies

$$\beta_i \leq \frac{p_j}{v_{ij}}.$$

By the third condition, we have that if $x_{ij} > 0$ then for all other items j' we have

$$\frac{p_j}{v_{ij}} = \beta_i \leq \frac{p_{j'}}{v_{ij'}}.$$

Thus, any item j that buyer i is assigned has at least as low of a utility price as any other item j'. In other words, they only buy items that have the best bang-per-buck among all the items. Thus we get that they only purchase optimal items, it remains to show that they spent their whole budget. Multiplying the third condition by x_{ij} and rearranging gives

$$x_{ij} v_{ij} \frac{B_i}{\langle v_i, x_i \rangle} = p_j x_{ij},$$

for any j such that $x_{ij} > 0$. Summing across all such j yields

$$\sum_j p_j x_{ij} = \sum_j x_{ij} v_{ij} \frac{B_i}{\langle v_i, x_i \rangle} = \langle v_i, x_i \rangle \frac{B_i}{\langle v_i, x_i \rangle} = B_i.$$

□

EG gives us an immediate proof of existence for the linear Fisher market setting: the feasible set is clearly non-empty, and the max is guaranteed to be achieved.

In a previous lecture note we referenced Pareto optimality as a property of market equilibrium. It is now trivial to see that Pareto optimality holds in Fisher-market equilibrium: since it is a solution to EG, it must be. Otherwise we construct a solution with strictly better objective!

From the EG formulation we can also see that the equilibrium utilities and prices are in fact unique. First note that any market equilibrium allocation would satisfy the optimality conditions of EG, and thus be an optimal solution. But if there were more than one set of utility vectors that were equilibria, then by the strong concavity of the log we would get that there is a strictly better solution, which is a contradiction. That equilibrium prices are unique now follows from the third optimality condition, since all terms except the utilities are constants.

10.3 More General Utilities

It turns out that EG can be applied to a broader class of utilities. This class is the set of utilities that are concave, homogeneous, and continuous.

In that case we get an optimization problem of the form

$$\begin{array}{llll} \max\limits_{x \geq 0} & \sum\limits_i B_i \log u_i(x_i) & & \text{Dual variables} \\ s.t. & \sum\limits_i x_{ij} \quad \leq 1, \quad \forall j = 1, \ldots, m, & \Big| & p_j \end{array} \tag{EG}$$

This is still a convex optimization problem, since composing a concave and nondecreasing function (the log) with a concave function (u_i) yields a concave function.

Beyond linear utilities, the most famous classes of utilities that fall under this category is:

1. Cobb-Douglas utilities: $u_i(x_i) = \prod_j (x_{ij})^{a_{ij}}$, where $\sum_j a_{ij} = 1$, $a_{ij} \geq 0$
2. Leontief utilities: $u_i(x_i) = \min_j \frac{x_{ij}}{a_{ij}}$
3. The family of constant elasticity of substitution (CES) utilities: $u_i(x_i) = \left(\sum_j a_{ij} x_{ij}^{\rho} \right)^{1/\rho}$, where a_{ij} are the utility parameters of a buyer, and ρ parameterizes the family, with $-\infty < \rho \leq 1$ and $\rho \neq 0$

CES utilities turn out to generalize all the other utilities we have seen so far: Leontief utilities are obtained as ρ approaches $-\infty$, Cobb-Douglas utilities as ρ approaches 0, and linear utilities when $\rho = 1$. More generally, $\rho < 0$ means that items are complements, whereas $\rho > 0$ means that items are substitutes.

If u_i is continuously differentiable then the proof that EG computes a market equilibrium in this more general setting essentially follows that of the linear case. The only non-trivial change is that when we derive optimality conditions by taking the derivative of the Lagrangian with respect to x_i we get

1. $\frac{B_i}{u_i(x_i)} \leq \frac{p_j}{\partial u_i(x_i)/\partial x_{ij}}$

2. $x_{ij} > 0 \Rightarrow \frac{B_i}{u_i(x_i)} = \frac{p_j}{\partial u_i(x_i)/\partial x_{ij}}$

In order to prove that buyers spend their budget exactly in this setting we can apply Euler's homogeneous function theorem $u_i(x_i) = \sum_j x_{ij} \frac{\partial u_i(x_i)}{\partial x_{ij}}$ to get

$$\sum_j x_{ij} p_j = \sum_j x_{ij} \frac{\partial u_i(x_i)}{\partial x_{ij}} \frac{B_i}{u_i(x_i)} = B_i.$$

10.4 Computing Market Equilibrium

So now we know how to write a market equilibrium problem as a convex program. How should we solve it? One option is to build the EG convex program explicitly using mathematical programming software. However, most contemporary software is not very good at handling this kind of objective function (formally this falls under exponential cone programming, which is still relatively new). As of 2019, my experience with open-source solvers was that they fail at 150 items and 150 buyers, with randomly-generated valuations. The Mosek solver [96] is currently the only industry-grade solver that supports exponential cone programming (support for exponential cones was only just added in 2018). It fares much better, and scales to a few thousand buyers and items. For problems of moderate-to-large size, this is the most effective approach. However, for very large instances, the iterations of the interior-point solver used in Mosek become too slow.

Instead, for extremely large problems we may invoke some of our earlier results on saddle-point problems. In particular, the formulation (10.1) is amenable to online mirror descent and the folk-theorem based approach for solving saddle-point problems. In that framework, we can interpret the repeated game as being played between a pricer trying to minimize over prices p, and the set of buyers choosing allocations x.

As an exercise, convince yourself that the OMD/folk theorem approach works. Pay particular attention to the assumptions needed for online mirror descent.

10.5 Historical Notes

The original Eisenberg-Gale convex program was given for linear utilities in [55]. [54] later extended it to utilities that are concave, continuous, and homogeneous.

Fairly assigning course seats to students via market equilibrium was studied by Budish [26]. Goldman and Procaccia [72] introduce an online service `spliddit.org` which has a user-friendly interface for fairly dividing many things such as estates, rent, fares, and others. The motivating example of fair recommender systems, where we fairly divide impressions among content creators via CEEI was suggested in [87] and [85]. Similar models, but where money has real value, were considered for ad auctions with budget constraints by several authors [15, 45, 46]

There is a rich literature on various iterative approaches to computing market equilibrium in Fisher markets. One can apply first-order methods or regret-minimization approaches to the saddle-point formulation (10.1) directly, which was done in Kroer et al. [87] and Gao et al. [68]. There is a large literature on interpreting first-order methods through the lens of dynamics between a pricer who increases and decreases prices as items become oversubscribed and undersubscribed, and buyers report their preferred bundles, or make gradient-steps in the direction of their preference [41, 13, 37]. There is also a literature deriving auction-like algorithms, which can similarly sometimes be viewed as instantiations of gradient descent and related algorithms [10, 102].

A fairly comprehensive recent overview of fair division can be found at `https://www.cs.toronto.edu/~nisarg/papers/Fair-Division-Tutorial.pdf`.

Chapter 11

Fair Allocation with Indivisible Goods

11.1 Introduction

In this lecture note we study the problem of performing fair allocation when the items are indivisible. This setting presents a number of challenges that were not present in the divisible case.

It is obviously an important setting in practice. For example, the website `http://www.spliddit.org/` allows users to fairly split estates, financial assets, toys, or other goods. Another important application is that of fairly allocating course seats to students. This setting is even more intricate, because valuations in that setting are combinatorial. In order to design suitable mechanisms for fairly dividing discrete goods, we will need to reevaluate our fairness concepts.

11.2 Setup

We have a set of m indivisible goods that we wish to divide among n agents. We assume that each good has supply 1. We will denote the bundle of goods given to agent i as x_i, where x_{ij} is the amount of good j that is allocated to buyer i. The set of feasible allocations is then $\{x | \sum_i x_{ij} \le 1, x_{ij} \in \{0, 1\}\}$

Unless otherwise specified, each agent is assumed to have a linear utility function $u_i(x_i) = \langle v_i, x_i \rangle$ denoting how much they like the bundle x_i.

11.3 Fair Allocation

In the case of indivisible items, several of our fairness properties become much harder to achieve. We will assume that we are required to construct a Pareto-efficient allocation.

Proportional fairness doesn't even make sense anymore: it rested on the idea of assigning each agent their fractional share $\frac{1}{n}$ of each item. There is however, a suitable generalization of proportionality that does make sense for the indivisible case: the *maximin share (MMS) guarantee*: For agent i, their MMS is the value they would get if they get to divide the items up into n bundles, and are required to take the worst bundle. Formally:

$$\begin{aligned} \max_{x \ge 0} \; & \min_j u_i(x_j) \\ \text{s.t.} & \sum_i x_{ij} \le 1, \forall j \\ & x_{ij} \in \{0, 1\}, \forall i, j \end{aligned}$$

We say that an allocation x is an MMS allocation if every agent i receives utility $u_i(x_i)$ that is at least as high as their MMS guarantee. In the case of 2 agents, an MMS allocation always exists. As an exercise, you might try to come up with an algorithm for finding such an allocation[1]

[1]Solution: compute one of the solutions to agent 1's MMS computation problem. Then let agent 2 choose their favorite

In the case of 3 or more agents, such a solution may not exist. The counterexample is very involved, so we won't cover it here.

Theorem 22. *For $n \geq 3$ agents, there exist additive valuations for which an MMS allocation does not exist. However, an allocation such that each agent receives at least $\frac{3}{4}$ of their MMS guarantee always exists.*

The original spliddit algorithm for dividing goods worked as follows: first, compute the $\alpha \in [0,1]$ such that every agent can be guaranteed an α fraction of their MMS guaranteee (this always ends up being $\alpha = 1$ in practice). Then, subject to the constraints $u_i(x_i) \geq \alpha \text{MMS}_i$, a social welfare-maximizing allocation was computed. However, this can lead to some weird results.

Example 3. *Three agents each have valuation 1 for 5 items. In that case, the MMS guarantee is 1 for each agent. But now the social welfare-maximizing solution can allocate three items to agent 1, and 1 item each to agents 2 and 3. Obviously a more fair solution would be to allocate 2 items to 2 agents, 1 item to the last agent.*

One observation we can make about the 3/1/1 solution versus the 2/2/1 solution is that envy is strictly higher in the 3/1/1/ solution.

With the above motivation, let us consider envy in the discrete setting. It is easy to see that we generally won't be able to get envy-free solutions if we are required to assign all items. Consider 2 agents splitting an inheritance: a house worth \$500k, a car worth \$10k, and a jewelry set worth \$5k. Since we have to give the house to a single agent, the other agent is guaranteed to have envy. Thus we will need a relaxed notion of envy:

Definition 4. *An allocation x is* envy-free up to one good *(EF1) if for every pair of agents i, k, there exists an item j such that $x_{kj} = 1$ and $u_i(x_i) \geq u_i(x_k - e_j)$, where e_j is the j'th basis vector.*

Intuitively, this definition says that for any pair of agents i, k such that i envies k, that envy can be removed by removing a single item from the bundle of k. Note that requiring EF1 would have forced us to use the 2/2/1 allocation in Example 3.

For linear utilities, an EF1 allocation is easily found (if we disregard Pareto optimality). As an exercise, come up with an algorithm for computing an EF1 allocation for linear valuations[2] In fact, EF1 allocations can be computed in polynomial time for any monotone set of utility functions (meaning that if $x_i \geq x_i'$ then $u_i(x_i) \geq u_i(x_i')$).

However, ideally we would like to come up with an algorithm that gives us EF1 as well as Pareto efficiency. To achieve this, we will consider the product of utilities, which we saw previously in Eisenberg-Gale. This product is also called the *Nash welfare* of an allocation:

$$NW(x) = \prod_i u_i(x_i)$$

The *max Nash welfare* (MNW) solution picks an allocation that maximizes $NW(x)$:

$$\begin{aligned} \max_x & \prod_i u_i(x_i) \\ s.t. & \sum_i x_{ij} \leq 1, \forall j \\ & x_{ij} \in \{0,1\}, \forall i, j \end{aligned}$$

Note that here we have to worry about the degenerate case where $NW(x) = 0$ for *all* x, meaning that it is impossible to give strictly positive utility to all agents. We will assume that there exists x such that $NW(x) > 0$. If this does not hold, typically one seeks a solution that maximizes the number of agents with strictly positive utility, and then the largest MNW achievable among subsets of that size is chosen.

The MNW solution turns out to achieve both Pareto optimality (obviously, since otherwise it would not solve the MNW optimization problem), and EF1:

bundle, and give the other bundle to agent 1. Agent 1 clearly receives their MMS guarantee, or better. Agent 2 also does: their MMS guarantee is at most $\frac{1}{2}\|v_2\|_1$, and here they receive utility of at least $\frac{1}{2}\|v_2\|_1$.

[2]This is achieved by the round-robin algorithm: simply have agents take turns picking their favorite item. It is easy to see that EF1 is an invariant of the partial allocations resulting from this process.

Theorem 23. *The MNW solution for linear utilities is Pareto optimal and EF1.*

Proof. Let x be the MNW solution. Say for contradiction that agent i envies agent k by more than one good. Let j be the item allocated to agent k that minimizes the ratio $\frac{v_{kj}}{v_{ij}}$. Let x' be the same allocation as x, except that $x'_{ij} = 1, x'_{kj} = 0$. The proof is concluded by showing that $NW(x') > NW(x)$, which contradicts optimality of x for the MNW problem.

Using the linearity of utilities we have $u_i(x'_i) = u_i(x_i) + v_{ij}$ and $u_k(x'_k) = u_k(x_k) - v_{kj}$. Every other utility stays the same. Now we have

$$\begin{aligned} &\frac{NW(x')}{NW(x)} > 1 \\ \Leftrightarrow\ &\frac{[u_i(x_i) + v_{ij}] \cdot [u_k(x_k) - v_{kj}]}{u_i(x_i)u_k(x_k)} > 1 \\ \Leftrightarrow\ &\left[1 + \frac{v_{ij}}{u_i(x_i)}\right] \cdot \left[1 - \frac{v_{kj}}{u_k(x_k)}\right] > 1 \\ \Leftrightarrow\ &\frac{v_{kj}}{v_{ij}}\left[u_i(x_i) + v_{ij}\right] < u_k(x_k) \end{aligned} \tag{11.1}$$

By how we choose j we have

$$\frac{v_{kj}}{v_{ij}} \le \frac{\sum_{j' \in x_k} v_{kj'}}{\sum_{j' \in x_k} v_{ij'}} \le \frac{u_k(x_k)}{u_i(x_k)},$$

and by the envy property we have

$$u_i(x_i) + v_{ij} < u_i(x_k).$$

Now we can multiply together the last two inequalities to get (11.1). □

The MNW solution also turns out to give a guarantee on MNW, but not a very strong one: every agent is guaranteed to get $\frac{2}{1+\sqrt{4n-3}}$ of their MMS guarantee, and this bound is tight. Luckily, in practice the MNW solution seems to fare much better. On Spliddit data, the following ratios are achieved. In the table below are shown the MMS approximation ratios across 1281 "divide goods" instances submitted to the Spliddit website for fairly allocating goods

MMS approximation ratio intervals	$[0.75, 0.8)$	$[0.8, 0.9)$	$[0.9, 1)$	1
% of instances in interval	0.16%	0.7%	3.51%	95.63%

Over 95% of the instances have every player receive their full MMS guarantee.

11.4 Computing Discrete Max Nash Welfare

11.4.1 Complexity

The problem of maximizing Nash welfare is generally not easy. In fact, the problem turns out to be not only NP-hard, but NP-hard to approximate within a factor $\mu \approx 1.00008$ (the best currently-known approximation factor is 1.45, so the gap between 1.00008 and 1.45 is open).

The reduction is based the vertex-cover problem on 3-regular graphs, which is NP-hard to approximate within factor ≈ 1.01. A 3-regular graph is a graph where each vertex has degree 3.

The proof is not particularly illuminating, so we will skip it here. However, let's see a quick way to prove a simpler statement: that the problem is NP-hard even for 2 players with identical linear valuations. Consider the following

Definition 5. PARTITION *problem: you are given a multiset of integers $S = \{s_1, \dots, s_m\}$ (potentially with duplicates), and your task is to figure out if there is a way to partition S into two sets S_1, S_2 such that $\sum_{i \in S_1} s_i = \sum_{i \in S_2} s_2$.*

We may now construct an MNW instance as follows: we create two agents and m items. Each agent has value s_j for item j. Now by the AM-GM inequality (2d case: $\sqrt{xy} \leq \frac{x+y}{2}$, with equality iff $x = y$) there exists a correct partitioning if and only if the MNW allocation has value $(\frac{1}{2}\sum_j s_j)^2$.

This result can be extended to show strong NP-hardness by considering the k-EQUAL-SUM-SUBSET problem: given a multiset $\mathcal{S}$ of $x_1, \dots, x_n$ positive integers, are there k nonempty disjoint subsets $S_1, \dots, S_k \subset \mathcal{S}$ such that $sum(S_1) = \ldots = sum(S_k)$. The exact same reduction as before works, but with k agents rather than 2.

11.4.2 Algorithms

Given these computational complexity problems, how should we compute an MNW allocation in practice?

We present two approaches here. First, we can take the log of the objective, to get a concave function. After taking logs, we get the following mixed-integer exponential-cone program:

$$\begin{array}{lrll} \max & \sum_i \log u_i & & \\ s.t. & u_i & \leq \langle v_i, x_i \rangle, & \forall i = 1, \dots, n \\ & \sum_i x_{ij} & \leq 1, & \forall j = 1, \dots, m \\ & x_{ij} & \in \{0,1\}, & \forall i, j \end{array} \tag{11.2}$$

This is simply the discrete version of the Eisenberg-Gale convex program. One approach is to solve this problem directly, e.g. using Mosek.

Alternatively, we can impose some additional structure on the valuation space: if we assume that all valuations are integer-valued, then we know that $u_i(x_i)$ will take on some integer value in the range 0 to $\|v_i\|_1$. In that case, we can add a variable w_i for each agent i, and use either (1) the linearization of the log at each integer value, or (2) the linear function from the line segment $(\log k, k), (\log(k+1), k+1)$, as upper bounds on w_i. This gives $\frac{1}{2}\|v_i\|_1$ constraints for each i using the line segment approach (the linearization uses twice as many constraints), but ensures that w_i is equal to $\log\langle v_i, x_i \rangle$ for all integer-valued $\langle v_i, x_i \rangle$. Using the line segment approach gives the following mixed-integer linear program (MILP):

$$\begin{array}{lrll} \max & \sum_i w_i & & \\ s.t. & w_i & \leq \log k + [\log(k+1) - \log k] \times (\langle v_i, x_i \rangle - k), & \forall i = 1, \dots, n, k = 1, 3, \dots, \|v_i\|_1 \\ & \sum_j v_{ij} x_{ij} & \geq 1, & \forall i \\ & \sum_i x_{ij} & \leq 1, & \forall j = 1, \dots, m \\ & x_{ij} & \in \{0,1\}, & \forall i, j \end{array} \tag{11.3}$$

These two mixed-integer programs both have some drawbacks: For the first mixed-integer exponential-cone program, we must resort to much less mature technology than for mixed-integer linear programs. On the other hand, the discrete EG program is reasonably compact: the program is roughly the size of a solution. For the MILP, the good news is that MILP technology is quite mature, and so we might expect this to solve quickly. On the other hand, adding $n \times \|v_i\|_1$ additional constraints can be quite a lot, and could lead to slow LP solves as part of the branch-and-bound procedure.

Figure 11.1 shows the performance of the two approaches.

11.5 Historical Notes

The maximin share was introduced by Budish [26]. The results on nonexistence of MMS allocation, and an approximation guarantee of $\frac{2}{3}$ were given by Kurokawa et al. [89]. The approximation guarantee was improved to $\frac{3}{4}$ by Ghodsi et al. [69]. The application of MNW to fair division was proposed by Caragiannis et al. [31].

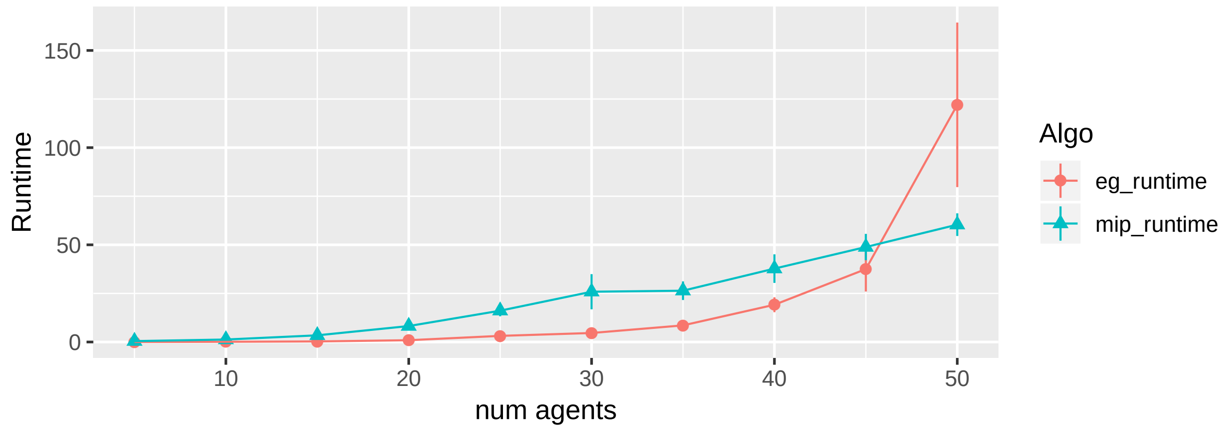


Figure 11.1: Plot showing the runtime of discrete Eisenberg-Gale and the MILP approach.

A really nice overview talk targeted at a technical audience is given by Ariel Procaccia here: `https://www.youtube.com/watch?v=7lUtS-l9ytI`. Most of the material here is based on his excellent presentations of these topics.

The 1.00008 inapproximability result was byZLee [92]. The 1.45-approximation algorithm was given by Barman et al. [6]. Strong NP-hardness of k-EQUAL-SUM-SUBSET is shown in Cieliebak et al. [39].

The MILP using approximation to the log at each integer point was introduced by Caragiannis et al. [32]. At the time, Mosek did not support exponential cones, and so they did not compare to the direct solving of discrete Eisenberg-Gale. The results shown here are the first direct comparison of the two, as far as I know.

Chapter 12

Internet Advertising Auctions: Position Auctions

12.1 Introduction

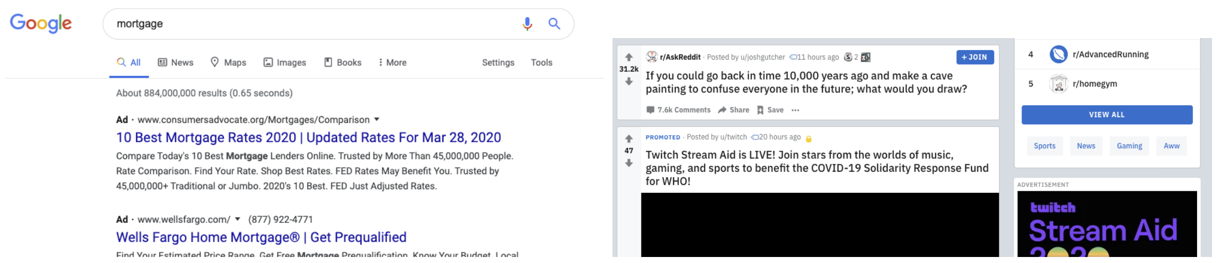
In this chapter we begin our study of more advanced auction concepts beyond first and second-price auctions. We will focus on a type of auction motivated by internet advertising auctions. Internet advertising auctions provide the funding for almost every free internet service such as google search, facebook, twitter, and so on. At the heart of these monetization schemes is a market design based around independently running auctions every time a user shows up. This happens many times per second, advertisers participate in thousands or even millions of auctions, have budget constraints that span the auctions, and each user query generates multiple potential slots for showing ads. For all these reasons, these markets turn out to require a lot of new theory for understanding them. Similarly, the scale of the problem necessitates the design of algorithmic agents for bidding on behalf of advertisers.

First we will introduce the *position auction*, which is a highly structured multi-item auction. There, we will look at the two most practically-important auction formats: the generalized second-price auction (GSP), and the Vickrey-Clarke-Groves (VCG) auction. Then in the following chapters, we will study auctions with budgets and repeated auctions.

12.1.1 Considerations for internet advertising

Consider the following problem: a user shows up and searches for the word “mortgage” on Google; now, you are Google, and you have thousands of ads that you could show to the user when returning their search result. Typically, google shows a few ads at the top of the results (say 2 ads, to be concrete); an example is shown in Fig. 12.1. This setting is referred to as the “sponsored search setting.” How do you decide which ads to show? And how do you decide how much to charge each advertiser for showing their ad? A natural suggestion would be to try to use auctions. Based on earlier chapters of the book, one might think of running four separate first or second-price auctions, one for each ad slot. In that case, it is clear how to decide winners and how to set prices. Yet this immediately runs into a problem: the same ad may win multiple auctions, and thus be shown in several slots. This looks bad for the user, and the advertiser almost surely does not want to pay for multiple slots. Instead, we need to design a multi-item auction. But we cannot simply use the multi-item generalization of e.g. the second-price auction, where each items is identical. This is because different slots are *not* identical: users are generally more likely to click on the first ad than the second ad, and so on. This motivates the *position auction*, which we study in this chapter.

The position auction model can also be used to approximate other settings such as the insertion of ads in a *news feed*; a news feed is the familiar infinitely-scrolling list of e.g. Facebook posts, Reddit posts, Instagram posts, or Twitter posts. For example, Reddit typically inserts 1 ad in the set of visible results before scrolling (see Figure 12.1 on the right), with another ad appearing in the next 10-15 results (this was tested on March 28th 2020). Similarly, Facebook and Twitter insert 1-2 sponsored posts near the top of the



About 884,000,000 results (0.65 seconds)

Ad · www.consumersadvocate.org/Mortgages/Comparison

10 Best Mortgage Rates 2020 | Updated Rates For Mar 28, 2020

Compare Today's 10 Best **Mortgage** Lenders Online. Trusted by More Than 45,000,000 People. Rate Comparison. Find Your Rate. Shop Best Rates. FED Rates May Benefit You. Trusted by 45,000,000+ Traditional or Jumbo. 2020's 10 Best. FED Just Adjusted Rates.

Ad · www.wellsfargo.com/ (877) 922-4771

Wells Fargo Home Mortgage® | Get Prequalified

31.2k r/AskReddit · Posted by u/joshgutcher 11 hours ago 2

If you could go back in time 10,000 years ago and make a cave painting to confuse everyone in the future; what would you draw?

47 PROMOTED · Posted by u/twitch 20 hours ago

Twitch Stream Aid is LIVE! Join stars from the worlds of music, gaming, and sports to benefit the COVID-19 Solidarity Response Fund for WHO!

4 r/AdvancedRunning

5 r/homegym

Figure 12.1: Left: A Google query for "mortgage" shows 2 ads. Organic search results follow further down. Right: The front page of Reddit. The second feed story is an ad.

feed. Truly capturing feed auctions does require some care, however. The assumption of there being a fixed number of items is incorrect for that setting. Instead, the number of ads shown depends on how far the user scrolls, the size of the ads, and what else is being shown in terms of organic content. We will focus on the simpler setting with a fixed number of slots, but properly handling feed auctions is an interesting problem.

Beyond the multi-item and budget aspects, internet advertising has a few other interesting quirks. Below these are discussed briefly, though we will mostly abstract away considerations around these issues.

Targeted advertising.

In a classical advertising setting such as TV or newspaper advertising, the same ad is shown to every viewer of a given TV channel, or every reader of a newspaper. This means that it is largely not feasible for smaller, and especially niche, retailers to advertise, since their return on investment is very low due to the small fraction of viewers or readers that fit their niche. All this changed with the advent of internet advertising, where niche retailers can perform much more fine-grained targeting of their ads. This has enabled many niche retailers to scale up their audience reach significantly.

One way that targeting can occur is directly through association with the search term in sponsored search. For example, by bidding on the search term "mortgage," a lender is effectively performing a type of targeting. However, a second type of targeting occurs by matching on query and user features (such targeting is used across many types of internet advertising including search, feed ads, and others). For example, a company selling surf boards might wish to target users at the intersection of the categories {age 16-30, lives in California}. Because each individual auction corresponds to a single user query, the idea of targeted advertising can be captured in the valuations that we will use for the buyers in our auction setup: each buyer corresponds to an advertiser, each auction corresponds to a query, and the buyer will have value zero for all items in a given auction if the associated query features do not match their targeting criteria.

Targeted advertising has the potential for some adverse effects. Of particular note are demographic biases in the types of ads being shown (a well-documented example is that in some settings, ads for new luxury housing developments were disproportionately shown to white people). In a later lecture note we will study such questions around demographic fairness. A second potential issue is that of user privacy. This is an interesting topic that we will unfortunately not have too much to say on, as it is outside the scope of the course.

Pay per click.

Another revolution compared to pre-internet advertising is the *pay per click* nature of most internet advertising auctions. Many advertisers are not actually interested in the user simply viewing their ad. Instead, their goal is to get the user to click on the ad, or even something downstream of clicking on the ad, such as selling the advertised product via the linked website. Because the platform, such as google, is in a much better position to predict whether a given user will click on a given ad, these auctions operate on a *cost per click* basis, rather than a cost per impression. What this means is that any given advertiser does not actually pay just because they won the auction and got their ad shown, instead they pay *only* if the user actually clicks on their ad.

From an auction perspective, this means that the valuations used in the auctions must take into account the probability that the user will click on the ad. Valuations are typically constructed by breaking down the value that a buyer i (in this case an advertiser) has for an item (which is a particular slot in the search query or user feed) into several components. The *value per click* of advertiser i is the value $v_i > 0$ they place on any user within their targeting criteria clicking on their ad (modern platforms generalize this concept to a value per *conversion*, where a conversion can be a click, an actual sale of a product, the user viewing a video, etc.). The *click-through-rate* is the likelihood that the user behind query j will click on the ad of advertiser i, independently of where on the page the ad is shown. We denote this by CTR_{ij}; we will assume that $CTR_{ij} = 0$ if query j does not fall under the targeting criteria of buyer i. Finally, the *slot qualities* $q_1, \dots, q_S$ are scalar values denoting the quality of each slot that an ad could end up in. These are monotonically decreasing values, indicating the fact that it's generally preferable to be shown higher up on the page. Now, finally, the value that buyer i has for being shown in slot s of query j is modeled as $v_{ijs} = v_i \cdot CTR_{ij} \cdot q_s$.

For the rest of the lecture note, we will assume that v_{ij} is the value that buyer i has for auction j; this value encodes the value per click, the CTR, and the targeting criteria (but can allow for more general valuations that do not decompose). Note that this assumes correct CTR predictions, which is obviously not true in practice. In practice the CTRs are estimated using machine learning, and it is of interest to understand which discrepancies this introduces into the market. Secondly, we are assuming that buyers are maximizing their expected utility, rather than observed utility. This is largely a non-problem, since they will participate in thousands or even millions of auctions, and thus their realized value can reasonably be expected to match the expectation (at least if the CTRs are correct). The slot quality q_s will be handled separately in the next section. Once we start discussing budgets, we will keep the presentation simple by assuming a single item per auction, thus avoiding the need for slot qualities.

12.2 Position Auctions

In the position auction model, a set of S *slots* are for sale. The slots are shown in ranked order, and the value that an advertiser derives from showing their ad in a particular slot s decomposes into two terms $v_{is} = v_i q_s$ where v_i is the value that the advertiser places on a user clicking on their ad, and q_s is the advertiser-independent click probability of slot s. Here we assume that v_i already incorporates the click-through rate (so in particular it could be that $v_i = v_i' \cdot CTR_i$ where v_i' is their actual value per click, and CTR_i is the click-through rate in the current auction). It is assumed that $q_1 \geq q_2 \geq \cdot \geq q_S$, i.e. the top slot is better than the second slot, and so on. Going back to the original setting, a position auction corresponds to the individual auction that is run when a particular user query shows up. Because we are analyzing this individual auction in isolation, we can drop the j index and simply assume that v_i gives the expected value per click for buyer i in the current auction.

Now suppose that the n advertisers submit bids $b \in \mathbb{R}_+^n$. Both auction formats we will use then proceed to perform allocation via welfare maximization, assuming that the bids are truthful. We will also refer to this as bid maximization. In particular, we sort b (suppose without loss of generality that the bids are conveniently already ordered by buyer index: $b_1 \geq b_2 \geq \dots \geq b_n$), and allocate the slots in order of bids (so buyer 1 with bid b_1 gets slot 1, buyer 2 gets slots 2, and so on up to bid b_S getting slot S).

Example 4. *Suppose we have two slots with quality scores $q_1 = 1, q_2 = 0.5$, and three buyers with values $v_1 = 10,\ v_2 = 8,\ v_3 = 2$, and suppose they all bid their values. Then buyer 1 is allocated slot 1, and they generate a value of $v_1 \cdot q_1 = 10$, buyer 2 is allocated slot 2 and they generate a value $v_2 \cdot q_2 = 4$, and buyer 3 gets nothing.*

12.2.1 Generalized Second-Price Auctions

The *generalized second-price* (GSP) auction sells the S slots as follows: First, we allocate via bid maximization as described above. If the user clicks on ad $i \leq S$, then advertiser i is charged the next-highest bid b_{i+1}. GSP generalizes second-price auctions in the sense that if $S = 1$ then this auction format is equivalent to the standard second-price auction (if we take expected values in lieu of the pay-per-click model). However, this is a fairly superficial generalization, since GSP turns out to lose the core property of the second-price auction: truthfulness!

In particular, consider Example 4 again. With GSP prices, buyer 1 gets utility $q_1(v_1 - v_2) = 2$ when everyone bids truthfully. If buyer 1 instead bids some value between 2 and 8, then they get utility $q_2(v_1 - v_3) = 4$. Thus, buyer 1 is better off misreporting. More generally, it turns out that the GSP auction can have several pure-Nash equilibria, and some of these lead to allocations that are not welfare-maximizing. Consider the following bid vector for Example 4, $b = (4, 8, 2)$. Buyer 1 gets utility $0.5(10 - 2) = 4$ (whereas they'd get utility 2 for bidding above 8). Buyer 2 gets utility $1(8 - 4) = 4$ (whereas they'd get utility $0.5(8 - 2) = 3$ for bidding below 4). Buyer 3 is priced out.

12.2.2 VCG for Position Auctions

The second pricing rule we will consider is the VCG rule. Recall that VCG computes the welfare-maximizing allocation (assuming truthful bids), and then charges buyer i their externality (i.e. how much the presence of buyer i decreases the social welfare across the remaining agents).

Let W^S_{-i} be the social welfare achieved by buyers $[n] \setminus i$ if we maximize welfare across only those buyers, and let W^{S-i}_{-i} be the social welfare of $[n] \setminus i$ if we maximize welfare using all slots except slot i. Buyer i gets charged their externality, which is as follows:

$$W^S_{-i} - W^{S-i}_{-i} = \sum_{k \in \{i+1,\dots,S+1\}} b_k \cdot q_{k-1} - \sum_{k \in \{i+1,\dots,S\}} b_k \cdot q_k \tag{12.1}$$

$$= \sum_{k \in \{i+1,\dots,S+1\}} b_k \cdot (q_{k-1} - q_k) \tag{12.2}$$

We already saw a sketch of the fact that VCG is truthful in Chapter 3, but here we show the result specifically for the position auction setting, where the proof is nice and short.

Theorem 24. *The VCG auction for position auctions is truthful.*

Proof. Suppose again that buyer bids are sorted, with buyer i winning slot i when bidding truthfully. Now suppose buyer i misreports and gets slot k instead. Now we want to show that bidding truthfully maximizes utility, which means:

$$q_i \cdot v_i - [W^S_{-i} - W^{S-i}_{-i}] \geq q_k \cdot v_i - [W^S_{-i} - W^{S-k}_{-i}].$$

Simplifying this expression gives

$$q_i \cdot v_i + W^{S-i}_{-i} \geq q_k \cdot v_i + W^{S-k}_{-i}.$$

Now we see that both the right-hand and left-hand sides correspond to social welfare under two different allocations (where we treat bids from other buyers as their true value). The left-hand side is social welfare when i bids truthfully, while the right-hand side is social welfare when i misreports in a way that gives them slot k. Given that VCG picked the left-hand side, and VCG allocates via welfare maximization, the left-hand side must be larger. □

12.3 Historical Notes

An early version of the GSP auction was introduced in the early internet search days at Overture, which was an innovator in sponsored search advertising, and they were later acquired by Yahoo, which used this rule as well. Google then started using the more modern version of GSP. From an academic perspective, the GSP rule and position auctions in general started to be studied by Varian [128] and Edelman et al. [53], motivated by its use in practice. An interesting historical perspective on why VCG was not chosen is discussed by Varian and Harris [129] who worked at Google at the time. The primary reasons are essentially inertial: a lot of engineering work was already going into GSP, and advertisers had gotten used to bidding in GSP. A major concern would be that they would need to raise their bids in VCG due to its truthfulness, which might be hard to explain to them given their existing experience with GSP. Facebook notably uses VCG rather than GSP [129], unlike the prior internet companies.

Chapter 13

Auctions with Budgets

13.1 Introduction

The previous chapter introduced a new aspect to auctions associated with internet advertising auctions: the multiple-slot issue. This chapter studies a second major practical aspect of internet advertising auctions: budgets. In these auctions, a large fraction of advertisers specify a budget constraint that must hold in aggregate across all of the payments made by the advertiser. Because these budget constraints are applied across all of the auctions that an advertiser participates in, they couple the auctions together, and force us to consider the aggregate incentives across auctions. This is in contrast to all of our previous auction results, which studied a single auction in isolation. Notably, these budgets constraints break the incentive compatibility of the second-price auction; for an advertiser with a budget constraint, it is not necessarily optimal to bid their true value in each auction!

13.2 Auctions Markets

Throughout the rest of this chapter, we will consider settings where each individual auction is a single-item auction, using either first or second-price rules. This is of course a simplification: in practice each individual auction would be more complicated (e.g. a position auction), but even just for single-item individual auctions it turns out that there are a lot of interesting problems.

In this setting we have n buyers and m goods. Buyer i has value v_{ij} for good j, and each buyer has some budget B_i. Each good j will be sold via sealed-bid auction, using either first or second-price. We assume that for all buyers i, there exists some item j such that $v_{ij} > 0$, and similarly for all j there exists i such that $v_{ij} > 0$. Let $x \in \mathbb{R}^{n \times m}$ be an allocation of items to buyers, with associated prices $p \in \mathbb{R}^m$. The utility that a buyer i derives from this allocation is

$$u_i(x_i, p) = \begin{cases} \langle v_i, x_i \rangle - \langle p, x_i \rangle & \text{if } \langle p, x_i \rangle \le B_i \\ -\infty & \text{otherwise} \end{cases}.$$

We call this setting an *auction market.* If second-price auctions are used then we call it a second-price auction market, and conversely we call it a first-price auction market if first-price auctions are used.

13.3 Second-Price Auction Markets

In Chapter 3 we saw that the second-price auction is strategyproof. However, this relied on there being a single auction, and no budgets. It's easy to construct an example showing that this is no longer true in second-price auction markets. Consider a market with two buyers and two items, with valuations $v_1 = (100, 100), v_2 = (1, 1)$ and budgets $B_1 = B_2 = 1$. If both buyers submit their true valuations then buyer 1 wins both items, pays 2, and gets $-\infty$ utility.

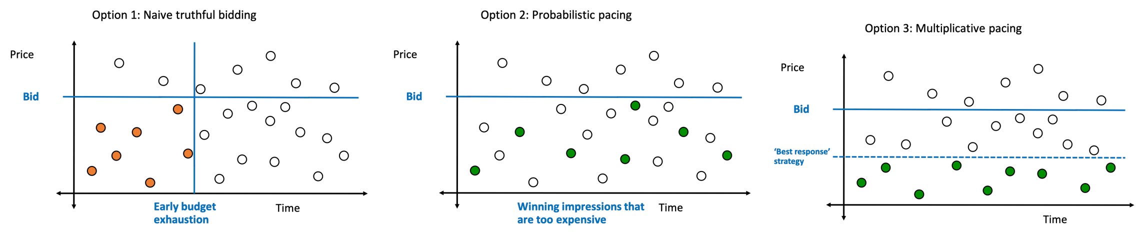


Figure 13.1: Comparison of pacing methods. Left: no pacing, middle: probabilistic pacing, right: multiplicative pacing.

Instead, each buyer needs to somehow smooth out their spending across auctions. For large-scale Internet auctions this is typically achieved via some sort of *pacing rule*. Here we will mention two that have been used in practice:

1. *Probabilistic pacing*: each buyer i is given a parameter $\alpha_i \in [0,1]$ denoting the probability that they should participate in each auction. For each auction j, an independent coin is flipped which comes up heads with probability α_i, and if it comes up heads then the buyer submits a bid $b_{ij} = v_{ij}$ to that auction.

2. *Multiplicative pacing*: each buyer i is given a parameter $\alpha_i \in [0,1]$, which acts as a scalar multiplier on their truthful bids. In particular, for each auction j, buyer i submits a bid $b_{ij} = \alpha_i v_{ij}$.

Both methods have been applied in real-life large-scale Internet ad markets.

Figure 13.1 shows a comparison of pacing methods for a simplified setting where time is taken into account. Here we assume that we are considering some buyer i whose value is the same for every item, but other bidders are causing the items to have different prices. On the x-axis we plot time, and on the y-axis we plot the price of each item. On the left is the outcome from naive bidding: the buyer spends their budget much too fast, and ends up running out of budget when there are many high-value items left for them to buy. In practice, many buyers also prefer to smoothly spend their budget throughout the day. In the middle we show probabilistic pacing, where we do get smooth budget expenditure. However, the buyer ends up buying some very expensive item, while missing out on much cheaper items that have the same value to them. Finally, on the right is the result from multiplicative pacing, where the buyer picks an optimal threshold to buy at, and thus buys item optimally in order of bang-per-buck.

In this note we will focus on multiplicative pacing, but see the historical notes section for some references to papers that also consider probabilistic pacing.

The intuition given in Figure 13.1 can be shown to hold more generally when items have different values to the buyer. Generally, it turns out that given a set of bids by all the other bidders, a buyer can always specify a best response by choosing an optimal pacing multiplier:

Proposition 1. *Suppose we allow arbitrary bids in each auction. If we hold all bids for buyers $k \neq i$ fixed, then buyer i has a best response that consists of multiplicatively-paced bids (assuming that if a buyer is tied for winning an auction, they can specify the fraction that they win).*

Proof. Since every other bid is held fixed, we can think of each item as having some price $p_j = \max_{k\neq i} b_{kj}$, which is what i would pay if they bid $b_{ij} \geq b_{kj}$. Now we may sort the items in decreasing order of bang-per-buck $\frac{v_{ij}}{p_j}$. An optimal allocation for i clearly consists of buying items in this order, until they reach some index j such that if they buy every item with index $l < j$ and some fraction x_{ij} of item j, they either spend their whole budget, or j is the first item with $\frac{v_{ij}}{p_j} \geq 1$ (if $\frac{v_{ij}}{p_j} > 1$ then $x_{ij} = 0$). Now set $\alpha_i = \frac{p_j}{v_{ij}}$. With this bid, i gets exactly this optimal allocation: for all items $l \leq j$ (which are the items in the optimal allocation), we have $\alpha_i v_{il} = \frac{p_j}{v_{ij}} v_{il} \geq \frac{p_l}{v_{il}} v_{il} = p_l$. □

The goal will be to find a *pacing equilibrium*:

Definition 6. *A second-price pacing equilibrium (SPPE) is a vector of pacing multipliers $\alpha \in [0,1]^n$, a fractional allocation x_{ij}, and a price vector such that for every buyer i:*

- *For all j, $\sum_i x_{ij} = 1$, and if $x_{ij} > 0$ then i is tied for highest bid on item j.*
- *If $x_{ij} > 0$ then $p_j = \max_{k \neq i} \alpha_k v_{kj}$.*
- *For all i, $\sum_j p_j x_{ij} \leq B_i$. Additionally, if the inequality is strict then $\alpha_i = 1$.*

The first and second conditions of pacing equilibrium simply enforce that the item always goes to winning bids at the second-price rule. The third condition ensures that a buyer is only paced if their budget constraint is binding. It follows (almost) immediately from Proposition 1 that every buyer is best responding in SPPE.

A nice property of SPPE is that it is always guaranteed to exist (this is not immediate from the existence of, say, a Nash equilibrium in a standard game, since an SPPE corresponds to a specific type of pure-strategy Nash equilibrium):

Theorem 25. *An SPPE of a pacing game is always guaranteed to exist.*

We won't cover the whole proof here, but we will state the main ingredients, which are useful to know more generally.

- First, a smoothed pacing game is constructed. In the smoothed game, the allocation is smoothed out among all bids that are within ϵ of the maximum bid, thus making the allocation a deterministic function of the pacing multipliers α. Several other smooth approximations are also introduced to deal with other discontinuities. In the end, a game is obtained, where each player simply has as their action space the interval $[0, 1]$ and utilities are nice continuous and quasi-concave functions.
- Secondly, the following fixed-point theorem is invoked to guarantee existence of a pure-strategy Nash equilibrium in the smoothed game.

  Theorem 26. *Consider a game with n players, strategy space A_i, and utility function $u_i(a_i, a_{-i})$. If the following conditions are satisfied:*

  - *A_i is convex and compact for all i*
  - *$u_i(s_i, \cdot)$ is continuous in s_{-i}*
  - *$u_i(\cdot, s_{-i})$ is continuous and quasi-concave in s_i (quasi-concavity of a function $f(x)$ means that for all x, y and $\lambda \in [0, 1]$ it holds that $f(\lambda x + (1 - \lambda)y) \geq \min(f(x), f(y))$)*

  then a pure-strategy Nash equilibrium exists.
- Finally, the limit point of smoothed games as the smoothing factor ϵ tends to zero is shown to yield an equilibrium in the original pacing problem.

Unfortunately, while SPPE is guaranteed to exist, it turns out that sometimes there are several SPPE, and they can have large differences in revenue, social welfare, and so on. An example is shown in Figure 13.2. In practice this means that we might need to worry about whether we are in a good and fair equilibrium.

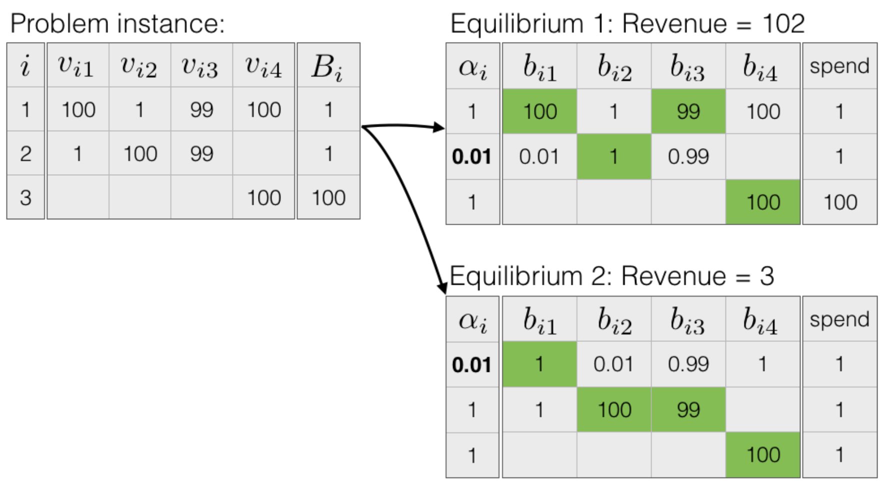
Another positive property of SPPE is that every SPPE is also a market equilibrium, if we consider a market equilibrium setting where each buyer has a quasi-linear demand function that respects the total supply as follows:

$$D_i(p) = \text{argmax}_{0 \leq x_i \leq 1} \langle v_i - p, x_i \rangle \text{ s.t. } \langle p, x_i \rangle \leq B_i.$$

This follows immediately by simply using the allocation x and prices p from the SPPE as a market equilibrium. Proposition 1 tells us that $x_i \in D_i(p)$, and the market clears by definition of SPPE. This means that SPPE has a number of nice properties such as no envy and Pareto optimality (although Pareto optimality requires considering the seller as an agent too).

Finally we turn to the question of computing an SPPE. Unfortunately the news there is bad. It was shown recently that computing an SPPE is a PPAD-complete problem. This means that there exists a polynomial-time reduction between the problem of computing a Nash equilibrium in a general-sum game and that of computing an SPPE, and thus the two problems are equally hard, from the perspective of computing a solution in polynomial time. Moreover, it was also shown that we cannot hope for iterative methods to efficiently compute an approximate SPPE. Beyond merely computing *any* SPPE, we could also try to find one that maximizes revenue or social welfare. This problem turns out to be NP complete.

There is a mixed-integer program for computing SPPE, but unfortunately it is not very scalable.



Problem instance:

i	v_{i1}	v_{i2}	v_{i3}	v_{i4}	B_i
1	100	1	99	100	1
2	1	100	99		1
3				100	100

Equilibrium 1: Revenue = 102

α_i	b_{i1}	b_{i2}	b_{i3}	b_{i4}	spend
1	100	1	99	100	1
0.01	0.01	1	0.99		1
1				100	100

Equilibrium 2: Revenue = 3

α_i	b_{i1}	b_{i2}	b_{i3}	b_{i4}	spend
0.01	1	0.01	0.99	1	1
1	1	100	99		1
1				100	1

Figure 13.2: Multiplicity of SPPE. On the left is shown a problem instance, and on the right is shown two possible second-price pacing equilibria.

13.4 First-Price Auction Markets

Next we consider what happens if we instead sell each item by first-price auction as part of an auction market.

First we start by defining what we call *budget-feasible pacing multipliers*. Intuitive, this is simply a set of pacing multipliers such that everything is allocated according to first-price auction, and everybody is within budget.

Definition 7. *A set of* budget-feasible pacing multipliers *(BFPM) is a vector of pacing multipliers $\alpha \in [0,1]^n$ and a fractional allocation x_{ij} such that for every buyer i:*

- *Prices are defined to be $p_j = \max_k \alpha_k v_{kj}$.*
- *For all $j, \sum_i x_{ij} = 1$, and if $x_{ij} > 0$ then i is tied for highest bid on item j.*
- *For all i, $\sum_j p_j x_{ij} \le B_i$.*

Again, the goal will be to find a *pacing equilibrium*. This is simply a BFPM that satisfied the complementarity condition on the budget constraint and pacing multiplier.

Definition 8. *A* first-price pacing equilibrium *(FPPE) is a BFPM (α, x) such that for every buyer i:*

- *For all i, if $\sum_j p_j x_{ij} < B_i$ then $\alpha_i = 1$.*

Notably, the only difference to SPPE is the pricing condition, which now uses first price.

A very nice property of the first-price setting is that BFPMs satisfy a monotonicity condition: if (α', x') and (α'', x'') are both BFPM, then the pacing vector $\alpha = \max(\alpha', \alpha'')$ (where the max is taken componentwise) is also a BFPM. The associated allocation is that for each item j, we first identify whether the highest bid comes from α' or α'', and use the corresponding allocation of j (breaking ties towards α').

Intuitively, the reason that (α, x) is also BFPM is that for every buyer i, their bids are the same as in one of the two previous BFPMs (say (α', x') WLOG.), and so the prices they pay are the same as in (α', x'). Furthermore, since every other buyer is bidding at least as much as in (α', x'), they win weakly less of each item (using the tie-breaking scheme described above). Since (α', x') satisfied budgets, (α, x) must also satisfy budgets. The remaining conditions are easily checked.

In addition to componentwise maximality, there is also a *maximal* BFPM (α, x) (there could be multiple x compatible with α) such that $\alpha \ge \alpha'$ for all α' that are part of any BFPM. Consider $\alpha_i^* = \sup\{\alpha_i | \alpha \text{ is part of a BFPM}\}$. For any ϵ and i, we know that there must exist a BFPM such that $\alpha_i > \alpha_i^* - \epsilon$. For a fixed ϵ we can take componentwise maxima to conclude that there exists $(\alpha^\epsilon, x^\epsilon)$ that is a BFPM. This yields a sequence $\{(\alpha^\epsilon, x^\epsilon)\}$ as $\epsilon \to 0$. Since the space of both α and x is compact, the sequence has a limit point (α^*, x^*). By continuity (α^*, x^*) is a BFPM.

We can use this maximality to show existence and uniqueness (of multipliers) of FPPE:

Theorem 27. *An FPPE always exists and the set of pacing multipliers $\{\alpha\}$ that are part of an FPPE is a singleton.*

Proof. Here we give a high-level proof, a more explicit proof can be found in the paper listed in the notes.

Consider the component-wise maximal α and an associated allocation x such that they form a BFPM.

Since α, x is a BFPM, we only need to check that it has no unnecessarily paced bidders. Suppose some buyer i is spending strictly less than B_i and $\alpha_i < 1$. If i is not tied for any items, then we can increase α_i for some sufficiently small ϵ and retain budget feasibility, contradicting the maximality of α. If i is tied for some item, consider the set $N(i)$ of all bidders tied with i. Now take the transitive closure of this set by repeatedly adding any bidder that is tied with any bidder in $N(i)$. We can now redistribute all the tied items among bidders in $N(i)$ such that no bidder in $N(i)$ is budget constrained (this can be done by slightly increasing i's share of every item they are tied on, then slightly increasing the share of every other buyer in $N(i)$ who is now below budget, and so on). But now there must exist some small enough $\delta > 0$ such that we can increase the pacing multiplier of every bidder in $N(i)$ by δ while retaining budget feasibility and creating no new ties. This contradicts α being maximal. We get that there can be no unnecessarily paced bidders under α

Finally, to show uniqueness, consider any alternative BFPM α', x'. Consider the set I of buyers such that $\alpha'_i < \alpha$; Since $\alpha \geq \alpha'$ and $\alpha \neq \alpha'$ this set must have size at least one. Since all buyers in I were spending less than their budget under α, and their collective spending strictly decreased, at least one buyer in I must not be spending their whole budget. But $\alpha'_i < \alpha_i \leq 1$ for all $i \in I$, so that buyer must be unnecessarily paced. □

13.4.1 Sensitivity

FPPE enjoys several nice monotonicity and sensitivity properties that SPPE does not. Several of these follow from the maximality property of FPPE: the unique FPPE multipliers α are such that $\alpha \geq \alpha'$ for any other BFPM (α', x').

The following are all guaranteed to weakly increase revenue of the FPPE:

1. Adding a bidder i: the old FPPE (α, x) is still BFPM by setting $\alpha_i = 0, x_i = 0$. By α monotonicity prices increase weakly.

2. Adding an item: The new FPPE α' satisfies $\alpha' \leq \alpha$ (for contradiction, consider the set of bidders whose multipliers increased, since they win weakly more and prices went up, somebody must break their budget). Now consider the bidders such that $\alpha'_i < \alpha_i$. Those bidders spend their whole budget by the FPPE "no unnecessary pacing" condition. For bidders such that $\alpha'_i = \alpha_i$, they pay the same as before, and win weakly more.

3. Increasing a bidder i's budget: the old FPPE (α, x) is still BFPM, so this follows by α maximality.

It is also possible to show that revenue enjoys a Lipschitz property: increasing a single buyer's budget by Δ increases revenue by at most Δ. Similarly, social welfare can be bounded in terms of Δ, though multiplicatively, and it does not satisfy monotonicity.

13.4.2 Convex Program

Next we consider how to compute an FPPE. This turns out to be easier than for SPPE. This is due to a direct relationship between FPPE and market equilibrium: FPPE solutions are exactly the set of solutions to the *quasi-linear* variant of the Eisenberg-Gale convex program for computing a market equilibrium:

$$\begin{aligned} \max_{x\geq 0,\delta\geq 0,u} \quad & \sum_i B_i \log(u_i) - \delta_i \\ u_i \leq \sum_j x_{ij}v_{ij} + \delta_i, \forall i & \qquad (13.1) \\ \sum_i x_{ij} \leq 1, \forall j & \qquad (13.2) \end{aligned}$$

$$\begin{aligned} \min_{p\geq 0,\beta\geq 0} \quad & \sum_j p_j - \sum_i B_i \log(\beta_i) \\ \forall i, p_j \geq v_{ij}\beta_i & \\ \beta_i \leq 1 & \end{aligned} \qquad (13.3)$$

On the left is shown the primal convex program, and on the right is shown the dual convex program. The variables x_{ij} denote the amount of item j that bidder i wins. The leftover budget is denoted by δ_i, it arises from the dual program: it is the primal variable for the dual constraint $\beta_i \leq 1$, which constrains bidder i to paying at most a price-per-utility rate of 1.

The dual variables β_i, p_j correspond to constraints (13.1) and (13.2), respectively. They can be interpreted as follows: β_i is the inverse bang-per-buck: $\min_{j s.t. x_{ij}>0} \frac{p_j}{v_{ij}}$ for buyer i, and p_j is the price of good j.

We may use the following basic fact from convex optimization to conclude that strong duality holds and get optimality conditions:

Theorem 28. *Consider a convex program and its dual*

$$\begin{aligned} \min_x \quad & f(x) \\ g_i(x) \leq 0,\ \forall i & \\ x \geq 0 & \end{aligned} \qquad (13.4)$$

$$\begin{aligned} \max_{\lambda\geq 0} \quad & q(\lambda) \\ q(\lambda) := \min_{x\geq 0} L(x,\lambda) & \\ L(x,\lambda) := f(x) + \sum_i \lambda_i g_i(x) & \end{aligned} \qquad (13.5)$$

with Lagrange multipliers λ_i for each constraint i. Assume that the following Slater constraint qualification *is satisfied: there exists some $x \geq 0$ such that $g_i(x) < 0$ for all i. If* (13.4) *has a finite optimal value f^* then* (13.5) *has a finite optimal value q^* and $f^* = q^*$. Furthermore a solution pair x^*, λ^* is optimal if and only if the following Karush-Kuhn-Tucker (KKT) conditions hold:*

- (primal feasibility) x^* *is a feasible solution of* (13.4)
- (dual feasibility) $\lambda^* \geq 0$
- (complementary slackness) $\lambda_i^* g_i(x^*) = 0$ *for all i*
- (stationarity) $x^* \in \operatorname{argmin}_{x\geq 0} L(x, \lambda^*)$

We can use the strong duality theorem above, and in particular the KKT conditions, to show that FPPE and EG are equivalent.

Informally, the correspondence between FPPE and solutions to the convex program follows because β_i specifies a single price-per-utility rate per bidder which exactly yields the pacing multiplier $\alpha_i = \beta_i$. Complementary slackness then guarantees that if $p_j > v_{ij}\beta_i$ then $x_{ij} = 0$, so any item allocated to i has exactly rate β_i. Similarly, complementary slackness on $\beta_i \leq 1$ and the associated primal variable δ_i guarantees that bidder i is only paced if they spend their whole budget.

Theorem 29. *An optimal solution to the quasi-linear Eisenberg-Gale convex program corresponds to an FPPE with pacing multiplier $\alpha_i = \beta_i$ and allocation x_{ij}, and vice versa.*

Proof. Clearly the quasi-linear Eisenberg-Gale convex program satisfies the Slater constraint qualification: we may use the proportional allocation where every buyers gets $\frac{1}{n}$ of every item to see this. Thus the optimal solution must satisfy the following KKT conditions:

1. $\frac{B_i}{u_i} = \beta_i \Leftrightarrow u_i = \frac{B_i}{\beta_i}$
2. $\beta_i \leq 1$
3. $\beta_i \leq \frac{p_j}{v_{ij}}$
4. $x_{ij}, \delta_i, \beta_i, p_j \geq 0$
5. $p_j > 0 \Rightarrow \sum_i x_{ij} = 1$
6. $\delta_i > 0 \Rightarrow \beta_i = 1$
7. $x_{ij} > 0 \Rightarrow \beta_i = \frac{p_j}{v_{ij}}$

It is easy to see that x_{ij} is a valid allocation: the primal program has the exact packing constraints. Budgets are also satisfied (here we may assume $u_i > 0$ since otherwise budgets are satisfied since the bidder wins no items): by KKT condition 1 and KKT condition 7 we have that for any item j that bidder i is allocated part of:

$$\frac{B_i}{u_i} = \frac{p_j}{v_{ij}} \Rightarrow \frac{B_i v_{ij} x_{ij}}{u_i} = p_j x_{ij}$$

If $\delta_i = 0$ then summing over all j gives

$$\sum_j p_j x_{ij} = B_i \frac{\sum_j v_{ij} x_{ij}}{u_i} = B_i$$

This part of the budget argument is exactly the same as for the standard Eisenberg-Gale proof [103]. Note that (13.1) always holds exactly since the objective is strictly increasing in u_i. Thus $\delta_i = 0$ denotes full budget expenditure. If $\delta_i > 0$ then (13.1) implies that $u_i > \sum_j v_{ij} x_{ij}$ which gives:

$$\sum_j p_j x_{ij} = B_i \frac{\sum_j v_{ij} x_{ij}}{u_i} < B_i$$

This shows that $\delta_i > 0$ denotes some leftover budget.

If bidder i is winning some of item j ($x_{ij} > 0$) then KKT condition 7 implies that the price on item j is $\alpha_i v_{ij}$, so bidder i is paying their bid as is necessary in a first-price auction. Bidder i is also guaranteed to be among the highest bids for item j: KKT conditions 7 and 3 guarantee $\alpha_i v_{ij} = p_j \geq \alpha_{i'} v_{i'j}$ for all i'.

Finally each bidder either spends their entire budget or is unpaced: KKT condition 6 says that if $\delta_i > 0$ (that is, some budget is leftover) then $\beta_i = \alpha_i = 1$, so the bidder is unpaced.

Now we show that any FPPE satisfies the KKT conditions for EG. We set $\beta_i = \alpha_i$ and use the allocation x from the FPPE. We set $\delta_i = 0$ if $\alpha < 1$, otherwise we set it to $B_i - \sum_j x_{ij} v_{ij}$. We set u_i equal to the utility of each bidder. KKT condition 1 is satisfied since each bidder either gets a utility rate of 1 if they are unpaced and so $u_i = B_i$ or their utility rate is α_i so they spend their entire budget for utility B_i/α_i. KKT condition 2 is satisfies since $\alpha_i \in [0, 1]$. KKT condition 3 is satisfied since each item bidder i wins has price-per-utility $\alpha_i = \frac{p_j}{v_{ij}} = \beta_i$, and every other item has a higher price-per-utility. KKT conditions (4) and (5) are trivially satisfied by the definition of FPPE. KKT condition 6 is satisfied by our solution construction. KKT condition 7 is satisfied because a bidder i being allocated any amount of item j means that they have a winning bid, and their bid is equal to $v_{ij}\alpha_i$. □

It follows that an FPPE can be computed in polynomial time, and that we can apply various first-order methods to compute large-scale FPPE.

13.5 In what sense are we in equilibrium?

We introduced the pacing equilibrium using terminology similar to how we previously discussed game-theoretic equilibria such as Nash equilibria. Yet, it is useful to take a moment to consider what the actual equilibrium properties that we are getting under pacing equilibria are. When we defied pacing equilibria, we asked for a certain complementarity condition on the pacing multipliers, the "no unecceasry pacing" condition. This condition is not a game-theoretic equilibrium condition, but rather a condition on the budget-management algorithms that the buyers are using. In particular, it is a condition that an online learning algorithm on the Lagrange multiplier of the budget constraint would try to maintain. Now, assuming that you are in a static environment where at each time step, the m items from the pacing model show up, then a pacing equilibrium would be stable, in the sense that if everyone bids according to the computed multipliers, and tied goods are split according to the fractional amounts from the equilibrium, then "no unnecessary pacing" is satisfied, so the budget-management algorithms won't change their pacing multipliers. In this sense we are in equilibrium.

However, from the perspective of the buyers, they may or may not be best responding to each other. In the context of *second-price* pacing equilibrium, it is possible to show that a pacing equilibrium is a pure Nash equilibrium of a game where each buyer is choosing their pacing multiplier, and observing their quasi-linear utility (with $-\infty$ utility for breaking the budget). Moreover, in the second-price setting, if we fix the bids of every other buyer, then a pacing multiplier α_i that satisfies no unnecessary pacing is actually a best response over the set of all possible ways to bid in each individual auction. In the case of *first-price* pacing equilibrium, we do not have this property. In FPPE, a buyer might wish to shade their own price. In that case, FPPE should be thought of only as a budget-management equilibrium among the algorithmic proxy bidders that control budget expenditure. Secondly, due to this shading, the values v_{ij} that we took as input to the FPPE problem should probably be thought of as the *bids* of the buyers, which would generally be lower than their true values.

13.6 Conclusion

There are interesting differences in the properties satisfied by SPPE and FPPE. We summarize them quickly here (these are all covered in the literature noted in the Historical Notes):

- FPPE is unique (this can be shown from the convex program, or directly from the monotonicity property of BFPM), SPPE is not
- FPPE can be computed in polynomial time, computing an SPPE is a PPAD-complete problem
- FPPE is less sensitive to perturbation (e.g. revenue increases smoothly as budgets are increased)
- SPPE corresponds to a pure-strategy Nash equilibrium, and thus buyers are best responding to each other
- Both correspond to different market equilibria (but SPPE requires buyer demands to be "supply aware")
- Neither of them are strategyproof
- Due to the market equilibrium connection, both can be shown strategyproof in an appropriate "large market" sense

FPPE and SPPE have also been studied experimentally, both via random instances, as well as instances generated from real ad auction data. The most interesting takeaways from those experiments are:

- In practice SPPE multiplicity seems to be very rare
- Manipulation is hard in both SPPE and FPPE if you can only lie about your value-per-click
- FPPE dominates SPPE on revenue
- Social welfare can be higher in either FPPE or SPPE. Experimentally it seems to largely be a toss-up on which solution concept has higher social welfare.

13.7 Historical Notes

The multiplicative pacing equilibrium results shown in this lecture note were developed by Conitzer et al. [45] for SP auction markets, and Conitzer et al. [46] for FP auction markets. Another strand of literature has studied models where items arrive stochastically and valuations are then drawn independently. Balseiro et al. [3] show existence of pacing equilibrium for multiplicative pacing as well as several other pacing rules for such a setting; they also give a very interesting comparison of revenue and social welfare properties of the various pacing option in the unique symmetric equilibrium of their setting. Most notably, multiplicative pacing achieves strong social welfare properties, while probabilistic pacing achieves higher revenue properties. Balseiro, Besbes, and Weintraub [5] show that when bidders get to select their bids individually, multiplicative pacing equilibrium arises naturally via Lagrangian duality on the budget constraint, under a fluid-based mean-field market model. The PPAD-completeness of computing an SPPE was given by Chen, Kroer, and Kumar [36]

The quasi-linear variant of Eisenberg-Gale was given by Chen, Ye, and Zhang [34] and independently by Cole et al. [42] (an unpublished note from one of the authors in Cole et al. [42] was in existence around a decade before the publication of Cole et al. [42]). Theorem 28 is a specialization to the FPPE setting. In reality much stronger statements can be made: For a more general statement of the strong duality theorem and KKT conditions used here, see Bertsekas, Nedic, and Ozdaglar [11] Proposition 6.4.4. The KKT conditions can be significantly generalized beyond convex programming.

The fixed-point theorem that is invoked to guarantee existence of a pure-strategy Nash equilibrium in the smoothed game is by Debreu [49], Glicksberg [71], and Fan [56].

Chapter 14

Online Budget Management

14.1 Introduction

In the last lecture note we studied auctions with budgets and repeated auctions. However, we ignored one important aspect: time. In this lecture note we consider an auction market setting where a buyer is trying to adaptively pace their bids over time. The goal is to hit the "right" pacing multiplier as before, but each bidder has to learn that multiplier as the market plays out. We'll see how we can approach this problem using ideas from regret minimization.

14.2 Dynamic Auctions Markets

In this setting we have n buyers who repeatedly participate in second-price auctions. At each time period $t = 1, \dots, T$ a single second-price auction is run. At time t, each bidder samples a valuation v_{it} independently from a cumulative distribution function F_i which is assumed to be absolutely continuous and with bounded density f_i whose support is $[0, \bar{v}_i]$. As usual, we assume that each buyer has some budget B_i that they should satisfy, and we denote by $\rho_i = B_i/T$ the per-period target expenditure; we assume $\rho_i \leq \bar{v}_i$. We may think of each buyer as being characterized by a type $\theta_i = (F_i, \rho_i)$.

At each time period t buyer i observes their valuation v_{it} and then submits a bid b_{it}. We will use $d_{it} = \max_{k \neq i} b_{it}$ to denote the highest bid other than that of i. As before the utility of an buyer is quasi-linear and thus if they win auction t they get utility $v_{it} - d_{it}$. We may write the utility using an indicator variable as $u_{it} = \mathbb{1}\{d_{it} \leq b_{it}\}(v_{it} - d_{it})$, and the expenditure $z_{it} = \mathbb{1}\{d_{i,t} \leq b_{it}\}d_{it}$.

It is assumed that each buyer has no information on the valuation distributions, including their own. Instead, they just know their own target expenditure rate ρ_i and the total number of time periods T. Buyers also do not know how many other buyers are in the market.

At time t, buyer i knows the *history* $(v_{i\tau}, b_{i\tau}, z_{i\tau}, u_{i\tau})_{\tau=1}^{t-1}$ of own values, bids, payments, and utilities. Furthermore, they know their current value v_{it}. Based on this history, they choose a bid b_{it}. We will say that a bidding strategy for buyer i is a sequence of mappings $\beta = (\beta_1, \beta_2, \dots)$ where β_t maps the current history to a bid (potentially in randomized fashion). The strategy β is budget feasible if the bids b_{it}^{β} generated by β are such that

$$\sum_{t=1}^{T} \mathbb{1}\{d_{it} \leq b_{it}^{\beta}\}d_{it} \leq B_i$$

under any vector of highest competitor bids d_i.

For a given realization of values $v_i = v_{i1}, \dots, v_{iT}$ and highest competitor bids d_i we denote the expected value of a strategy β as

$$\pi_i^{\beta}(v_i, d_i) = \mathbb{E}\left[\sum_{t=1}^{T} \mathbb{1}\{d_{it} \leq b_{it}^{\beta}\}(v_{it} - d_{it})\right],$$

where the expectation is taken with respect to randomness in β.

We would like to compare our outcome to the *hindsight optimal* strategy. We denote the expected value of that strategy as

$$\begin{aligned} \pi_i^H(v_i, d_i) := \max_{x_i \in \{0,1\}^T} \quad & \sum_{t=1}^{T} x_{it}(v_{it} - d_{it}) \\ s.t. \quad & \sum_{t=1}^{T} x_{it} d_{it} \leq B_i \end{aligned} \tag{14.1}$$

The hindsight-optimal strategy has a simple structure: we simply choose the optimal subset of items to win while satisfying our budget constraint. In the case where the budget constraint is binding, this is a knapsack problem.

Ideally we would like to choose a strategy such that π_i^β approaches π_i^H. However, this turns out not to be possible. We will use the idea of asymptotic γ-competitiveness to see this. Formally, β is asymptotic γ-competitive if

$$\limsup_{\substack{T\to\infty, \\ B_i=\rho_i T}} \sup_{\substack{v_i \in [0,\bar{v}_i]^T, \\ d_i \in \mathbb{R}_+^T}} \frac{1}{T}\left(\pi_i^H(v_i, d_i) - \gamma \pi_i^\beta(v_i, d_i)\right) \leq 0$$

Intuitively, the condition says that asymptotically, β should achieve at least $1/\gamma$ of the hindsight-optimal expected value.

For any $\gamma < \bar{v}_i/\rho_i$, asymptotic γ-competitiveness turns out to be impossible to achieve. Thus, if our target expenditure ρ_i is much smaller than our maximum possible valuation, we cannot expect to do anywhere near as well as the hindsight-optimal strategy.

The general proof is quite involved, but the high-level idea is not too complicated. Here we show the construction for $\bar{v}_i = 1, \rho_i = 1/2$, and thus the claim is that $\gamma < \bar{v}_i/\rho_i = 2$ is unachievable. The impossibility is via a worst-case instance. In this instance, the highest other bid comes from one of the two following sequences:

$$\begin{aligned} d^1 &= (d_{high}, \ldots, d_{high}, \bar{v}_i, \ \ldots\ldots \ , \bar{v}_i) \\ d^2 &= (d_{high}, \ldots, d_{high}, d_{low}, \ldots, d_{low}), \end{aligned}$$

for $\bar{v}_i \geq d_{high} > d_{low} > 0$. The general idea behind this construction is that in the sequence d^1, buyer i must buy many of the expensive items in order to maximize their utility, since they receive zero utility for winning items with price $\bar{v}_i$. However, in the sequence d^2, buyer i must save money so that they can buy the cheaper items priced at d_{low}.

For the case we consider here, there are $T/2$ of each type of highest other bid (assume T is even for convenience). Now, we may set $d_{high} = 2\rho_i - \epsilon$ and $d_{low} = 2\rho_i - k\epsilon$, where ϵ and k are constants that can be tuned. For sufficiently small ϵ, i can only afford to buy $T/2$ items total, no matter the combination of items. Furthermore, buying an item at price d_{low} yields k times as much utility as buying an item at d_{high}.

Now, in order to achieve at least half of the optimal utility under d^1, buyer i must purchase at least $T/4$ of the items priced at d_{high}. Since they don't know whether d^1 or d^2 occurred until after deciding whether to buy at least $T/4$ of the d_{high} items, this must also occur under d^2. But then buyer i can at most afford to buy $T/4$ of the items priced at d_{low} when they find themselves in the d^2 case. Now for any $\gamma < 2$, we can pick k and ϵ such that achieving $\gamma\pi_i^H$ requires buying at least $T/4 + 1$ of the d_{low} items.

It follows that we cannot hope to design an online algorithm that competes with $\gamma\pi_i^H$ for $\gamma < \bar{v}_i/\rho_i$. However, it turns out that a subgradient descent algorithm can achieve exactly $\gamma = \bar{v}_i/\rho_i$

14.3 Adaptive Pacing Strategy

The idea is to construct a pacing multiplier $\alpha_i = \frac{1}{1+\mu}$ by running a subgradient descent scheme on the value for μ that allows i to smoothly spend their budget across the T time periods.

The algorithm takes as input a stepsize $\epsilon_i > 0$ and some initial value $\mu_1 \in [0, \bar{\mu}_i]$ (where $\bar{\mu}_i$ is some upper bound on how large μ needs to be). We use $P_{[0,\bar{\mu}_i]}$ to denote projection onto the interval $[0, \bar{\mu}_i]$. The algorithm, which we call APS, proceeds as follows

- Initialize the remaining budget at $\tilde{B}_{i1} = B_i$
- For every time period $t = 1, \dots, T$:
  1. Observe v_{it}, construct a paced bid $b_{it} = \min(\frac{v_{it}}{1+\mu_t}, \tilde{B}_{it})$
  2. Observe spend z_{it}, and update the pacing multiplier:
  $$\mu_{t+1} = P_{[0,\bar{\mu}_i]}(\mu_t - \epsilon_i(\rho_i - z_{it}))$$
  3. Update remaining budget $\tilde{B}_{i,t+1} = \tilde{B}_{it} - z_{it}$

This algorithm is motivated by Lagrangian duality. Consider the following Lagrangian relaxation of the hindsight-optimal optimization problem (14.1):

$$\max_{x \in \{0,1\}^T} \sum_{t=1}^{T} \left[x_{it}(v_{it} - (1-\mu)d_{it}) + \mu\rho_i\right].$$

The optimal solution for the relaxed problem is easy to characterize: we set $x_{it} = 1$ for all t such that $v_{it} \geq (1-\mu)d_{it}$. Importantly, this is achieved by the bid $b_{it} = \frac{v_{it}}{1+\mu}$ that we use in APS.

The Lagrangian dual is the minimization problem

$$\inf_{\mu \geq 0} \sum_{t=1}^{T} \left[(v_{it} - (1-\mu)d_{it})^+ + \mu\rho_i\right], \tag{14.2}$$

where $(\cdot)^+$ denotes thresholding at 0. This dual problem upper bounds π_i^H (but we do not necessarily have strong duality since we did not even start out with a convex primal program). The minimizer of the dual problem yields the strongest possible upper bound on ϕ_i^H, however, solving this requires us to know the entire sequences v_i, d_i. APS approximates this optimal μ by taking a subgradient step on the t'th term of the dual:

$$\partial_\mu \left[(v_{it} - (1-\mu)d_{it})^+ + \mu\rho_i\right] \ni \rho_i - d_{it}\mathbb{1}\{b_{it} \geq d_{it}\} = \rho_i - z_{it}.$$

Thus APS is taking subgradient steps based on the subdifferential of the t'th term of the Lagrangian dual of the hindsight-optimal optimization problem.

The APS algorithm achieves exactly the lower bound we derived earlier, and is thus asymptotically optimal:

Theorem 30. *APS with stepsize $\epsilon_i = O(T^{-1/2})$ is $\frac{\bar{v}_i}{\rho_i}$-asymptotic competitive, and converges at a rate of $O(T^{-1/2})$.*

This result holds under adversarial conditions: for example, the sequence of highest other bids may be as d^1, d^2 in the lower bound. However, in practice we do not necessarily expect the world to be quite this adversarial. In a large-scale ad market, we would typically expect the sequences v_i, d_i to be more stochastic in nature. In a fully stochastic setting with independence, APS turns out to achieve π_i^H asymptotically:

Theorem 31. *Suppose (v_{it}, d_{it}) are sampled independently from stationary, absolutely continuous CDFs with differentiable and bounded densities. Then the expected payoff from APS with stepsize $\epsilon_i = O(T^{-1/2})$ approaches π_i^H asymptotically at a rate of $T^{-1/2}$.*

Theorem 31 shows that if the environment is well-behaved then we can expect much better performance from APS.

14.4 Historical Notes

The material presented here was developed by Balseiro and Gur [4]. Beyond auction markets, the idea of using paced bids based on the Lagrange multiplier μ has been studied in the revenue management literature, see e.g. Talluri and Van Ryzin [121], where it is shown that this scheme is asymptotically optimal as T tends to infinity. There is also recent work on the adaptive bidding problem using multi-armed bandits [65].

Chapter 15

Demographic Fairness

15.1 Introduction

This chapter studies the issue of *demographic fairness*. This is a separate topic from the types of fairness we have studied so far, which was largely focused on individual fairness notions such as envy-freeness and proportionality. Moreover, in the context of ad auctions, those fairness guarantees are with respect to advertisers, since they are the buyers/agents in the market equilibrium model of the ad auction markets. Demographic fairness, on the other hand, is a fairness notion with respect to the users who are being shown the ads. In the context of the Fisher market models we have studied so far, this means that demographic fairness will be a property measured on the item side, since items correspond to ad slots for particular users. Secondly, some demographic fairness notions will be with respect to groups of users, rather than individual users. A serious concern with internet advertising auctions and recommender systems is that the increased ability to target users based on features could lead to harmful effects on subsets of the population, such as gender or race-based biases in the types of ads or content being shown. We will start by looking at a few real-world examples where notions of demographic fairness were observed to be violated. We will then describe some potential ideas for implementing fairness in the context of Fisher markets and first-price ad auctions, but it is important to emphasize that this is an evolving area, and it is not clear that there is a simple answer to the question of how to guarantee certain types of demographic fairness, and moreover there are tradeoffs involved between various notions, as well as between fairness and other objectives such as revenue or welfare.

15.1.1 Age Discrimination in Job Ads

ProPublica reported in 2017 that many companies were using age as part of their targeting criteria for job ads they were placing on Facebook [2]. This included Amazon, Verizon, UPS and Facebook itself. Quoting from the article:

> Verizon placed an ad on Facebook to recruit applicants for a unit focused on financial planning and analysis. The ad showed a smiling, millennial-aged woman seated at a computer and promised that new hires could look forward to a rewarding career in which they would be "more than just a number."
>
> Some relevant numbers were not immediately evident. The promotion was set to run on the Facebook feeds of users 25 to 36 years old who lived in the nation's capital, or had recently visited there, and had demonstrated an interest in finance.

Whether age-based targeting of job ads is illegal was not completely clear, as of 2017 when this article was written. The federal *Age Discrimination in Employment Act* of 1967 prohibits bias against people aged 40 or older both in hiring and employment. Whether the company placing the ad, as well as Facebook, could be held liable for age discrimination was not clear, since the law was written before the internet age, and it was not clear whether the law applied to targeted ads.

15.1.2 Targeting Housing Ads along Racial Boundaries

ProPublica also reported in 2016 on the fact that advertisers had the ability to run ads that exclude certain "ethnic affinities" such as "hispanic affinity" or "african-american affinity" on Facebook [1]. Since Facebook does not ask users about race, these affinity categories are stand-in estimates based on user interests and behavior. On the benign side, these features can be used to test for example how an ad in Spanish versus English will perform in a hispanic population. More generally, it can be used as a tool for advertisers to understand how their products are received by different groups.

However, ProPublica reported that they were able to create a (fake) ad for an event related to first-time home buying, where they could use these categories to exclude various ethnic groups from seeing the ad. When it comes to topics such as housing, the *Fair Housing Act* from 1968 made it illegal

> "to make, print, or publish, or cause to be made, printed, or published any notice, statement, or advertisement, with respect to the sale or rental of a dwelling that indicates any preference, limitation, or discrimination based on race, color, religion, sex, handicap, familial status, or national origin."

In other contexts, such as e.g. traditional newspapers, advertisements are reviewed before being accepted to be shown, in order to ensure that they do not violate these laws. However, in the context of online advertising, the process is much more automated and algorithmic, and the targeting criteria are powerful enough that one has to think carefully about what fairness means and how it can be implemented algorithmically.

For the remainder of the lecture, we will operate under the assumption that we wish to ensure various demographic properties of how ads are shown, for ads that are viewed as "sensitive". Beyond employment and housing, another category of ads that are viewed as sensitive are credit opportunities. Again, existing laws that were created prior to the internet disallow discrimination based on demographic properties in lending.

15.2 Disallowing Targeting

If we wish to prohibit the potential discrimination described above, we could introduce a category of "sensitive ads," where we do not allow age, gender, or racial features to be used as a feature. One might naively think that this would work, but unfortunately there are many ways to perform indirect targeting of these categories. For example, zip code can often be a strong proxy for race, and thus care is needed in order to ensure that we do not allow proxy-based targeting of these sensitive features.

Facebook took such an approach in 2019 [114], based on a settlement with various civil rights organizations. In that approach, they disallow targeting on age, gender, zip code, and "cultural affinities" for what they categorize as sensitive ads. That categorization includes housing, employment, and credit opportunities.

While this approach ensures that a certain type of discrimination cannot occur, it does not necessarily rule out other forms of biases in how ads are serverd.

15.3 Demographic Fairness Measures

We will next study explicit quantifiable measures of fairness. These can potentially be used to audit whether a given ad or system contains biases, or as guiding measures for how to adaptively change the allocation system in order to ensure unbiasedness.

To make things concrete, suppose we have m users, and a single sensitive ad i. We will assume that each user j is associated with a non-sensitive feature vector w_j, and each user also belongs to one of two demographic groups, A or B, which is considered a sensitive attribute; let g_j denote this group. We let G_A and G_B be the set of all indices denoting users in group A or group B, respectively. As usual, we will use $x_{ij} \in [0, 1]$ to denote the probability that the ad i is shown to user j.

Statistical Parity This notion of demographic fairness asks that ad i is shown at an equal rate across the two groups, in the following sense:

$$\frac{1}{|G_A|}\sum_{j\in G_A} x_{ij} = \frac{1}{|G_B|}\sum_{j\in G_B} x_{ij}.$$

This guarantees that, in aggregate, the groups are being shown the ad at an equal rate.

Next, let's see an example of how statistical parity could be broken even though targeting by demographic features is disallow. Suppose that a sensitive ad (say a job ad) wishes to target users in either demographic, and has a value of \$1 per click, with a click-through rate that depends only on w_j and not g_j. Secondly, there's another ad which is not sensitive, which has a value per click of \$2, and click-through rates of 0.1 and 0.6 for groups A and B respectively. Now, the sensitive ad will never be able to win any slots for group B since even with a CTR of 1, their bid will be lower than $0.6 \cdot 2 = 1.2$. As a result, the sensitive ad will be shown only to group A. A concrete example of how this competition-driven form of bias might occur is when the non-sensitive ad is some form of female-focused product such as clothing or make-up.

A potential criticism of this fairness measure is that it does not require the ad to be shown to equally interested users in both groups. Thus, one could for example worry that the ad might end up buying highly relevant slots among one group, and cheap irrelevant slots in the other group in order to satisfy the constraint.

Similar Treatment Similar treatment (ST) asks for an individual-level fairness guarantee: if two users i and k have the same non-sensitive feature vector $w_j = w_k$, then they should be treated similarly regardless of the value of g_j and g_k. A simple version of this principle for ad auctions could be that we require $x_{ij} = x_{ik}$ whenever $w_j = w_k$. However, if the feature space is large, some features are continuous, or we just want this to hold even when users are similar in terms of w_j and w_k, then we need a slightly more complicated constraint. Suppose we have a measure $d(w_j, w_k)$ that measures similarity between feature vectors. Then, ST can be defined as

$$|x_{ij} - x_{ik}| \le d(w_j, w_k).$$

With this definition, we are asking for more than just equality when $w_j = w_k$; instead we also ask that the difference between x_{ij} and x_{ik} should decrease smoothly as the non-sensitive feature vectors get closer to each other, as measured by d.

15.4 Implementing Fairness Measures on a Per-Ad Basis

In this section we highlight some difficulties in applying these fairness notions straightforwardly in ad auction markets. We will focus on statistical parity; similar treatment seems even more difficult to implement.

If we consider the hindsight optimization problem faced by an individual ad, we could add a constraint that the ad's allocation satisfies statistical parity.

$$\max_{x_i\in[0,1]^T} \sum_{t=1}^{m} (v_{it} - p_{it})x_{it} \tag{15.1}$$

$$\text{s.t. } \sum_{t=1}^{T} p_{it}x_{it} \le B_i \tag{15.2}$$

$$\frac{1}{|G_A|}\sum_{j\in G_A} x_{ij} = \frac{1}{|G_B|}\sum_{j\in G_B} x_{ij}. \tag{15.3}$$

However, this constraint is not easy to implement as part of an online allocation procedure, for two reasons. The first is that equality constraints such as this one are harder to handle as part of an online learning procedure, than the simpler "packing constraint" needed for the budgets (a less-than-or-equals constraints with only positive coefficients). The second reason is that we do not know the normalizing factors until the end.

15.5 Fairness Constraints in FPPE via Taxes and Subsidies

Now we study a potential way that we could implement demographic fairness in the context of Fisher markets and first-price ad auctions. Specifically, we will see that the Eisenbeg-Gale convex program lets us derive a tax/subsidy scheme for demographic fairness. The high-level idea is that we can consider a more constrained variant of EG for FPPE, where we insist that the computed allocation satisfies our fairness constraints, and then we can use KKT conditions to derive appropriate taxes and subsidies from the resulting Lagrange multipliers on the fairness constraints. To be concrete, suppose that for a group of buyers $I \subset [n]$, perhaps representing a particular group of sensitive ads such as job ads, we wish to enforce statistical parity across this group in an FPPE setting. Then, we can consider the following constrained version of the EG program:

$$\max_{x\geq 0,\delta\geq 0,u} \quad \sum_i B_i \log(u_i) - \delta_i$$

$$u_i \leq \sum_j x_{ij} v_{ij} + \delta_i, \forall i \tag{15.4}$$

$$\sum_i x_{ij} \leq 1, \forall j, \tag{15.5}$$

$$\sum_{i\in I} \sum_{j\in G_A} x_{ij} = \sum_{i\in I} \sum_{j\in G_B} x_{ij} \tag{15.6}$$

Now, our EG program maximizes the quasilinear EG objective, but over a smaller set of feasible allocations: those that satisfy the statistical parity constraint across buyers in I.

The key to analyzing this new quasilinear EG variant is to use the Lagrange multipliers on Eq. (15.6). Let (x, p) be the optimal allocation, and let p be the prices derived from the Lagrange multipliers on the supply constraints Eq. (15.5). Let λ be the Lagrange multiplier on Eq. (15.6). We will show that (x, p, λ) is a form of market equilibrium, where we charge each buyer $i \in I$ a price of $p_j + \lambda$ for $j \in A$ and a price of $p_j - \lambda$ for $j \in B$, where λ is the Lagrange multiplier on Eq. (15.6). Buyers $i \notin I$ are simply charged the price vector p. Clearly, this is not our usual notion of market equilibrium: we are charging two different sets of prices: prices for buyers in I and prices for buyers not in I.

First, consider some non-sensitive buyer $i \notin I$. For such a buyer, we can show that $x_i \in D_i(p)$ using the exact same argument as in the case of the standard quasilinear EG program in Theorem 29. Similarly, we can show that each item is fully allocated if $p_j > 0$ using the same arguments as before. It is also direct from feasibility that the statistical parity constraint is satisfied.

Given the above, we only need to see what happens for buyers $i \in I$. Ignoring feasibility conditions which are straightforward, the KKT conditions pertaining to buyer i are as follows:

1. $\frac{B_i}{u_i} = \beta_i \Leftrightarrow u_i = \frac{B_i}{\beta_i}$
2. $\beta_i \leq 1$
3. $\beta_i \leq \frac{p_j \pm \lambda}{v_{ij}}$
4. $\delta_i > 0 \Rightarrow \beta_i = 1$
5. $x_{ij} > 0 \Rightarrow \beta_i = \frac{p_j \pm \lambda}{v_{ij}}$

Here, the $\pm$ should be interpreted as $+$ for $j \in A$ and $-$ for $j \in B$. Now it is straightforward from KKT conditions 3 and 5 that buyer i buys only items with optimal price-per-utility under the prices $p_j \pm \lambda$. From here, the same argument as in Theorem 29 can be performed in order to show that buyer i spends their whole budget, which shows that they received a bundle $x_i \in D_i(p \pm \lambda)$.

It follows from the above that (x, p, λ) is a market equilibrium (with different prices for I and $[n] \setminus I$), and thus we can use the Lagrange multiplier λ as a tax/subsidy scheme in order to enforce statistical parity.

15.6 Historical Notes

The field of "algorithmic fairness" pioneered a lot of the fairness considerations that we considered in this note, in the context of machine learning. Dwork et al. [50] introduced similar treatment in the context of machine learning classification, and the notion that we use here for ad auction allocation is an adaptation of their definitions. They also study statistical parity in the classification context. A book-level treatment of fairness in machine learning is given by Barocas et al. [7]. Many of these fairness notions were also previously known in the education testing and psychometrics literature. See the biographical notes in Barocas et al. [7] for an overview of these older works. The quasilinear Fisher market model with statistical parity constraints via taxes and subsidies was studied in Peysakhovich et al. [107], which also studies several other fairness questions in the context of Fisher markets. A related work is Jalota et al. [77]. This work does not study fairness directly, but shows how per-buyer linear constraints can be implemented in a similar way to what we describe in Section 15.5.

Chapter 16

Stackelberg equilibrium and Security Games

16.1 Introduction

In this lecture we introduce *Stackelberg equilibrium*. Stackelberg equilibrium is an equilibrium notion for two-player general-sum games where one player is a *leader* and the other player is a *follower* (it can also be generalized to multiple leaders and/or followers). This model is appropriate for example when modeling competing firms and first-mover advantage, or as we will see in this lecture, security settings centered around asset protection.

16.2 Stackelberg Equilibrium

We will consider a two-player normal-form game where there is a leader ℓ and a follower f. The leader has a finite set of actions A_ℓ and the follower has a finite set of actions A_f. We let Δ^ℓ, Δ^f denote the set of probability distributions over the leader and follower actions. We will consider a general-sum game with utilities $u_i(a_\ell, a_f)$ for $i \in \{\ell, f\}$. We abuse notation slightly and let

$$u_i(x, y) = \mathbb{E}_{a_\ell \sim x, a_f \sim y} \left[u_i(a_\ell, a_f) \right],$$

where $x \in \Delta^\ell, y \in \Delta^f$ are probability distributions over A_ℓ and A_f respectively. In general we assume that the leader is able to commit to a strategy $x \in X$, and given such an x, the follower chooses their strategy from the best-response set

$$BR(x) = \operatorname{argmax}_{y \in \Delta^f} u_f(x, y).$$

The goal of the leader is to choose a strategy a strategy x maximizing their utility subject to the follower best responding. Formally, they wish to solve

$$\max_{x \in \Delta^\ell} u_\ell(x, y) \ s.t. \ y \in BR(x). \tag{16.1}$$

However, this optimization problem has a problem currently. Can you see what it is?

The issue is that $BR(x)$ may be set valued, and $u_\ell(x, y)$ generally would differ defending on which $y \in BR(x)$ is chosen. In that case we need a rule for how to choose among the set of best responses. In a *strong Stackelberg equilibrium* (SSE) we assume that the follower breaks ties in favor of the leader. In that case the optimization problem is

$$\max_{x \in \Delta^\ell, y \in BR(x)} u_\ell(x, y). \tag{16.2}$$

SSE is, in a sense, the most optimistic variant. Conversely, we may consider the most pessimistic assumption, that ties are broken adversarially. This yields the *weak Stackelberg equilibrium* (WSE)

$$\max_{x\in\Delta^\ell}\min_{y\in BR(x)} u_\ell(x,y). \tag{16.3}$$

In practice SSE has been by far the most popular. One major advantage of SSE is that it is always guaranteed to exist, whereas WSE is not.

A first question we might ask ourselves is whether it always helps or hurts to be able to first commit to a strategy, as compared to playing a Nash equilibrium.

First, let us consider the zero-sum case. If we are in a zero-sum game, then we already saw from von Neumann's minimax theorem that we can represent the Nash equilibrium problem as

$$\min_{x\in\Delta^\ell}\max_{y\in\Delta^f}\langle x, Ay\rangle = \max_{y\in\Delta^f}\min_{x\in\Delta^\ell}\langle x, Ay\rangle.$$

It follows that Nash equilibrium and Stackelberg equilibrium are equivalent in this setting.

Second, consider the case where we restrict the leader to only committing to pure actions $a \in A_\ell$, then committing to a strategy first may hurt the leader (consider rock-paper-scissors). On the other hand, if we allow commitment to any $x \in \Delta^\ell$, then it turns out that committing to a strategy only helps.

Theorem 32. *In a general-sum game, the leader achieves weakly more utility in SSE than in any Nash equilibrium.*

Proof. Consider the Nash equilibrium (x, y) that yields the highest utility for the leader. Since the follower breaks ties in favor of the leader, we get that if the leader commits to x then the follower can at worst pick y from $BR(x)$. If they don't pick y, then they must pick something that yields even better utility for the leader. $\square$

Similarly, it can be shown that the WSE solution is at least as good as *some* Nash equilibrium payoff for the leader (see von Stengel and Zamir [135] for a proof). Thus, if we consider the range of payoffs $[L, H]$ from the lowest to highest in Stackelberg equilibrium, then that range lies above the range that we would get for Nash equilibrium.

A classic example of the difference between Nash equilibrium and Stackelberg equilibrium is in the context of *inspection games*. In an inspection game, an inspector chooses whether to inspect or not, and the inspectee chooses whether to cheat or not. An example game is shown below

	cheat	no cheat
inspect	-6, -9	-1,0
no inspection	-10, 1	0, 0

The goal of the inspector is to deter cheating, and inspecting incurs a cost of -1. When cheating occurs the inspector incurs a heavy negative cost, whether detected or not (so the goal is *not* to catch cheaters, but rather to deter cheating). The inspectee gains utility from cheating undetected (-10, 1), but incurs a heavy fine if they cheat and are inspected (-6,-9).

There is a single unique Nash equilibrium in this game, where the inspector inspects with probability $\frac{1}{10}$, and the inspectee cheats with probability $\frac{1}{5}$. This yields expected utilities of (-2, 0) for the two players.

Now consider the same game, but where we allow the inspector to be the leader in a Stackelberg game. Any strategy that inspects with probability at least $\frac{1}{10}$ will make not cheating a best response for the follower. The SSE of the game is for the inspector to inspect with probability $\frac{1}{10}$ and the inspectee to not cheat. This yields expected utilities $(-\frac{1}{10}, 0)$, which is much better for the inspector. Note furthermore that if we consider the WSE solution concept, then the inspector must inspect with probability *strictly* greater than $\frac{1}{10}$ in order to make not cheating the only best response. But this means that a WSE does not exist, since for every leader strategy that inspects with probability $p > \frac{1}{10}$, the leader can improve their utility by inspecting with any probability in the open interval $(\frac{1}{10}, p)$.

In the normal-form game setup given above, an SSE can be computed in polynomial time. In particular, say that we wanted to maximize leader utility while getting the follower to commit to a particular action

$a_f \in A_f$. We may solve this problem using the following LP:

$$\begin{aligned} \max_{x \in \Delta^\ell} & \sum_{a \in A_\ell} x_a u_\ell(a, a_f) \\ s.t. & \sum_{a \in A_\ell} x_a u_f(a, a_f) \geq \sum_{a \in A_\ell} x_a u_f(a, a'_f), \ \forall a'_f \in A_f \end{aligned}$$

Now, in order to find the optimal strategy to commit to, we may iterate over all $a_f \in A_f$, solve the LP for each, and pick the optimal solution x^* associated to the LP with the highest value.

Once we have the optimal strategy x^*, we may find the associated follower strategy simply by picking the pure strategy a_f for which x^* was the LP solution. Generally, it is easy to see that it is always enough to consider only pure strategies when choosing the follower strategy in an SSE (why?). The same holds true for WSE.

This LP-based algorithm also proves that an SSE is always guaranteed to exist.

16.3 Security Games

In the security games model (SGM) a defender (the leader) is interested in protecting a set of targets using limited resource, while an attacker (the follower) is able to observe the strategy of the leader, and best respond to it. A classical example would be that of protecting an airport: say we have 5 vulnerable locations at the airport, but only 2 patrol units. How can we schedule the patrols so as to provide maximum coverage across the 5 vulnerable locations, while taking into account the fact that an attacker would prefer certain locations over others?

The basic security games model has a set T of targets (note that we could have a single target appear twice in T, representing multiple time steps). The defender controls a set of resources R that can be assigned to a *schedule* from a set $S \subseteq 2^T$ of possible schedules. A schedule is a subset of targets that are simultaneously covered if a resource is assigned that given schedule (for example in the airport example, a resource would be a patrol, and schedules would be the set of feasible patrols across targets). We say that a target is "covered" if the defender assigns a resource to a schedule that covers it. The action space for the attacker consists of choosing which single target to attack. In the basic SSG model, the utility function of both the defender and attacker depends only on which target is attacked, and whether it is covered or not. Formally, we say that the defender receives utility $u_d^c(t)$ if target t is attacked and covered, and utility $u_d^u(t)$ if target t is attacked an not covered. Similarly, the attacker gains utility $u_a^c(t)$ if target t is attacked and covered, and $u_a^u(t)$ if target t is attacked and not covered. If the resources R are not homogenous then there may be an *assignment function* $A : R \to S$ denoting the set of schedules s that resource r can be assigned.

For security games we will restrict our attention to SSE. Given a strategy x for the defender, we get a deployment of resources to targets for the defender, with an induced probability distribution $p_c(t|x)$ of whether each target is covered. A strategy for the attacker simply specifies a single target t to attack. Thus for a strategy pair x, t the expected utility for the defender is $p_c(t|x)u_d^c(t) + (1 - p_c(t|x))u_d^u(t)$, with attacker utility defined analogously.

16.3.1 Algorithms for Security Games

So now that we have a game model for security games, can we just apply our LP result on computing SSE in order to get an SSE for security games? Not quite: in order to convert the SGM into a standard normal-form game we get a combinatorial blow-up: consider that a pure strategy would be a deployment of resources to targets. But now let's say that we just have d patrols as our resources and k targets, and a simple model where each patrol can cover exactly one target. In that case we have $\binom{k}{d}$ pure strategies for the leader. A similar blow-up happens for other natural setups such as when each resource can cover two targets (e.g. air marshals that protect an outgoing and then ingoing flight as their daily routine).

In the special case where each resource covers exactly one target (equivalently, schedules have size 1) there is an LP-based approach that can still construct an SSE in polynomial time. This LP still allows heterogeneous resources; below we let $A(r)$ be the set of targets that resource r is allowed to cover. The key idea in the LP is to use the marginal coverage probability $p_c(t|x)$ as our decision variable. We will have an

LP where the variable c_t is the coverage probability on target t, and the variable $c_{r,t}$ is the probability that resource r provides coverage for $t \in A(r)$. The goal is to maximize the defender utility subject to making some target t^* a best response for the attacker. We can then solve for each $t^* \in T$ as before, and pick the best. In this LP, we let $u_a(t|c) = c_t u_a^c(t) + (1 - c_t)u_a^u(t)$, with $u_d(t|c)$ defined analogously.

$$\begin{aligned}
\max_{c \geq 0} \quad & u_d(t^*|c) & & \\
s.t. \quad & c_t = \sum_{r \in R \text{ s.t. } t \in A(r)} c_{r,t} & \leq 1, & \quad \forall t \in T \\
& \sum_{t \in A(r)} c_{r,t} & \leq 1, & \quad \forall r \in R \\
& u_a(t|c) & \leq u_a(t^*|c), & \quad \forall t \in T.
\end{aligned} \tag{16.4}$$

This LP is polynomial in size, even though the set of pure strategies is exponential in size. It is however not immediately obvious whether the given coverage probabilities are implementable. It turns out that they are, and this can be shown via the famous Birkhoff-von Neumann theorem. Before stating the theorem, we need the definition of a *doubly substochastic matrix*, which is a matrix $M \in \mathbb{R}^{m \times n}$ such that all entries are nonnegative, each row sums to at most 1, and each column sums to at most 1.

Theorem 33 (Birkhoff-von Neumann theorem). *If M is doubly substochastic, then there exist matrices $M_1, M_2, \ldots, M_q$, and weights $w_1, w_2, \ldots, w_q \in (0, 1]$, such that:*

1. $\sum_k w_k = 1$

2. $\sum_k w_k M_k = M$

3. *For all k, M_k is doubly substochastic, and all entries are in $\{0, 1\}$*

Informally, the theorem states that if we have a doubly substochastic matrix, then it is possible to express it as a convex combination of "pure" or $\{0, 1\}$ doubly substochastic matrices.

The coverage probabilities $c_{r,t}$ from our LP can be viewed as a matrix with rows corresponding to resources and columns corresponding to targets. By the constraints in our LP, that matrix is doubly substochastic. It follows from the Birkhoff-von Neumann theorem that there exists pure-strategy matrices M_k (they are pure strategies by the 3rd condition of the theorem) such that their convex combination under the weight vector w adds up the correct coverage probabilities (by the 2nd condition of the theorem). By the first condition, the vector w defines a mixed strategy.

One final worry is that we don't know how large q will be in our application of the Birkhoff-von Neumann theorem. Luckily, it turns out one can show (in general), that q is $O((m+n)^2)$, and the corresponding M_k, w_k can be found in $O((m + n)^{4.5})$ time using the Dulmage-Halperin algorithm.

Unfortunately, in the more general case where schedules may cover more than one target the trick using marginal coverage probabilities turns out to fail. In that case, computing an SSE turns out to be NP-hard. Still, in practice we are usually in some variant of the hard case. There are a variety of mixed-integer programming approaches that have been used to handling this case, usually leading to acceptable performance on the real-world instances at hand. Typically these approaches are tailored to the specific application, in order to get the most compact formulation. For that reason we will not cover them here.

16.4 Criticisms of Security Games

In security games we make a number of assumptions that can easily be critiqued: first, we assume that the attacker perfectly observes the defender strategy. Secondly, the defender knows exactly what the utility function of the attacker is (and SSE relies heavily on this). Thirdly, we assume that the attacker is perfectly rational. There are ways to address these assumptions. For example, a lot of work has gone into modeling adversaries in a way that is robust either to misspecification of the utility functions or followers not being perfectly rational.

16.5 Bayesian Games

One way to deal with uncertainty around utility is to assume that each player has a parameterized utility function $u_i(\cdot, \cdot|\theta_i)$, where $\theta_i \in \Theta_i$ is the *type* of player i. In Bayesian games, we assume each player draws their type from a pair of known distributions over Θ_ℓ, Θ_f. The player observes their own type before choosing an action, but not the type of the follower.

It turns out that in the special case where the follower has a single type θ_f and the leader has a probability mass $p_\ell(\theta)$ over a finite set Θ_ℓ, the LP approach for normal-form games can easily be extended to this setting and yields an optimal strategy for the leader. However, the more interesting case where the follower has multiple types is unfortunately NP-hard.

16.6 Historical Notes

The Stackelberg game model was introduced by von Stackelberg [133] in order to analyze competing firms and first-mover advantage.

The foundations for the use of Stackelberg equilibrium in security games were laid by von Stengel and Zamir [135] who showed that it always helps to commit to a strategy, as long as mixed strategies are allowed, and Conitzer and Sandholm [43] who gave efficient algorithms and complexity results around computing Stackelberg equilibrium for various game models.

In the context of security, Stackelberg equilibrium was first applied to airport security at Los Angeles International Airport [108], and has since been applied to problems such as preventing poaching and illegal fishing [57] and airport security screening [18]. An overview of deployed systems and new directions can be found in Sinha et al. [117].

The approach based on representing strategies in terms of the marginal probability of coverage was introduced by Kiekintveld et al. [79], and the results on polynomial-time algorithms and computational complexity in this model were given by Korzhyk et al. [83].

Chapter 17

Fair Allocation with Combinatorial Utilities

17.1 Introduction

Recall that for the setting of indivisible goods, a market equilibrium is not guaranteed to exist. Moreover, envy-free allocations are also not guaranteed to exist. In this lecture note we will see how to recover existence by considering an appropriate notion of approximate market equilibria. We will use this to design a fair method for allocating course seats to students.

Specifically, we will look at a generalization of the *competitive equilibrium from equal incomes* (CEEI) allocation mechanism. Since a market equilibrium is not guaranteed to exist for equal budgets, we will instead look at *approximate CEEI* (A-CEEI). In A-CEEI the idea is to relax two parts of CEEI: (1) we give agents approximately equal, rather than exactly equal, budgets, and (2) we only clear the market approximately.

Let's see how this works with an example. Consider an example where two agents are trying to divide four goods: two diamonds (one large (LD), one small (SD)), and two rocks (one pretty (PR), one ugly (UR)). Say the agents both have utilities such that they can take at most two items, and they prefer bundles in the order

$$(LD, SD) > (LD, PR) > (LD, UR) > (LD) > (SD, PR) > (SD, UR) > (SD) > (PR, UR) > (PR) > (UR).$$

Clearly if budgets are equal we cannot hope to price these items in a way that clears the market, since both agents will always want the bundle with the large diamond if they can afford it. But if we instead give agent 1 a budget of 1.2 and agent 2 a budget of 1, then we can set the prices as follows:

LD	SD	PR	UR
1.10	0.8	0.2	0.1

Now agent 1 wishes to buy (LD, UR) for a total price of 1.2, and agent 2 wishes to buy (SD, PR) for a total price of 1. As long as we decide the budget perturbations in a randomized way this is in some sense fair in expectation, and furthermore we might hope that the budget perturbations are small enough that for instances with more than four items, things look even fairer. Note that the allocation we found satisfies both EF1 and the MMS guarantee. The example also achieves Pareto optimality, but we will in general only guarantee approximate Pareto optimality for A-CEEI for more general valuations.

17.2 Approximate CEEI

We will describe the problem in the context of matching students to seats in courses. This setup is used in the *Course Match* software, which is used for matching students at Wharton and several other schools. There is a set of m courses, and each course j has some capacity s_j. There is a set of n students. Each student has a set $\Psi_i \subseteq 2^m$ of feasible subsets of courses that they may be allocated, with each bundle containing at most $k \leq m$ courses (note that this assumes that each student can only consume one unit of a good, even if $s_j > 1$; this is of course reasonable in course allocation, but not for all applications). The set Ψ_i encodes both scheduling constraints such as courses meeting at the same time, as well as constraints specific to the student such as whether they satisfy the prerequisites. The preferences of student i are assumed to be given as a complete and transitive ordinal preference ordering $\succcurlyeq_i$ over Ψ_i. Completeness simply means that for all schedules $x, x' \in \Psi_i$, $x \succcurlyeq_i x'$, $x' \succcurlyeq_i x$, or both. Transitivity means that if $x \succcurlyeq_i x'$ and $x' \succcurlyeq_i x''$ then $x \succcurlyeq_i x''$.

Given a set of prices p for each course, a vector x_i^* is in the demand set for student i if

$$x_i^* \in \operatorname{argmax}_{\succcurlyeq_i}\{x_i \in \Psi_i : \langle x_i, p\rangle \leq B_i\}.$$

In the actual Course Match implementation, $\succcurlyeq_i$ is represented numerically by an utility function for each student, but the A-CEEI theory works for the more general case of ordinal preferences.

Since we have existence issues (these arise both from indivisibility as seen earlier, but also from the very general preference orderings allowed), we resort to an approximation to CEEI:

Definition 9. *An allocation x, prices p, and budgets B constitute an (α, β)-CEEI if:*

1. $x_i \in \operatorname{argmax}_{\succcurlyeq_i}\{x' \in \Psi_i : \langle p, x'\rangle \leq B_i\}$ *for all i*
2. $\|z\|_2 \leq \alpha$, *where $z \in \mathbb{R}_+^m$ is defined as $z_j = \sum_i x_{ij} - s_j$ if $p_j > 0$, and $z_j = \max(\sum_i x_{ij} - s_j, 0)$ if $p_j = 0$*
3. $B_i \in [1, 1+\beta]$ *for all i*

The first condition in (α, β)-CEEI simply says that each student i buys an item in their demand set. The second condition says that supply constraints are approximately satisfied. The third constraint says that all budgets are almost the same, up to a difference of β.

The main theorem regarding (α, β)-CEEI is that they are guaranteed to exist:

Theorem 34. *Let $\sigma = \min(2k, m)$. For any $\beta > 0$, there exists a $(\sqrt{\sigma m}/2, \beta)$-CEEI. Moreover, given budgets $B \in [1, 1+\beta]^n$ and any $\epsilon > 0$, there exists a $(\sqrt{\sigma m}/2, \beta)$-CEEI using budgets B^* such that $\|B^* - B\|_\infty \leq \epsilon$.*

One major concern with this result is that we are not quite guaranteed a feasible solution. In general the allocation may oversubscribe some courses, though the oversubsciption vector z has bounded ℓ_2 norm. In practice, the bound is relatively modest: First, the bound $\sqrt{\sigma m}/2$ does not grow with the number of agents or number of course seats. Second, in practice students take at most a modest number of courses per semester among a reasonably-small number of courses offered (an example given in the literature is that students take $k = 5$ courses out of 50 courses total at Harvard's MBA program), thus yielding a bound of roughly 11. Technically a single course could be oversubscribed by 11 students, but in practice we expect this to be smoothed out reasonably across many courses.

The proof of the existence theorem is rather involved and relies on smoothing out the market in order to invoke fixed-point theorems. Here we give some intuition for the role that each approximation plays.

As in other discontinuous settings, the main difficulty for existence without approximation is the discontinuity of student demands with respect to price. However, in the course match setting, $\sqrt{\sigma}$ is an upper bound on the discontinuity of the demand of any single agent. To see this, note that a demand x_i has at most k entries set to 1, and so a student can at most drop all courses from x_i and switch to k new courses under their new demand x_i'. At the same time, there's only m courses total, so the change is bounded by $\min(2k, m)$, and thus $\|x_i - x_i'\|_2 \leq \sqrt{\sigma}$.

The second discontinuity issue is to avoid large discontinuous aggregate changes in demand across the students. When budgets are the same, as in standard CEEI, the demand discontinuity across students may occur at the same point in the space of prices. Thus, if this happens, aggregate discontinuity may be on the order of $n\sigma$. With distinct budgets, it becomes possible to change a single student's demand without changing those of other students. For each bundle x, we may think of the hyperplane $H(i, x) = \{p : \langle p, x\rangle \leq B_i\}$ which denotes the boundary between two halfspaces in the price space: those where student i can afford x, and those where i cannot afford x. By having each budget distinct, one can show that in a generic sense, at most m hyperplanes can intersect at any particular point in price space. This implies that aggregate demand changes by at most σm.

The remainder of the proof is concerned with smoothing out the aggregate demands so that a fixed-point existence theorem can be applied to show existence.

17.2.1 Fairness and Optimality Properties of A-CEEI

Since we are only approximately clearing the market, we do not get Pareto optimality. However, it is possible to show that if we construct a modified market where $\tilde{s}_j = s_j - z_j$, then we have Pareto optimality in that market. Thus, any Pareto-improving allocation must utilize unused supply, which can potentially be used to bound the inefficiency once more structure is imposed on utilities.

Crucially, (α, β)-CEEI does guarantee some fairness properties. If we select $\beta \leq \frac{1}{k-1}$, then EF1 is guaranteed in any (α, β)-CEEI. Furthermore, there exists β small enough such that each student is also guaranteed to receive their $(n+1)$-MMS share, which is their utility if they were forced to partition the items into $n+1$ bundles and take the worst one.

17.2.2 Practical Course Match Concerns

In Course Match, the representation of $\succcurlyeq_i$ is as follows: the set of feasible schedules Ψ_i is taken as given. Then, student i ranks each course on a scale from $0 - 100$, and is additionally allowed to specify pairwise penalties or bonuses in $-200, 200$ for being assigned a given pair of courses.

17.2.3 Computing A-CEEI

In general computing an A-CEEIis PPAD complete. This is the same class of problem that general-sum Nash equilibrium falls in. It is conjectured to require exponential time in the worst case, and thus we cannot hope to have nice scalable algorithms like we had for the divisible case.

In practice, A-CEEI is computed using local search. A *tabu search* is used on the space of prices. This works as follows:

1. A price vector is generated randomly
2. A set of "neighbors" are generated using two different generation approaches:
   - "Price gradient:" all the demands under the current prices are added up, and the excess demand vector is treated as a gradient. Then, 20 different stepsizes are tried along the price gradient
   - A single item has its price changed, and all other prices are kept the same. The new price on the chosen item is set high enough to stop it from being oversubscribed, or low enough to stop being underscribed. A neighbor is generated for each over or undersubscribed item
3. The best neighbor (among the ones generating a previously-unseen allocation) is selected as the next price vector, and the procedure repeats from step 2 (unless the last 5 iterations yielded no improving prices, in which case the local search stops)
4. Finally, step 1 is repeated with a new random price vector. This repeats until a time limit is reached

In practice this procedure generates an A-CEEI solution with significantly better α and β values than the theory predicts, within roughly two days of computation. In the process, about 4.25 billion MIPs are solved. After an A-CEEI has been generated, additional heuristics are implemented in order to force the solution to not have oversubscription.

17.3 Historical Notes

A-CEEI was introduced by Budish [26], and an implementation of A-CEEI used at Wharton was given by Budish et al. [27]. The proof of PPAD completeness was by Othman et al. [106].

Chapter 18

Large-Scale Fisher Market Equilibrium

18.1 Introduction

We saw that market equilibrium comes up in Internet scale settings such as fair recommender systems and budget-smoothed auctions (via pacing equilibrium). In this lecture note we will look at methods for computing market equilibrium at scale. In particular, we will consider two complementary approaches: 1) how to run fast iterative methods in order to compute a market equilibrium, and 2) how to abstract the market, either down to a manageable size, or in order to deal with incomplete valuations.

18.2 Setup Recap

As in previous lecture notes, we study Fisher markets: we have a set of m infinitely-divisible goods that we wish to divide among n buyers. Without loss of generality we assume that each good has supply 1. We will denote the bundle of goods given to buyer i as x_i, where x_{ij} is the amount of good j that is allocated to buyer i. We shall use x to denote an assignment of goods to buyers. Each buyer is endowed with a budget B_i of currency.

Each buyer is assumed to have a linear utility function $u_i(x_i) = \langle v_i, x_i \rangle$ denoting how much they like the bundle x_i. The results in this lecture note all carry over to quasi-linear utilities $u_i(x_i, p) = \langle v_i - p, x_i \rangle$ unless otherwise noted. Since we will be solving the Eisenberg-Gale convex program, the quasi-linear results also carry over to computing a first-price pacing equilibrium.

As mentioned in a prior note, a *market equilibrium* is a set of prices $p \in \mathbb{R}_+^m$ for each of the m goods, as well as an allocation x of goods to buyers such that everybody is assigned an optimal allocation given the prices and their budget. Formally, the *demand set* of an buyer i with budget B_i is

$$D(p) = \text{argmax}_{x_i \geq 0} u_i(x_i) \text{ s.t. } \langle p, x_i \rangle \leq B_i$$

A market equilibrium is an allocation-price pair (x, p) s.t. $x_i \in D(p)$ for all buyers i, and $\sum_i x_{ij} = 1$.

18.3 Interlude on Convex Conjugates

Given a function $f : \mathbb{R}^n \to \mathbb{R}$ we say that its *convex conjugate* is the function

$$f^*(y) = \sup_x \langle y, x \rangle - f(x)$$

We will be interested in the convex conjugate of the function $f(x) = -\log x$. We get

$$f^*(y) = \sup_x \; yx + \log x$$

and using first-order optimality we get $x^* = -1/y$, so we get that for $y < 0$

$$f^*(y) = -1 + \log(-1/y) = -1 - \log(-y) \tag{18.1}$$

18.4 Duals of the Eisenberg-Gale Convex Program

In a previous lecture we saw that the following convex program, which we called the *Eisenberg-Gale convex program* (EG) yields a market equilibrium for Fisher markets with linear utilities:

$$\begin{array}{llll|l} \max_{x\geq 0} & \sum_i B_i \log u_i & & & \text{Dual variables} \\ s.t. & u_i & \leq \langle v_i, x_i\rangle, & \forall i = 1,\dots,n, & \beta_i \\ & \sum_i x_{ij} & \leq 1, & \forall j = 1,\dots,m, & p_j \end{array} \tag{EG}$$

Remember that $x_i \in \mathbb{R}^m$ is the allocation for buyer i, and u_i is the utility.

We will now show how to derive the dual of this convex, and eventually use a further duality step to derive an interesting and very practical algorithm for solving EG. We introduce dual variables β_i (corresponding to the utility price of buyer i), and p_j (the price of item j). The dual variables are listed on the right of their corresponding primal constraint in EG. We construct the Lagrangian

$$L(x,\beta,p) = \sum_i B_i \log u_i + \sum_i \beta_i(\langle v_i, x_i\rangle - u_i) + \sum_j p_j(1 - \sum_i x_{ij})$$

The standard Lagrangian dual is then

$$\min_{p\geq 0,\beta\geq 0} \max_{x\geq 0} L(x,\beta,p) \tag{18.2}$$

Now, we simplify the inner max:

$$\begin{aligned} \max_{x\geq 0} L(x,\beta,p) &= \sum_j p_j + \sum_i \left[\max_{u_i}(B_i\log u_i - \beta_i u_i) + \max_{x_i\geq 0}\langle \beta_i v_i - p, x_i\rangle\right] \\ &= \sum_j p_j + \sum_i \left[\max_{u_i}(B_i\log u_i - \beta_i u_i) + \delta\left[\beta_i v_i \leq p\right]\right] \\ &= \sum_j p_j + \sum_i \left[B_i \max_{u_i}\left(\log u_i - \frac{\beta_i}{B_i}u_i\right) + \delta\left[\beta_i v_i \leq p\right]\right] \\ &= \sum_j p_j + \sum_i \left[B_i\left(-1 - \log\beta_i + \log B_i\right) + \delta\left[\beta_i v_i \leq p\right]\right] \end{aligned}$$

The first equality is by rearranging terms. The second equality is by noting that the max over $x_i \geq 0$ is positive infinity if $\beta_i v_{ij} > p_j$ for any j. The third equality is by rearranging B_i. The fourth equality is by (18.1).

Thus we get that the dual (18.2) is equal to

$$\begin{aligned} \min_{p\geq 0,\beta\geq 0} \quad & \sum_j p_j - \sum_i B_i \log(\beta_i) + \sum_i (\log B_i - B_i) \\ & p_j \geq v_{ij}\beta_i, \quad \forall i,j \end{aligned} \tag{18.3}$$

Finally we may drop the terms $\sum_i (\log B_i - B_i)$ since they are constant, which finally yields the standard dual of EG:

$$\begin{aligned} \min_{p\geq 0,\beta\geq 0} \quad & \sum_j p_j - \sum_i B_i \log(\beta_i) \\ & p_j \geq v_{ij}\beta_i, \quad \forall i,j \end{aligned} \tag{18.4}$$

18.4.1 Shmyrev's Convex Program

Now we introduce a change of variables to (18.4), by letting $q_j = \log p_j$ and $\gamma_i = -\log \beta_i$. Plugging these definitions into (18.4) we get

$$\begin{aligned} \min_{q,\gamma} \quad & \sum_j e^{q_j} + \sum_i B_i \gamma_i \\ & q_j + \gamma_i \geq \log v_{ij}, \quad \forall i, j \end{aligned} \tag{18.5}$$

Now we introduce Lagrangian variables b_{ij} for the constraint in (18.5) to get the following dual:

$$\begin{aligned} &\max_{b \geq 0} \min_{q,\gamma} \sum_j e^{q_j} + \sum_i B_i \gamma_i + \sum_{ij} b_{ij} \left[\log v_{ij} - q_j - \gamma_i\right] \\ =& \max_{b \geq 0} \left[\sum_{ij} b_{ij} \log v_{ij} + \sum_j \min_{q_j} \left[e^{q_j} - \sum_i b_{ij} q_j \right] + \sum_i \min_{\gamma_i} \gamma_i \left[B_i - \sum_j b_{ij} \right] \right] \end{aligned}$$

Now first-order optimality on γ_i shows $B_i = \sum_j b_{ij}$ and first-order optimality on q_j shows $e^{q_j} = \sum_i b_{ij}$. In a slight abuse of notation, we will introduce a dual variable $p_j = e^{q_j}$. Putting this together we get *Shmyrev's convex program*:

$$\begin{aligned} \max_{b \geq 0} \quad & \sum_{ij} b_{ij} \log v_{ij} + \sum_j (p_j - p_j \log p_j) \\ s.t. \quad & \sum_i b_{ij} = p_j, \quad \forall j = 1, \ldots, m, \\ & \sum_j b_{ij} = B_i, \quad \forall i = 1, \ldots, n, \end{aligned} \tag{18.6}$$

Since $\sum_j p_j = \sum_i B_i$, which is a constant, we may rewrite Shmyrev's CP as

$$\begin{aligned} \max_{b \geq 0} \quad & \sum_{ij} b_{ij} \log v_{ij} - \sum_j p_j \log p_j \\ s.t. \quad & \sum_i b_{ij} = p_j, \quad \forall j = 1, \ldots, m, \\ & \sum_j b_{ij} = B_i, \quad \forall i = 1, \ldots, n, \end{aligned} \tag{Shmyrev}$$

18.5 First-Order Methods

We will now apply online mirror descent (OMD) to (Shmyrev). Remember that OMD makes updates according to the rule:

$$x_{t+1} = \operatorname*{argmin}_{x \in X} \langle \eta \nabla f_t(x), x \rangle + D(x \| x_t).$$

where $\eta > 0$ is the stepsize and $D(x\|x_t)$ is the Bregman divergence between x and x_t.

In order to instantiate OMD, we first rewrite (Shmyrev) in terms of b_{ij} only (letting $p_j(b) = \sum_i b_{ij}$) to get the objective function

$$f(b) = -\sum_{ij} b_{ij} \log v_{ij} + \sum_j p_j(b) \log p_j(b) = -\sum_{ij} b_{ij} \log(v_{ij}/p_j(b)).$$

The feasible set is

$$X = \left\{ b \in \mathbb{R}_+^{n \times m} | \sum_j b_{ij} = B_i, \forall i \right\}.$$

Finally, we use the distance function $d(b) = \sum_{ij} b_{ij} \log b_{ij}$ which gives $D(b\|a) = \sum_{ij} b_{ij} \log(b_{ij}/a_{ij})$

At each time t, we simply see the loss $f(b^t)$. The gradient is $\nabla_{ij} f(b) = 1 - \log(v_{ij}/p_j(b))$. Similar to when using the negative entropy on the simplex, the OMD update becomes (setting $\eta = 1$):

$$\begin{aligned} b_{ij}^{t+1} &\propto b_{ij}^t \exp(-1 + \log(v_{ij}/p_j(b))) \\ &\propto b_{ij}^t \, (v_{ij}/p_j(b)) \\ &= \frac{1}{Z} b_{ij}^t \, (v_{ij}/p_j(b)) \end{aligned}$$

where Z is a normalization constant such that $\sum_j b_{ij}^{t+1} = B_i$.

Amazingly, OMD on (Shmyrev) using a stepsize of 1 becomes the following very natural algorithm:

- At each time t, each buyer i submits a bid vector b_i^t (the current OMD recommendation)
- Given the bids, a price $p_j^t = \sum_i b_{ij}^t$ is computed for each item
- Each buyer is given $x_{ij}^t = \frac{b_{ij}^t}{p_j^t}$ of each item
- Each buyer submits their next bid on item j proportional to the utility they received from item j in round t:

$$b_{ij}^{t+1} = B_i \frac{x_{ij}^t v_{ij}}{\sum_{j'} x_{ij'}^t v_{ij'}}$$

It remains to discuss the fact that we set $\eta = 1$. In past lecture notes we saw that the uniform average of OMD iterates converges to zero average regret at a rate of $O(1/\sqrt{T})$, when using a stepsize proportional to the inverse of the largest observed dual norm of gradients. However, our objective f does not admit such a bound: the gradient for i, j goes to infinity as $p_j(b)$ tends to zero. Thus based on our existing framework for OMD we are not even guaranteed a bound on regret.

However, it turns out that one can show the following "1-Lipschitz" condition relative to D:

Lemma 5. *For all $a, b \in S$,*

$$f(b) \le f(a) + \langle \nabla f(a), b - a \rangle + D(b \| a), \quad \forall b, a \in X.$$

This inequality is a sort of generalized Lipschitz condition where we replace the ℓ_2 norm $\|a - b\|_2^2$ that is typically used with our Bregman divergence D (this is analogous to how OMD itself generalized projected gradient descent by changing the distance function).

To show this inequality, we will need the fact that the Bregman divergence $D(b\|a)$ is convex in both arguments for $b, a \in \mathbb{R}_{++}^{n \times m}$. To see that convexity holds, one can expand $D(b\|a) = \sum_{ij} b_{ij} \log(b_{ij}/a_{ij})$ and note that taking a sum preserves convexity. At that point, we only need to check convexity of the function $h(t, x) = t \log(t/x) = -t \log(x/t)$, which is simply the perspective of $-\log(x)$ with respect to t. Taking perspectives is known to preserve convexity, and the negative log is of course convex.

Proof. The proof of the inequality can be split into two parts. First, it can be observed that the difference between $f(b)$ and its linearization at a is the Bregman divergence $D(p(b)\|p(a))$:

$$\begin{aligned}
&f(b) - f(a) - \langle \nabla f(a), b - a\rangle \\
=&-\sum_{ij} b_{ij}\log(v_{ij}/p_j(b)) + \sum_{ij} a_{ij}\log(v_{ij}/p_j(a)) - \sum_{ij}\left(1 - \log(v_{ij}/p_j(a))\right)(b_{ij} - a_{ij}) \\
=&-\sum_{ij} b_{ij}\log(v_{ij}/p_j(b)) + \sum_{ij} b_{ij}\log(v_{ij}/p_j(a)) - \sum_{ij}(b_{ij} - a_{ij}) \\
=&\sum_{ij} b_{ij}\log(p_j(b)/p_j(a)) - \sum_{ij}(b_{ij} - a_{ij}) \\
=&\sum_{ij} b_{ij}\log(p_j(b)/p_j(a)) \quad ;\text{ since } \|a\|_1 = \|b\|_1 = \sum_i B_i \\
=&\sum_{j} p_j(b)\log(p_j(b)/p_j(a)) \quad ;\text{ since } p_j(b) = \sum_i b_{ij} \\
=&D(p(b)\|p(a))
\end{aligned}$$

Secondly, we can bound $D(p(b)\|p(a))$ as follows (where $h(t,x) = t\log(t/x)$)

$$\begin{aligned}
D(p(b)\|p(a)) =& n\sum_j \frac{1}{n}h(p_j(b), p_j(a)) \\
=& n\sum_j h\left(\frac{1}{n}p_j(b), \frac{1}{n}p_j(a)\right) \\
\le& n\sum_j \frac{1}{n}\sum_i h\left(b_{ij}, a_{ij}\right) \\
=& D(b\|a)
\end{aligned}$$

Putting together the two bounds we get Lemma 5. □

Using the Lipschitz-like condition on f, one can show a stronger statement when running OMD on a static objective f (which means that it is the same as running normal mirror descent):

Theorem 35. *The OMD iterates with $\eta = 1$ converge at the rate:*

$$f(b^t) - f(b^*) \le \frac{\log nm}{t}.$$

This holds for any convex and differentiable f and D satisfying 5

Note two very nice properties here: the convergence rate is improved by a factor of $\sqrt{t}$, and the iterates themselves converge, with no need for averaging. We won't prove the above theorem here, but it holds for any convex minimization problem that satisfies the relative Lipschitz condition in Lemma 5.

18.6 Abstraction Methods

So far we have described a scalable first-order method for computing market equilibrium. Still, this algorithms makes a number of assumptions that may not hold in practice. First, the size of an iterate b^t is nm; if both are on the order of 100,000 then writing down an iterate using 64-bit floats requires about 80 GB of memory. For an application such as an Internet advertising market we might expect n, and especially m, to be even larger than that. Thus we may need to find a way to abstract that market down to some manageable size where we can at least hope to write down iterates. Secondly, in practice we may not have access to all v_{ij}. Instead, we may only have samples from v_{ij}, and we need to somehow infer the remaining valuations.

We now move to considering abstraction methods, which will allow us to deal with both of the above issues.

For the purposes of abstraction, it will be useful to think of the set of valuations v_{ij} as a matrix V, where the i'th row corresponds to the valuation vector of buyer i. We will be interested in what happens if we compute a market equilibrium using some valuation matrix $\tilde{V} \neq V$, where $\tilde{V}$ would typically be obtained from some abstraction method. Can we say anything about how "close" to market equilibrium we are in terms of the original V, for example if $\|\tilde{V} - V\|_F$ is small?

We first describe two reasons that we might compute a market equilibrium for $\tilde{V}$ rather than V:

1. *Low-rank markets*: When there are missing valuations, we need to somehow impute the missing values. Of course, if there is no relationship between the entries of V that we observed, and those that are missing, then we have no hope of recovering V. However, in practice this is typically not the case. In practice, the valuations are often assumed to (approximately) belong to some low-dimensional space. A popular model is to assume that the valuations are *low rank*, meaning that every buyer i has some d-dimensional vector ϕ_i, every good j has some d-dimensional vector ψ_j, and the valuation of buyer i for good j is $\tilde{v}_{ij} = \langle \phi_i, \psi_j \rangle$. One may interpret this model as every item having some associated set of d *features*, with ψ_j describing the value for each feature, and ϕ_i describes the value that i places on each feature. In a low-rank model d is expected to be much smaller than $\min(n, m)$, meaning that V is far from full rank. If the real valuations are approximately rank d (meaning that the remaining spectrum of V is very small), then $\tilde{V}$ will be close to V.

   This model can also be motivated via the singular-value decomposition (SVD). Assume that we wish to solve the following problem:

$$\begin{aligned} &\min_{\tilde{V}} \sum_{ij} (v_{ij} - \tilde{v}_{ij})^2 = \|V - \tilde{V}\|_F^2 \\ &s.t. \ \operatorname{rank}(\tilde{V}) \leq d \end{aligned}$$

   The optimal solution to this problem can be found easily via the SVD: Letting $\sigma_1, \dots, \sigma_d$ be the first d singular values of V, and $\bar{u}_1, \dots, \bar{u}_d$ the first left singular vectors, and $\bar{v}_1, \dots, \bar{v}_d$ the first right singular vectors, the optimal solution is

$$\tilde{V} = \sum_{k=1}^{d} \sigma_k \bar{u}_k \bar{v}_k^T.$$

   If the remaining singular values $\sigma_{k+1}, \dots$ are small relative to the first k singular values, then this model captures most of the valuation structure.

   In practice we don't know V, and so we can't solve this mathematical program to get $\tilde{V}$. Instead, we search for a low-rank model that minimizes some loss on the observed entries, e.g. $\sum_{ij \in \Omega} (v_{ij} - \langle \phi_i, \psi_j \rangle)^2$ (in practice this objective is typically also regularized by the Frobenius norm of the low-rank matrices). Under the assumption that V is generated from a true low-rank model via some simple distribution, it is possible to recover the original matrix with only samples of entries by minimizing the loss on observed entries. In practice this approach is also known to perform extremely well, and it is used extensively at major Internet companies (the hypothesis here would be that in practice the data is approximately low rank, so we don't lose much accuracy from a rank-d model).

2. *Representative Markets*: We may wish to try to generate a smaller set of representative buyers, where each original buyer i maps to some particular representative buyer $r(i)$. Similarly, we may wish to generate representative goods that correspond to many non-identical but similar goods from the original market. In practice these representative buyers and goods would typically be generated via clustering techniques. In this case, our approximate valuation matrix $\tilde{V}$ has as row i the valuation vector of the representative buyer $r(i)$. This means that all i, i' such that $r(i) = r(i')$ have the same valuation vector in $\tilde{V}$, and thus they can be treated as a single buyer for equilibrium-computation purposes. The same grouping can also be applied to the goods. If the number of buyers and goods is reduced by a factor of 10, then the resulting mathematical program is reduced by a factor of 10^2, since we have $n \times m$ variables.

18.6.1 Measuring Solution Quality

We now analyze what happens when we compute a market equilibrium under $\tilde{V}$ rather than V. Throughout this section we will let $(\tilde{x}, \tilde{p})$ be a market equilibrium for $\tilde{V}$. We will use the error matrix $\Delta V = V - \tilde{V}$ to quantify the solution quality, and we will measure the size of ΔV using the $\ell_1 - \ell_\infty$ matrix norm:

$$\|\Delta V\|_{1,\infty} = \max_i \|\Delta v_i\|_1.$$

We will also use the norm of the error vector for an individual buyer $\|\Delta v_i\|_1 = \|v_i - \tilde{v}_i\|_1$.

A very useful property is that under linear utilities, the change in utility when going from v_i to $\tilde{v}_i$ is linear in Δv_i.

Proposition 2. *If $\langle \tilde{v}_i, x_i\rangle + \epsilon \geq \langle \tilde{v}_i, x_i'\rangle$ then $\langle v_i, x_i\rangle + \epsilon + \|\Delta v_i\|_1 \geq \langle v_i, x_i'\rangle$*

Proof. We have

$$\begin{aligned} &\langle \tilde{v}_i, x_i\rangle + \epsilon \geq \langle \tilde{v}_i, x_i'\rangle \\ \Leftrightarrow\ &\langle v_i - \Delta v_i, x_i\rangle + \epsilon \geq \langle v_i + \Delta v_i, x_i'\rangle \\ \Leftrightarrow\ &\langle v_i, x_i\rangle + \langle \Delta v_i, x_i' - x_i\rangle + \epsilon \geq \langle v_i, x_i'\rangle \end{aligned}$$

Now the proposition follows by $\langle \Delta v_i, x_i' - x_i\rangle \leq \|\Delta v_i\|_1$. □

This proposition can be used to immediately derive bounds on envy, proportionality, and regret (how far each buyer is from achieving the utility of their demand bundle). For example, we know that under $\tilde{V}$, each buyer i has no envy towards any other buyer k: $\langle \tilde{v}_i, \tilde{x}_i\rangle \geq \langle \tilde{v}_i, \tilde{x}_k\rangle$. By Proposition 2 each buyer i has envy at most $\|\Delta v_i\|_1$ under V when using $(\tilde{x}, \tilde{p})$. All envies are thus bounded by $\|\Delta V\|_{1,\infty}$. Regret and proportionality is bounded similarly using guaranteed inequalities under $\tilde{V}$.

Market equilibrium also guarantees Pareto optimality. Can we give any meaningful guarantees on how much social welfare improves under Pareto-improving allocati8ons for $\tilde{V}$? Unfortunately the answer to that is no, as the following example of real and abstracted matrices shows:

$$V = \begin{bmatrix} 1 & \epsilon & \epsilon \\ 0 & 1 & \epsilon \end{bmatrix}, \quad \tilde{V} = \begin{bmatrix} 1 & \epsilon & 0 \\ 0 & 1 & \epsilon \end{bmatrix}.$$

If we set $B_1 = B_2 = 1$, then for supply-aware market equilibrium, we end up with competition only on item 2, and we get prices $\tilde{p} = (0, 2, 0)$ and allocation $\tilde{x}_1 = (1, 0.5, 0), \tilde{x}_2 = (0, 0.5, 1)$. Under V this is a terrible allocation, and we can Pareto improve by using $x_1 = (1, 0, 0.5), x_2 = (0, 1, 0.5)$, which increases overall social welfare by $\frac{1}{2} - \epsilon$, in spite of $\|\Delta V\|_1 = \epsilon$.

On the other hand, we can show that under any Pareto-improving allocation, some buyer i improves by at most $\|\Delta V\|_{1,\infty}1$. To see this, note that for any Pareto improving allocation x, under $\tilde{V}$ there existed at least one buyer i such that $\langle \tilde{v}_i, \tilde{x}_i - x_i\rangle \geq 0$, and so this buyer must improve by at most $\|\Delta v_i\|_1$ under V.

18.7 Historical Notes

The Shmyrev CP was given by Shmyrev [115]. The observation that the Shmyrev CP is related to EG via duality and change of variables was by Cole et al. [42]. The original proportional response dynamics were given by Wu and Zhang [136], and was shown to be effective for BitTorrent sharing dynamics by Levin et al. [93]. The relationship of PR dynamics to Shmyrev's CP and mirror descent were given by Birnbaum et al. [13]. For rules on convexity-preserving operations, see Boyd and Vandenberghe [17].

There is a long history of algorithms for computing market equilibrium in various Fisher-market models. In this lecture note we focused on a particular method that is, to the best of our knowledge, one of the fastest simple and scalable first-order methods for computing a market equilibrium.

The material on abstracting large market equilibrium problems is from Kroer et al. [87].

A brief introduction to low-rank models can be found in Udell and Townsend [126]. Udell et al. [127] gives a more through exposition and describes more general model types. There's also a fascinating theory of low-rank models, where a number of cool results are known: there's a class of nuclear-norm-regularized convex optimization problems that can recover the original matrix with only a small number of entry samples [30, 111]. One might think that this would then be the preferred method in practice, but surprisingly non-convex models are often preferred instead. These non-convex methods also have interesting guarantees on statistical recovery under certain assumptions. An overview of non-convex methods is given in Chi et al. [38].

Low-rank market equilibrium models were also studied in Kroer and Peysakhovich [85], where it is shown that large low-rank markets enjoy a number of properties not satisfied by small-scale markets.

Chapter 19

Power Flows and Equilibrium Pricing

This chapter introduces a new topic: electricity markets, and their associated optimization problems. As we shall see, both economics and optimization play a key role in modern electricity grids.

For the first hundred years or so of the existence of the US power grid, it was managed by what are called *vertically integrated* utilities. These were companies that generated, sold, and transferred electricity directly to users. Typically these would also be monopolies, meaning that they were the only possible supplier in a given region. In contrast, the late 1990's and early 2000's saw what's usually referred to as the *deregulation*[1] of the electric grid,

In the deregulated markets, the choice of who generates what is made using auction-based mechanisms where the auctioneer is an *independent system operator* (ISO). ISOs are quasi-governmental entities whose charter is to operate the grid, including deciding who generates what using auctions. The overarching setup is very complicated, because e.g. the New York market uses two electricity auctions: a spot auction every five minutes (which decides on the allocation of generation and purchasing for the next five minutes), and a day-ahead auction every hour (which allocates power generation and purchasing for that hourly interval of the following day), as well as several *capacity auctions* meant to ensure that the grid has sufficient generation capacity. We will focus more on these auctions in the next lecture notes. First, this note will introduce the operational optimization problems that ISOs need to solve on a continuous basis.

Compared to a normal markets, the electric grid has many peculiarities. For example:

1. The grid operates in a continuous fashion, whereas the spot markets are operating every 5 minutes.

2. Supply (power generation) and demand (load generated by users) must be balanced at all times. The system will collapse if these quantities are not kept in check.

3. Goods (electricity) is generated at particular locations, and must be "transported" to the point of usage, potentially with a loss in power, or congestion of the wires

4. Electricity should be thought of as a "flow" in a network; therefore it's generally not possible to say that a particular user takes electricity from a particular plant. Both simply take electricity in and out of the "pool."

5. Different types of electricity generators (e.g. wind, gas, nuclear) all have very different operating constraints, and thus differ in their ability to increase or decrease productions, and the speed at which they can do so.

These peculiarities are good to keep in mind when thinking about the grid and its markets, because they mean that e.g. incentives can be a tricky subject.

[1]This name is arguably misleading, as the electric grid is actually a highly regulated industry still. A better term would perhaps be restructuring or decompositioning.

19.1 Optimal Power Flow

We now introduce the *optimal power flow* (OPF) problem. In OPF, we are given a directed network V, E of nodes and edges representing the electric grid in question. The set of nodes V in power parlance is called the set of *buses*. I will use nodes and buses interchangeably. The buses should be thought of as important locations in the physical grid, e.g. generation points, load points, or substations. The set of edges E is the connections between buses. In power parlance, these are called transmission *lines*. We let E_i be the set of edges departing bus i.

The alternating current OPF (ACOPF) problem is a nonconvex quadratic optimization problem which models physics of the power flow problem including the fact that complex variables are needed. In particular, the net addition or removal of flow at a bus i will be a complex variable $p_i + \mathbf{i}q_i$, and similarly the power flow on a line $(i,j) \in E$ will be a complex variable $p_{ij} + \mathbf{i}q_{ij}$. We will mostly work with a linearization of this model, but I want to briefly describe it, so that you are aware of the approximation that is being made in the eventual LP we will use. To represent the problem, we will need the following variables:

- v_i is a complex number describing the voltage at bus $i \in V$
- p_i is a real number describing the difference between generation and demand of *real* power at bus $i \in V$
- q_i is the complex part of the difference between generation and demand of *reactive* power at bus $i \in V$
- p_{ij} is the real part of the power flow on line $i, j \in E$; $p_{ij} > 0$ means power is flowing from i to j and $p_{ij} < 0$ means power flows the opposite direction
- q_{ij} is the reactive power flow on line $i, j \in E$

We will also need the following constants:

- $\mathbf{i}$ will refer to the imaginary unit satisfying $\mathbf{i}^2 = -1$
- $y_{ij} = g_{ij} + \mathbf{i}b_{ij}$ is a complex number describing the *admittance* of the line i to j
- $\underline{v}_i, \overline{v}_i$ are lower and upper bounds on the voltage at bus i
- Each bus $i \in V$ is subject to box constraints on its real power $\underline{p}_i, \overline{p}_i$, and reactive power $\underline{q}_i, \overline{q}_i$
- Each line $i, j \in E$ is subject to a bound $\overline{s}_{ij}$ on the apparent power flow $p_{ij}^2 + q_{ij}^2$

With all that, the ACOPF problem looks as follows, where f is some objective functions that we wish to optimize subject to the power flow constraints.

$$
\begin{aligned}
\min_{v,p,q} \ & f(v,p,q) \\
\text{s.t.} \ & p_{ij} + \mathbf{i}q_{ij} = v_i(v_i^* - v_j^*)y_{ij}^*, && \forall (i,j) \in E \\
& p_{ij}^2 + q_{ij}^2 \le \overline{s}_{ij}, && \forall (i,j) \in E \\
& \sum_{j \in E_i} p_{ij} = p_i, && \forall i \in V \\
& \sum_{j \in E_i} q_{ij} = q_i, && \forall i \in V \\
& p_i \in [\underline{p}_i, \overline{p}_i], && \forall i \in V \\
& q_i \in [\underline{q}_i, \overline{q}_i], && \forall i \in V \\
& |v_i| \in [\underline{v}_i, \overline{v}_i], && \forall i \in V
\end{aligned}
\tag{ACOPF}
$$

The above problem is a very difficult optimization problem. In particular, even if f is a linear function, the first constraint is a nonconvex quadratic constraint, which makes the problem NP-hard in general. This leads to numerous problems, including the fact that this problem is typically too hard to solve to optimality for real-world OPF problems. A second issue is the lack of strong duality, which is something that we will need later.

19.2 Linearized Power Flow

Going forward, we will work with a simplified model of power flows, which linearizes the nonconvex quadratic constraint in Eq. (ACOPF). We will call this model *DC power flow* (DCOPF), though this terminology is misleading, because it does not actually model direct-current power flows. Instead, it is simply a linearized approximation to AC power flows.

This model is obtained by making a number of simplifying assumptions of Eq. (ACOPF). First, because reactive power is negligible relative to real power, we set all reactive power variables to zero, meaning that we can remove all q variables and associated constraints.

Next, we write the complex variables using polar coordinates $v_i = m_i e^{\mathbf{i}\theta_i}$ for each i. Then, we get the following equation for the real part of the nonconvex equation:

$$p_{ij} = g_{ij} m_i^2 - m_i m_j \left(g_{ij} \cos(\theta_i - \theta_j) - b_{ij} \sin(\theta_i - \theta_j) \right).$$

Then, we set all voltage magnitudes equal to one, i.e. $|m_i| = 1$. Finally, we set $g_{ij} = 0$ because $g_{ij} \ll b_{ij}$.

After making all these simplifications, the DCOPF problems has only linear constraints:

$$\begin{aligned} \min_{\theta,p}\ & f(\theta, p) \\ \text{s.t.}\ & p_{ij} = b_{ij}(\theta_i - \theta_j), && \forall (i,j) \in E \\ & \sum_{j \in E_i} p_{ij} = p_i, && \forall i \in V \\ & p_i \in [\underline{p}_i, \overline{p}_i], && \forall i \in V \\ & |p_{ij}| \le \overline{s}_{ij}, && \forall (i,j) \in E \end{aligned} \tag{DCOPF}$$

If f is also a linear function, then Eq. (DCOPF) is an LP.

In the formulation given here, each node $i \in V$ has a single power flow p_i into it (if $p_i > 0$) or out of it (if $p_i < 0$).

19.3 Economic Dispatch

In practice, nodes are often thought of as locations that potentially have both generators and demands. While Eq. (DCOPF) is completely general, it will be more convenient to include these multiple types of generators and demands in the model. To that end, let Ψ_i^D be the set of demands at node i, where each demand $d \in \Psi_i^D$ has some utility u_d of receiving power, and some upper bound $\overline{p}_d$ on how much power they can consume. Similarly, let Ψ_i^g be the set of generators at node i, where each generator $g \in \Psi_i^G$ has some cost c_d of generating power, and a maximum generating capacity $\overline{p}_g$. If we now set our objective f to be equal to the social welfare of the resulting allocation, we get the following LP:

$$\max_{\theta,p} \sum_{i \in V} \left(\sum_{d \in \Psi_i^D} u_d p_d - \sum_{g \in \Psi_i^G} c_g p_g \right)$$

$$\text{s.t.}\ p_{ij} = b_{ij}(\theta_i - \theta_j), \qquad \forall (i,j) \in E \tag{19.1}$$

$$\sum_{j \in E_i} p_{ij} = \sum_{g \in \Psi_i^G} p_g - \sum_{d \in \Psi_i^D} p_d, \qquad \forall i \in V \tag{19.2}$$

$$p_d \in [0, \overline{p}_d], \qquad \forall i \in V, d \in \Psi_i^D \tag{19.3}$$

$$p_g \in [0, \overline{p}_g], \qquad \forall i \in V, g \in \Psi_i^G \tag{19.4}$$

$$|p_{ij}| \le \overline{s}_{ij}, \qquad \forall (i,j) \in E \tag{19.5}$$

A solution of this LP is referred to as *economic dispatch* because it maximizes efficiency. It also has a market equilibrium interpretation: let λ_i^* be the dual variable associated to Eq. (19.2) in an optimal solution, i.e. an economic dispatch solution. Then λ_i^* can be thought of as the *locational marginal price* (LMP) of electricity at node i: each demand at i is charged this price, and each generator at i is paid this price per unit of electricity. In fact, a variant of this LP that takes into account additional operational constraints is used for pricing in many real-world electricity markets.

19.3.1 Market Equilibrium Properties for Generators and Demands

If we consider the Lagrangified problem using the λ_i^* dual variables, we get the problem

$$\begin{aligned}
\max_{\theta,p} \ & \sum_{i\in V}\left(\sum_{d\in\Psi_i^D} u_d p_d - \sum_{g\in\Psi_i^G} c_g p_g\right) + \sum_{i\in V}\lambda_i^*\left(\sum_{g\in\Psi_i^G} p_g - \sum_{d\in\Psi_i^D} p_d - \sum_{j\in E_i} p_{ij}\right) \\
\text{s.t. } & p_{ij} = b_{ij}(\theta_i-\theta_j), && \forall (i,j)\in E && (19.6)\\
& p_d \in [0,\overline{p}_d], && \forall i\in V, d\in\Psi_i^D && (19.7)\\
& p_g \in [0,\overline{p}_g], && \forall i\in V, g\in\Psi_i^G && (19.8)\\
& |p_{ij}| \le \overline{s}_{ij}, && \forall (i,j)\in E && (19.9)
\end{aligned}$$

Now, if we consider the problem faced by an individual generator $g\in\Psi_i^G$ for some node i, in order to maximize their own utility they would like to solve the problem

$$\begin{aligned}
\max_{p_g} \ & (\lambda_i^* - c_g)p_g \\
\text{s.t. } & p_g \in [0,\overline{p}_g] \qquad (19.10)
\end{aligned}$$

But we can see that the Lagrangified LP decomposes along generators, in the sense that p_g appears only in its own constraint Eq. (19.8), and with the exact same coefficients as in the individual generator utility maximization problem. Thus, by stationarity conditions, we get that the value p_g^* from the economic dispatch solution is also optimal for the individual generator given λ_i^*. A completely analogous argument shows that each demand also maximizes its utility.

It follows from the above that the prices and allocation from economic dispatch constitute a market equilibrium.

19.3.2 Spatial Arbitrage

Finally, let us try to understand the transmission variables p_{ij} which also depend on λ_i in the objective of Eq. (19.8). Consider the following problem given the optimal λ^*:

$$\begin{aligned}
\max_{p_{ij}} \ & \sum_{i\in V}\lambda_i^* \sum_{j\in E_i} p_{ij} \\
\text{s.t. } & p_{ij} = b_{ij}(\theta_i-\theta_j), \quad \forall (i,j)\in E \\
& |p_{ij}| \le \overline{s}_{ij}, \qquad\quad \forall (i,j)\in E
\end{aligned} \qquad (19.11)$$

This can be thought of as a spatial arbitraging operation. Since $\sum_{j\in Ei} p_{ij} = \sum_{d\in\Psi_i^D} p_d - \sum_{d\in\Psi_i^G} p_g$, we know that $\lambda_i^* \sum_{j\in Ei} p_{ij}$ is the *excess* payment at node i, which can be either positive or negative. While individual line revenues for the arbitrageur may thus be positive or negative, we see that Eq. (19.11) maximizes all the possible ways of transferring power across the network, given the prices. By a similar argument as before, we see that the economic dispatch solution optimally solves the spatial arbitrage problem. Thus, if we let the transmission operator collect these excess payments, then the transmission operator acts as a spatial arbitrageur, who optimally tries to buy and sell power while satisfying the (linearized) transmission constraints.

19.3.3 Economic Dispatch as a Mechanism

The economic dispatch framework derived in this section gives us a way to use markets to allocate power consumption and generation:

- Have every demand and generator submit their utility per unit of electricity, along with the consumption and generation caps
- Compute an economic dispatch solution for who generates and consumes what
- Charge everyone according to the dual prices

This is how allocation and pricing is performed in many of the *spot markets* used by various ISOs. Spot markets run on a frequent basis (e.g. every five minutes), and determine generation and consumption for any *uncommitted* load and generation capacity. I stress the uncommitted part here, because some generators and demands will already have entered binding contracts on price and quantity in earlier markets, such as the day-ahead market.

We now investigate a few properties that would be nice to have for this market.

- **Truthfulness**: Unfortunately this mechanism is not truthful. To see this, note that while each participant acts optimally *given* the prices, they can themselves influence the prices. If one considers a network with a single node, then it is straightforward to see that some generator and demander end up being the two entities setting the marginal price. They could then misreport in order to shift this price.
- **Efficiency**: if the submitted bids are truthful, then we would get efficiency by definition of the economic dispatch model. That said, we already noted that this mechanism is easily seen to not be truthful. A second concern for efficiency is that we introduced a lot of approximations in order to arrive at an LP.
- **Budget balance**: The ISO needs to ensure that after paying generators and charging demands it ends up with a nonnegative amount of leftover money. However, we already saw in the spatial arbitrage section that the excess payments are captured via the p_{ij} variables, and the spatial arbitrager can make their utility at least zero, so revenue adequacy is guaranteed. ISOs are typically not allowed to make money either; for that reason the money made from spatial arbitrage is usually thought of as going to the providers of the transmission network, or towards additional investment in the network.
- **Individual rationality**: is every participant incentivized to participate in the market? This is easily seen to be true from the market equilibrium condition, as long as participants do not overstate their capacity, or report utilities/costs that are respectively higher/lower than their true values.

In addition to the approximations that we made going from ACOPF to DCOPF, this note also made some implicit assumptions. One of the biggest is that every generator can choose in continuous fashion how much electricity to produce. In practice, generators have various types of constraints on how they can change their output. For example, several types of energy producers require a long time to ramp up or down production (say up to a day), and they may have minimum generation levels for when they are turned on. This is the case for several traditional generators such as nuclear and coal. Natural gas also has similar constraints. This introduces a discrete nature into the problem: we may need a day or more to reach certain production levels, and so the real-time market is operating "too late" for some decisions to be made. This motivates the use of day-ahead markets, which we will study in the next lecture note.

Renewables also have different types of constraints on their production, that depend on the type of renewable. For example, wind generators are not necessarily able to adjust their output at all, and are thus required to produce electricity at whatever level the weather dictates. This can even lead to negative energy prices, depending on whether we have a cost-free way of handling excess power. All these constraints, as well as a general desire on the part of market participants for a certain amount of predictability in their revenues, necessitate additional market mechanisms that allow us to settle some generation and consumption further in advance than the spot market allows. This will be the topic of the next note.

19.4 Historical Notes

Taylor [125] covers many of the optimization aspects of the power grid. This book also has some coverage of energy markets. Kirschen and Strbac [80] has extensive coverage of the economic aspects of energy system.

Sweeney [119] provides a detailed account of the California energy crisis, which is an interesting case study in how not to design an energy markets. That crisis lead to severe blackouts, huge budget deficits for several energy companies (with one going bankrupt), and had large ramifications for the state budget.

Jalal Kazempour from the Danish Technical University has a set of slides and lecture videos[2] that give a really nice optimization-based introduction to energy markets.

[2]Found here: `https://www.jalalkazempour.com/teaching`

Chapter 20

Unit Commitment

So far, we have talked about the economic dispatch problem as if we solve it once, using a simple LP for finding the optimal generation and demand allocations. However, this is not how the ISOs actually decide on how to allocate. Instead, as mentioned briefly, there are several stages of allocation at various points in time. A key issue that we mentioned last time is that many types of power-generating plants require long startup and shutdown times (on the order of hours to a day). This is one reason to consider day-ahead (DA) markets, where we commit some plants to producing energy on the following day. Beyond startup and shutdown times, another attractive property of DA markets is that they reduce uncertainty for the parties that settle on generation and load taking in the DA market. This may, for example, simplify staffing scheduling.

20.1 Unit Commitment

In this section we study how to handle binary operational decisions. For example, a nuclear or coal power plant must decide ahead of time whether to commit to turning the plant on or not. If they do commit, they usually have some minimum power output level (in addition to an upper bound), and if they do not, then they cannot generate any power. This binary decision problem obviously causes some problems for our market-based mechanism from the last lecture note: we used strong duality to get locational marginal prices for each node in the network. But with binary variables, we will not have strong duality! This section will discuss a few potential remedies to this problem, though none of them are perfect.

For simplicity, let us consider a single-node problem, where demand is fixed at p_d. Then we get the following market clearing problem with non-convexity due to binary decisions, which is a mixed-integer linear program (MILP):

$$\min_{p,z} \sum_{g\in\Psi^G} c_g p_g + C_g z_g \tag{20.1}$$

$$\text{s.t.} \sum_{g\in\Psi^G} p_g \geq p_d \tag{20.2}$$

$$p_g \leq z_g \overline{p}_g, \qquad \forall g \in \Psi^G \tag{20.3}$$

$$p_g \geq z_g \underline{p}_g, \qquad \forall g \in \Psi^G \tag{20.4}$$

$$z_g \in \{0,1\} \qquad \forall g \in \Psi^G \tag{20.5}$$

Now suppose we solve this problem, and get a set of optimal binary variables z^*. Then it turns out that we can in fact construct prices using these binary variables. The idea is to introduce a continuous version of the MILP, where we constrain each continuous variable z_g to take on exactly the value z_g^*, and then we will use the Lagrange multiplier on that constraint to price the non-convexity. This yields the following LP, which we call EDLP:

$$\min_{p,z} \sum_{g\in\Psi^G} c_g p_g + C_g z_g \tag{20.6}$$

$$\text{s.t.} \sum_{g\in\Psi^G} p_g \geq p_d \tag{20.7}$$

$$p_g \leq z_g \overline{p}_g, \qquad \forall g \in \Psi^G \tag{20.8}$$

$$p_g \geq z_g \underline{p}_g, \qquad \forall g \in \Psi^G \tag{20.9}$$

$$z_g = z_g^* \qquad \forall g \in \Psi^G \tag{20.10}$$

Now consider an optimal solution x^*, z^*, and let λ^* be the corresponding Lagrange multiplier on Eq. (20.20) and μ_g^* be the Lagrange multiplier for Eq. (20.10) for each g. We will pay λ^* for generating electricity, and for each generator g such that $z_g^* = 1$, we pay them μ_g^* for turning on.

This turns out to yield a market equilibrium, as we will now show. Consider a generator g; they wish to solve the following problem:

$$\max_{p_g,z_g} \sum_{g\in\Psi^G} (\lambda^* - c_g)p_g + (\mu_g - C_g)z_g \tag{20.11}$$

$$\text{s.t. } p_g \leq z_g \overline{p}_g \tag{20.12}$$

$$p_g \geq z_g \underline{p}_g \tag{20.13}$$

$$z_g \in \{0,1\} \tag{20.14}$$

One way to solve this problem is to make z_g continuous, and hope that an integral solution happens to pop out. That yields the following program

$$\max_{p_g,z_g} \sum_{g\in\Psi^G} (\lambda^* - c_g)p_g + (\mu_g - C_g)z_g \tag{20.15}$$

$$\text{s.t. } p_g \leq z_g \overline{p}_g \tag{20.16}$$

$$p_g \geq z_g \underline{p}_g \tag{20.17}$$

$$z_g \in \mathbb{R} \tag{20.18}$$

Clearly an optimal solution to this problem upper bounds the optimal solution to the integral version. But now it is easy to see that if we form the Lagrangian of EDLP:

$$\min_{p,z} \sum_{g\in\Psi^G} c_g p_g + C_g z_g + \lambda^* \left(p_d - \sum_{g\in\Psi^G} p_g \right) + \sum_{g\in\Psi^G} \mu_g^* \left(z_g^* - z_g \right) \tag{20.19}$$

$$\text{s.t.} \tag{20.20}$$

$$p_g \leq z_g \overline{p}_g, \qquad \forall g \in \Psi^G \tag{20.21}$$

$$p_g \geq z_g \underline{p}_g, \qquad \forall g \in \Psi^G, \tag{20.22}$$

then we get a problem which includes exactly the same constraints on p_g, z_g, and has the same coefficients in the objective. But then by strong duality we know that $p_g = p_g^*, z_g = z_g^*$ is an optimal solution to this problem, which shows that it must be an optimal solution to the LP for generator i.

While the above approach was described in the context of unit commitment, it works much more broadly. If a generator has multiple binary decision then we can simply add one per constraint per decision, and we will then get a price for each of their binary decisions.

One drawback of this pricing approach is that it tends to produce highly volatile prices, which can be both negative and positive. This can lead to prices that can seem very unfair (and materialize suddenly through minor changes to the pricing problem). A second concern is that we may no longer have budget balance, meaning that the ISO could potentially fall short on money due to the unit commitment prices.

20.2 Uplift Payments

In practice, ISOs often use what are called *uplift payments*. Uplift payments are an asymmetric variant of the previous pricing approach. The ISO will compute only locational marginal prices. Then, for generators with discrete decisions such as unit commitment, if the LMPs do not support their assigned decisions and power output, the ISO will pay the difference. Note that this can make the generator better or worse off depending on context. For example, μ_g being negative is ignored which helps the generator, but when μ_g is positive the uplift payment could be smaller than μ_g still.

20.3 Convex Hull Pricing

An alternative pricing approach is that of *convex hull pricing* (CH pricing). CH pricing is very easy to set up. We simply Lagrangify the demand constraint, and solve the resulting minimization problem over electricity prices. Formally, we solve

$$\min_{\lambda} q(\lambda),$$

where $q(\lambda)$ is defined as

$$q(\lambda) := \begin{aligned} \min_{p,z} \ & \sum_{g \in \Psi^G} c_g p_g + C_g z_g + \lambda\big(p_d - \sum_{g \in \Psi^G} p_g\big) & \\ \text{s.t. } \ & p_g \leq z_g \overline{p}_g, & \forall g \in \Psi^G \\ & p_g \geq z_g \underline{p}_g, & \forall g \in \Psi^G \\ & z_g \in \{0,1\} & \forall g \in \Psi^G \end{aligned}$$

From an optimization perspective this approach has some attractive properties, especially the fact that given a fixed λ, solving $q(\lambda)$ decomposes into simple per-generator optimization problems. On the other hand, since we do not have strong duality, this approach does not necessarily give us a feasible solution. In practice, the resulting CH prices λ^* would be extracted, but the allocation would use the original MILP for finding a feasible allocation. This means that in general CH pricing will not be such that generators get allocations that are in their demand set. To fix this issue, ISOs would then provide additional uplift payments.

20.4 Connecting DA and RT Markets

So far we have discussed RT and DA markets in isolation. In practice, the RT market operates after a number of contracts for consumption and generation have been settled in the DA market. For example, suppose a generator was assigned 100 megawatt (MW) of generation for an RT period, but it turns out that they will only be able to produce 97MW. In that case, the remaining 3MW must be purchased in the RT market. Financially speaking, the generator would then be viewed as having purchased 3MW of power in that RT market. Similarly, a demand that purchased 100MW of power in the DA market but then consumed only 90MW would be viewed as selling 10MW of power in the RT market. In general, we can view the RT market as a balancing operation that corrects any imbalances that occur due to increased or decreased consumption or generation specified in the DA market.

If not for uncertainty, it is easy to convince yourself that the price in the DA and RT markets should be the same. If they were not, then any generator that was assigned to generate in the market with the lower price would simply wish to change their bids such that they end up getting assigned the same generation in the market with the higher price. A similar argument holds for demands.

A key reason why the RT market may nonetheless require balancing is that consumer electricity usage as forecasted in the DA market will differ from the realized usage in the RT market. This causes relatively manageable imbalances in the market, and the ISO needs to correct these imbalances in order to keep the system functioning. A second and more severe imbalance issue that can occur is generator outages. A generator outage can lead to large imbalances that require significant additional generation allocation in the RT market.

Due to these imbalances, and the very short-term nature of the RT market, flexible generation and consumption entities will be rewarded at a higher rate in the RT market when realized demands turns out to be higher than realized generation. On the other hand, expensive generators that are primarily used to cover the case of excess demand in the RT market will not make any money when realized demand is lower than realized generation. Thus, the cost of generation for such plants is often high, which can lead to higher volatility in RT market prices.

20.5 Historical Notes

The approach for pricing binary decision by using the MIP solution as constraints in the LP was introduced by O'Neill et al. [104]. Convex hull pricing was introduced by Gribik et al. [73].

Appendix A

Bregman Divergences and Proximal Mappings

This appendix chapter collects some useful facts around Bregman divergences and proximal mappings that are used in various parts of the book.

Consider the problem

$$\mathrm{Prox}(g) = \arg\min_{x \in \mathcal{X}} \langle g, x\rangle + d(x),$$

where $d : X \to \mathbb{R}$ is a strongly convex function with modulus $\mu > 0$ with respect to a norm $\|\cdot\|$. Let $\|\cdot\|_*$ denote the dual norm.

First, let us see that this function is Lipschitz

Lemma 6. *The prox function satisfies*

$$\|\mathrm{Prox}(g) - \mathrm{Prox}(\hat{g})\| \le \frac{1}{\mu}\|g - \hat{g}\|_*.$$

Proof. Let $x^* = \mathrm{Prox}(g)$ and $\hat{x}^* = \mathrm{Prox}(\hat{g})$. Let $f(x) = \langle g, x\rangle + d(x)$ and $\hat{f}(x) = \langle \hat{g}, x\rangle + d(x)$. Since the sum of a linear and strongly convex function is strongly convex with the same modulus, we have that f is strongly convex with modulus μ. Combining strong convexity and optimality of x^* and $\hat{x}^*$, we have that

$$\frac{\mu}{2}\|x^* - \hat{x}^*\|^2 \le f(\hat{x}^*) - f(x^*) = \langle g, \hat{x}^* - x^*\rangle + d(\hat{x}^*) - d(x^*),$$
$$\frac{\mu}{2}\|x^* - \hat{x}^*\|^2 \le f(x^*) - f(\hat{x}^*) = \langle \hat{g}, x^* - \hat{x}^*\rangle + d(x^*) - d(\hat{x}^*).$$

Summing the inequalities and applying Hölder's inequality yields

$$\mu\|x^* - \hat{x}^*\|^2 \le \langle g - \hat{g}, \hat{x}^* - x^*\rangle \le \|g - \hat{g}\|_*\|x^* - \hat{x}^*\|.$$

□

Alternatively, Lemma 6 can also be seen directly from a basic fact from convex analysis: if a function d is strongly convex modulus μ, then the gradient of its convex conjugate is $1/\mu$-Lipschitz. Now note that the prox function is the gradient of the convex conjugate at $-g$, i.e. $\mathrm{Prox}(g) = \nabla d^*(-g)$.

Bibliography

[1] Julia Angwin and Terry Parris Jr. Facebook lets advertisers exclude users by race. *ProPublica*, 2016. URL `https://www.propublica.org/article/facebook-lets-advertisers-exclude-users-by-race`.

[2] Julia Angwin, Noam Scheiber, and Ariana Tobin. Dozens of companies are using facebook to exclude older workers from job ads. *ProPublica*, 2016. URL `https://www.propublica.org/article/facebook-ads-age-discrimination-targeting`.

[3] Santiago Balseiro, Anthony Kim, Mohammad Mahdian, and Vahab Mirrokni. Budget management strategies in repeated auctions. In *Proceedings of the 26th International Conference on World Wide Web*, pages 15–23, 2017.

[4] Santiago R Balseiro and Yonatan Gur. Learning in repeated auctions with budgets: Regret minimization and equilibrium. *Management Science*, 65(9):3952–3968, 2019.

[5] Santiago R Balseiro, Omar Besbes, and Gabriel Y Weintraub. Repeated auctions with budgets in ad exchanges: Approximations and design. *Management Science*, 61(4):864–884, 2015.

[6] Siddharth Barman, Sanath Kumar Krishnamurthy, and Rohit Vaish. Finding fair and efficient allocations. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 557–574, 2018.

[7] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning*. fairmlbook.org, 2019. `http://www.fairmlbook.org`.

[8] Amir Beck. *First-order methods in optimization*, volume 25. SIAM, 2017.

[9] Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

[10] Xiaohui Bei, Jugal Garg, and Martin Hoefer. Ascending-price algorithms for unknown markets. *ACM Transactions on Algorithms (TALG)*, 15(3):1–33, 2019.

[11] Dimitri P Bertsekas, A Nedic, and A Ozdaglar. Convex analysis and optimization. 2003. *Athena Scientific*, 2003.

[12] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena scientific Belmont, MA, 1997.

[13] Benjamin Birnbaum, Nikhil R Devanur, and Lin Xiao. Distributed algorithms via gradient descent for fisher markets. In *Proceedings of the 12th ACM conference on Electronic commerce*, pages 127–136. ACM, 2011.

[14] David Blackwell. An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 6(1):1–8, 1956.

[15] Christian Borgs, Jennifer Chayes, Nicole Immorlica, Kamal Jain, Omid Etesami, and Mohammad Mahdian. Dynamics of bid optimization in online advertisement auctions. In *Proceedings of the 16th international conference on World Wide Web*, pages 531–540, 2007.

[16] Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149, 2015.

[17] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[18] Matthew Brown, Arunesh Sinha, Aaron Schlenker, and Milind Tambe. One size does not fit all: A game-theoretic approach for dynamically and effectively screening for threats. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[19] Noam Brown and Tuomas Sandholm. Safe and nested subgame solving for imperfect-information games. In *Advances in neural information processing systems*, pages 689–699, 2017.

[20] Noam Brown and Tuomas Sandholm. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.

[21] Noam Brown and Tuomas Sandholm. Solving imperfect-information games via discounted regret minimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1829–1836, 2019.

[22] Noam Brown and Tuomas Sandholm. Superhuman AI for multiplayer poker. *Science*, 365(6456): 885–890, 2019.

[23] Noam Brown, Adam Lerer, Sam Gross, and Tuomas Sandholm. Deep counterfactual regret minimization. *arXiv preprint arXiv:1811.00164*, 2018.

[24] Noam Brown, Tuomas Sandholm, and Brandon Amos. Depth-limited solving for imperfect-information games. In *Advances in Neural Information Processing Systems*, pages 7663–7674, 2018.

[25] Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[26] Eric Budish. The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. *Journal of Political Economy*, 119(6):1061–1103, 2011.

[27] Eric Budish, Gérard P Cachon, Judd B Kessler, and Abraham Othman. Course match: A large-scale implementation of approximate competitive equilibrium from equal incomes for combinatorial allocation. *Operations Research*, 65(2):314–336, 2016.

[28] Neil Burch, Michael Johanson, and Michael Bowling. Solving imperfect information games using decomposition. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[29] Neil Burch, Matej Moravcik, and Martin Schmid. Revisiting cfr+ and alternating updates. *Journal of Artificial Intelligence Research*, 64:429–443, 2019.

[30] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Found Comput Math*, 9:717–772, 2009.

[31] Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D Procaccia, Nisarg Shah, and Junxing Wang. The unreasonable fairness of maximum Nash welfare. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, pages 305–322. ACM, 2016.

[32] Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D Procaccia, Nisarg Shah, and Junxing Wang. The unreasonable fairness of maximum nash welfare. *ACM Transactions on Economics and Computation (TEAC)*, 7(3):1–32, 2019.

[33] Antonin Chambolle and Thomas Pock. On the ergodic convergence rates of a first-order primal–dual algorithm. *Mathematical Programming*, 159(1-2):253–287, 2016.

[34] Lihua Chen, Yinyu Ye, and Jiawei Zhang. A note on equilibrium pricing as convex optimization. In *International Workshop on Web and Internet Economics*, pages 7–16. Springer, 2007.

[35] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player nash equilibria. *Journal of the ACM (JACM)*, 56(3):1–57, 2009.

[36] Xi Chen, Christian Kroer, and Rachitesh Kumar. The complexity of pacing for second-price auctions. In *Proceedings of the 2021 ACM Conference on Economics and Computation*, 2021.

[37] Yun Kuen Cheung, Richard Cole, and Nikhil R Devanur. Tatonnement beyond gross substitutes? gradient descent to the rescue. *Games and Economic Behavior*, 2019.

[38] Yuejie Chi, Yue M Lu, and Yuxin Chen. Nonconvex optimization meets low-rank matrix factorization: An overview. *IEEE Transactions on Signal Processing*, 67(20):5239–5269, 2019.

[39] Mark Cieliebak, Stephan J Eidenbenz, Aris Pagourtzis, and Konrad Schlude. On the complexity of variations of equal sum subsets. *Nord. J. Comput.*, 14(3):151–172, 2008.

[40] Edward H Clarke. Multipart pricing of public goods. *Public choice*, pages 17–33, 1971.

[41] Richard Cole and Lisa Fleischer. Fast-converging tatonnement algorithms for one-time and ongoing market problems. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 315–324, 2008.

[42] Richard Cole, Nikhil R Devanur, Vasilis Gkatzelis, Kamal Jain, Tung Mai, Vijay V Vazirani, and Sadra Yazdanbod. Convex program duality, fisher markets, and Nash social welfare. In *18th ACM Conference on Economics and Computation, EC 2017*. Association for Computing Machinery, Inc, 2017.

[43] Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*, pages 82–90, 2006.

[44] Vincent Conitzer and Tuomas Sandholm. New complexity results about Nash equilibria. *Games and Economic Behavior*, 63(2):621–641, 2008.

[45] Vincent Conitzer, Christian Kroer, Eric Sodomka, and Nicolás E Stier-Moses. Multiplicative pacing equilibria in auction markets. In *International Conference on Web and Internet Economics*, 2018.

[46] Vincent Conitzer, Christian Kroer, Debmalya Panigrahi, Okke Schrijvers, Eric Sodomka, Nicolas E Stier-Moses, and Chris Wilkens. Pacing equilibrium in first-price auction markets. In *Proceedings of the 2019 ACM Conference on Economics and Computation*. ACM, 2019.

[47] Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009.

[48] Constantinos Daskalakis, Alan Deckelbaum, and Anthony Kim. Near-optimal no-regret algorithms for zero-sum games. *Games and Economic Behavior*, 92:327–348, 2015.

[49] Gerard Debreu. A social equilibrium existence theorem. *Proceedings of the National Academy of Sciences*, 38(10):886–893, 1952.

[50] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226. ACM, 2012.

[51] David Easley, Jon Kleinberg, et al. *Networks, crowds, and markets: Reasoning about a highly connected world*, volume 1. Cambridge university press Cambridge, 2010.

[52] Benjamin Edelman and Michael Ostrovsky. Strategic bidder behavior in sponsored search auctions. *Decision support systems*, 43(1):192–198, 2007.

[53] Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *American economic review*, 97(1): 242–259, 2007.

[54] Edmund Eisenberg. Aggregation of utility functions. *Management Science*, 7(4):337–350, 1961.

[55] Edmund Eisenberg and David Gale. Consensus of subjective probabilities: The pari-mutuel method. *The Annals of Mathematical Statistics*, 30(1):165–168, 1959.

[56] Ky Fan. Fixed-point and minimax theorems in locally convex topological linear spaces. *Proceedings of the National Academy of Sciences of the United States of America*, 38(2):121, 1952.

[57] Fei Fang, Peter Stone, and Milind Tambe. When security games go green: Designing defender strategies to prevent poaching and illegal fishing. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[58] Fei Fang, Thanh H Nguyen, Rob Pickles, Wai Y Lam, Gopalasamy R Clements, Bo An, Amandeep Singh, Brian C Schwedock, Milin Tambe, and Andrew Lemieux. Paws—a deployed game-theoretic application to combat poaching. *AI Magazine*, 38(1):23–36, 2017.

[59] Gabriele Farina, Christian Kroer, and Tuomas Sandholm. Regret minimization in behaviorally-constrained zero-sum games. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1107–1116. JMLR. org, 2017.

[60] Gabriele Farina, Christian Kroer, and Tuomas Sandholm. Online convex optimization for sequential decision processes and extensive-form games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1917–1925, 2019.

[61] Gabriele Farina, Christian Kroer, and Tuomas Sandholm. Optimistic regret minimization for extensive-form games via dilated distance-generating functions. In *Advances in Neural Information Processing Systems*, pages 5222–5232, 2019.

[62] Gabriele Farina, Christian Kroer, and Tuomas Sandholm. Stochastic regret minimization in extensive-form games. In *International Conference on Machine Learning*. PMLR, 2020.

[63] Gabriele Farina, Christian Kroer, and Tuomas Sandholm. Faster game solving via predictive blackwell approachability: Connecting regret matching and mirror descent. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI, 2021.

[64] Gabriele Farina, Julien Grand-Clément, Christian Kroer, Chung-Wei Lee, and Haipeng Luo. Regret matching+:(in) stability and fast convergence in games. *arXiv preprint arXiv:2305.14709*, 2023.

[65] Arthur Flajolet and Patrick Jaillet. Real-time bidding with side information. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5168–5178. Curran Associates Inc., 2017.

[66] Genevieve E Flaspohler, Francesco Orabona, Judah Cohen, Soukayna Mouatadid, Miruna Oprescu, Paulo Orenstein, and Lester Mackey. Online learning with optimism and delay. In *International Conference on Machine Learning*, pages 3363–3373. PMLR, 2021.

[67] Sam Ganzfried and Tuomas Sandholm. Endgame solving in large imperfect-information games. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 37–45, 2015.

[68] Yuan Gao, Christian Kroer, and Donald Goldfarb. Increasing iterate averaging for solving saddle-point problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[69] Mohammad Ghodsi, MohammadTaghi HajiAghayi, Masoud Seddighin, Saeed Seddighin, and Hadi Yami. Fair allocation of indivisible goods: Improvements and generalizations. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 539–556, 2018.

[70] Itzhak Gilboa and Eitan Zemel. Nash and correlated equilibria: Some complexity considerations. *Games and Economic Behavior*, 1(1):80–93, 1989.

[71] Irving L Glicksberg. A further generalization of the kakutani fixed point theorem, with application to nash equilibrium points. *Proceedings of the American Mathematical Society*, 3(1):170–174, 1952.

[72] Jonathan Goldman and Ariel D Procaccia. Spliddit: Unleashing fair division algorithms. *ACM SIGecom Exchanges*, 13(2):41–46, 2015.

[73] Paul R Gribik, William W Hogan, Susan L Pope, et al. Market-clearing electricity prices and energy uplift. *Cambridge, MA*, pages 1–46, 2007.

[74] Theodore Groves. Incentives in teams. *Econometrica: Journal of the Econometric Society*, pages 617–631, 1973.

[75] Sergiu Hart and Andreu Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.

[76] Samid Hoda, Andrew Gilpin, Javier Pena, and Tuomas Sandholm. Smoothing techniques for computing Nash equilibria of sequential games. *Mathematics of Operations Research*, 35(2):494–512, 2010.

[77] Devansh Jalota, Marco Pavone, Qi Qi, and Yinyu Ye. Fisher markets with linear constraints: Equilibrium properties and efficient distributed algorithms. *Games and Economic Behavior*, 141:223–260, 2023.

[78] Ramesh Johari. MS&E 336: dynamics and learning in games, 2007.

[79] Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordóñez, and Milind Tambe. Computing optimal randomized resource allocations for massive security games. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 689–696, 2009.

[80] Daniel S Kirschen and Goran Strbac. *Fundamentals of power system economics*. John Wiley & Sons, 2018.

[81] Tinne Hoff Kjeldsen. John von neumann's conception of the minimax theorem: a journey through different mathematical contexts. *Archive for history of exact sciences*, 56(1):39–68, 2001.

[82] Daphne Koller, Nimrod Megiddo, and Bernhard von Stengel. Efficient computation of equilibria for extensive two-person games. *Games and economic behavior*, 14(2):247–259, 1996.

[83] Dmytro Korzhyk, Vincent Conitzer, and Ronald Parr. Complexity of computing optimal stackelberg strategies in security resource allocation games. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[84] Vijay Krishna. *Auction theory*. Academic press, 2009.

[85] Christian Kroer and Alexander Peysakhovich. Scalable fair division for'at most one'preferences. *arXiv preprint arXiv:1909.10925*, 2019.

[86] Christian Kroer, Gabriele Farina, and Tuomas Sandholm. Solving large sequential games with the excessive gap technique. In *Advances in Neural Information Processing Systems*, pages 864–874, 2018.

[87] Christian Kroer, Alexander Peysakhovich, Eric Sodomka, and Nicolas E Stier-Moses. Computing large market equilibria using abstractions. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 745–746, 2019.

[88] Christian Kroer, Kevin Waugh, Fatma Kılınç-Karzan, and Tuomas Sandholm. Faster algorithms for extensive-form game solving via improved smoothing functions. *Mathematical Programming*, pages 1–33, 2020.

[89] David Kurokawa, Ariel D Procaccia, and Junxing Wang. Fair enough: Guaranteeing approximate maximin shares. *Journal of the ACM (JACM)*, 65(2):1–27, 2018.

[90] Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. Monte Carlo sampling for regret minimization in extensive games. In *Advances in neural information processing systems*, pages 1078–1086, 2009.

[91] Chung-Wei Lee, Christian Kroer, and Haipeng Luo. Last-iterate convergence in extensive-form games. In *Advances in Neural Information Processing Systems, NeurIPS 2019*, 2021.

[92] Euiwoong Lee. Apx-hardness of maximizing nash social welfare with indivisible items. *Information Processing Letters*, 122:17–20, 2017.

[93] Dave Levin, Katrina LaCurts, Neil Spring, and Bobby Bhattacharjee. Bittorrent is an auction: analyzing and improving bittorrent's incentives. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pages 243–254, 2008.

[94] Matej Moravcik, Martin Schmid, Karel Ha, Milan Hladik, and Stephen J Gaukrodger. Refining subgames in large imperfect information games. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[95] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

[96] APS Mosek. The mosek optimization software. *Online at http://www. mosek. com*, 54(2-1):5, 2010.

[97] Arkadi Nemirovski. Prox-method with rate of convergence o (1/t) for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251, 2004.

[98] Arkadi Nemirovsky and David Borisovich Yudin. Problem complexity and method efficiency in optimization. 1983.

[99] Yu Nesterov. Excessive gap technique in nonsmooth convex minimization. *SIAM Journal on Optimization*, 16(1):235–249, 2005.

[100] Yu Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103:127–152, 2005.

[101] Yurii Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009.

[102] Yurii Nesterov and Vladimir Shikhman. Computation of fisher–gale equilibrium by auction. *Journal of the Operations Research Society of China*, 6(3):349–389, 2018.

[103] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani. *Algorithmic game theory*. Cambridge University Press, 2007.

[104] Richard P O'Neill, Paul M Sotkiewicz, Benjamin F Hobbs, Michael H Rothkopf, and William R Stewart Jr. Efficient market-clearing prices in markets with nonconvexities. *European journal of operational research*, 164(1):269–285, 2005.

[105] Francesco Orabona. A modern introduction to online learning. *arXiv preprint arXiv:1912.13213*, 2019.

[106] Abraham Othman, Christos Papadimitriou, and Aviad Rubinstein. The complexity of fairness through equilibrium. *ACM Transactions on Economics and Computation (TEAC)*, 4(4):1–19, 2016.

[107] Alexander Peysakhovich, Christian Kroer, and Nicolas Usunier. Implementing fairness constraints in markets using taxes and subsidies. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, pages 916–930, 2023.

[108] James Pita, Manish Jain, Janusz Marecki, Fernando Ordóñez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. Deployed armor protection: the application of a game theoretic model for security at the los angeles international airport. 2008.

[109] Leonid Denisovich Popov. A modification of the arrow-hurwicz method for search of saddle points. *Mathematical notes of the Academy of Sciences of the USSR*, 28(5):845–848, 1980.

[110] Alexander Rakhlin and Karthik Sridharan. Optimization, learning, and games with predictable sequences. In *Proceedings of the 26th International Conference on Neural Information Processing Systems-Volume 2*, pages 3066–3074, 2013.

[111] Benjamin Recht. A simpler approach to matrix completion. *Journal of Machine Learning Research*, 12:3413–3430, 2011.

[112] IV Romanovskii. Reduction of a game with full memory to a matrix game. *Doklady Akademii Nauk SSSR*, 144(1):62–+, 1962.

[113] Tim Roughgarden. *Twenty lectures on algorithmic game theory.* Cambridge University Press, 2016.

[114] Sheryl Sandberg. Doing more to protect against discrimination in housing, employment and credit advertising. *Facebook*, 2019. URL `https://about.fb.com/news/2019/03/protecting-against-discrimination-in-ads/`.

[115] Vadim I Shmyrev. An algorithm for finding equilibrium in the linear exchange model with fixed budgets. *Journal of Applied and Industrial Mathematics*, 3(4):505, 2009.

[116] Yoav Shoham and Kevin Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations.* Cambridge University Press, 2008.

[117] Arunesh Sinha, Fei Fang, Bo An, Christopher Kiekintveld, and Milind Tambe. Stackelberg security games: Looking beyond a decade of success. IJCAI, 2018.

[118] Maurice Sion et al. On general minimax theorems. *Pacific Journal of mathematics*, 8(1):171–176, 1958.

[119] James L Sweeney. *The California electricity crisis.* Hoover Press, 2013.

[120] Vasilis Syrgkanis, Alekh Agarwal, Haipeng Luo, and Robert E Schapire. Fast convergence of regularized learning in games. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pages 2989–2997, 2015.

[121] Kalyan Talluri and Garrett Van Ryzin. An analysis of bid-price controls for network revenue management. *Management science*, 44(11-part-1):1577–1593, 1998.

[122] Milind Tambe. *Security and game theory: algorithms, deployed systems, lessons learned.* Cambridge university press, 2011.

[123] Oskari Tammelin. Solving large imperfect information games using CFR+. *arXiv preprint arXiv:1407.5042*, 2014.

[124] Oskari Tammelin, Neil Burch, Michael Johanson, and Michael Bowling. Solving heads-up limit Texas hold'em. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[125] Joshua Adam Taylor. *Convex optimization of power systems.* Cambridge University Press, 2015.

[126] Madeleine Udell and Alex Townsend. Why are big data matrices approximately low rank? *SIAM Journal on Mathematics of Data Science*, 1(1):144–160, 2019.

[127] Madeleine Udell, Corinne Horn, Reza Zadeh, Stephen Boyd, et al. Generalized low rank models. *Foundations and Trends® in Machine Learning*, 9(1):1–118, 2016.

[128] Hal R Varian. Position auctions. *international Journal of industrial Organization*, 25(6):1163–1178, 2007.

[129] Hal R Varian and Christopher Harris. The vcg auction in theory and practice. *American Economic Review*, 104(5):442–45, 2014.

[130] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance*, 16(1):8–37, 1961.

[131] John von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische annalen*, 100(1):295–320, 1928.

[132] John von Neumann. On the theory of games of strategy. *Contributions to the Theory of Games*, 4: 13–42, 1959.

[133] Heinrich von Stackelberg. *Marktform und gleichgewicht*. J. springer, 1934.

[134] Bernhard von Stengel. Efficient computation of behavior strategies. *Games and Economic Behavior*, 14(2):220–246, 1996.

[135] Bernhard von Stengel and Shmuel Zamir. Leadership games with convex strategy sets. *Games and Economic Behavior*, 69(2):446–457, 2010.

[136] Fang Wu and Li Zhang. Proportional response dynamics leads to market equilibrium. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 354–363, 2007.

[137] Haifeng Xu. The mysteries of security games: Equilibrium computation becomes combinatorial algorithm design. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, pages 497–514. ACM, 2016.

[138] H Peyton Young. *Strategic learning and its limits*. OUP Oxford, 2004.

[139] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Advances in neural information processing systems*, pages 1729–1736, 2007.