Theoretical Guarantees for Algorithms in Multi-Agent Settings

Martin Zinkevich

August 2004

CMU-CS-04-161

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Thesis Committee:
Avrim Blum, Chair
Tuomas Sandholm
Daniel Sleator
Michael Littman, Rutgers, The State University of New Jersey
Tucker Balch, Georgia Institute of Technology

This material is based upon work supported under NSF grants CCR-9732705, CCR-0085982, CAREER Award IRI-9703122, IIS-9800994, ITR IIS-0081246, ITR IIS-0121678, and an NSF Graduate Research Fellowship.

Any opinion, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Keywords: multi-agent,regret minimization,competitive analysis, mechanism design,repeated bimatrix games, stochastic games

Acknowledgements

I would like to thank all of my coauthors. Particularly, parts of this thesis were based on papers written with Avrim Blum and Tuomas Sandholm. I would like to thank my committee for many useful comments. I would also like to thank Michael Bowling, Amy Greenwald, Adam Kalai, Pat Riley, Manfred Warmuth, and Ke Yang. Finally, thanks to my family and friends for their love and support.

Abstract

In this thesis we will develop and analyze algorithms in multi-agent settings. We focus on two areas: that of auctions, negotiation, and exchanges, which are of increasing interest to computer scientists with the rise of e-commerce, and that of repeated bimatrix games and multi-stage games, which provide an interesting test bed for agents that will be interacting with other intelligent agents. The main thrust of this thesis is designing algorithms for agents who are missing critical information about the environment or other agents (like what is going to happen in the future, or what the other agents' motivations are) that perform well compared to the optimal behavior which has this information.

In the area of auction design, we consider online double auctions (exchanges) for commodities, where there are buyers and sellers trading indistinguishable goods who arrive and depart over time. We consider this from the perspective of the broker, who decides what trades occur and what payments are made. The broker may be interested in maximizing social welfare, maximizing its own profit, or maximizing market liquidity. We devise an algorithm for a broker who does not know what agents will arrive in the future that is within a logarithmic factor of the optimal offline algorithm, if the agents are truthful (although the algorithm does not motivate them to bid truthfully). This is the best competitive ratio that can be achieved. In the simpler bilateral trading problem with one buyer and one seller, we consider strategic agents and study direct mechanisms where both agents are motivated to bid truthfully, partially direct mechanisms where one agent is motivated to bid truthfully and the other agent will strategize, and general mechanisms where both agents will strategize. Agents which have to strategize must have knowledge about the typical values that buyers and sellers have in order to perform well. However, we show that higher efficiency for a broker who does not have knowledge of the domain requires more strategizing on the behalf of the buyer and seller. For instance, the broker can achieve a logarithmic factor of the optimal broker who knows the distributions if the buyer is motivated to be truthful and the seller strategizes.

We also consider combinatorial auctions. We show a connection between query learning in machine learning theory and preference elicitation. We show how certain natural hypotheses spaces that can be learned with membership queries correspond to natural classes of preferences that can be learned with value queries.

Finally, we consider repeated bimatrix games. One would like two reasonable agents who encounter each other many times in the same setting (e.g. a bimatrix game) to eventually perform well together. We show how a simple gradient ascent technique performs well in a bimatrix game, as well as in an arbitrary online convex programming domain. We introduce the concept of response regret, an extension of traditional concepts of regret, to account for the short-term effects that one agent's actions have on the other agent, which also can be extended to stochastic games and partially observable stochastic games. Two behaviors that minimize internal response regret in a stochastic game or a partially observable stochastic game will approach the set of correlated equilibria.

One of the hardest parts of working in a domain with other intelligent agents is defining what it means to perform well and understanding what are the right assumptions to be made about the other agent. It is well known that there exists an algorithm that has no regret against an arbitrary algorithm. Furthermore, it is not hard to show that there exists an algorithm that achieves the minimum Nash equilibrium value against a no-external-regret algorithm. However, we show here that no algorithm can achieve both these guarantees.

Contents

Chapter 1

Introduction

A multi-agent environment is an environment where there are multiple agents, each with its own goals and algorithms for achieving these goals. These agents can be either humans or computer programs. Examples of multi-agent environments can range from stock markets to auctions to video games to classrooms to life itself. How is behaving in the presence of other intelligent agents different than behaving in an unintelligent environment? As an environment designer, how can one construct the rules for interaction to obtain globally beneficial behavior? These are questions addressed by the areas of Game Theory and Mechanism Design, respectively. The area of focus of this thesis is on incomplete knowledge in multi-agent environments. How does an agent's lack of full knowledge about the other agents and the environment affect its ability to achieve its goals, and what algorithms should it use? How can an environment designer's lack of full knowledge of the agents affect its ability to obtain a globally beneficial behavior? We will address these questions in several settings by combining game theory and mechanism design with techniques from competitive analysis and computational learning theory.

1.1 Examples of Multi-Agent Environments

This section describes some examples of multi-agent environments that we will be studying in this thesis.

One example of a multi-agent environment is an online double auction, also known as an exchange. In an online double auction, there is a set of indistinguishable goods, like Playstations, and there are agents interested in buying or selling them who will be entering and leaving the market. Each buyer and seller has a value on a Playstation, as well as an interval of time in which they are willing to trade. In other words, each buyer (seller) has an arrival time when they enter the market and an expiration time when they leave the market. There is also an auctioneer, or broker, that decides which agents trade and at what price. A buyer and a seller can be matched if their intervals of time overlap, and the broker can make a profit matching them if the price given by the buyer is higher than the price given by the seller. The auctioneer only learns about a buyer or seller when it arrives, and cannot match an agent after it leaves the auction. Consider this setting from the perspective of an auctioneer attempting to maximize profit. Even if buyers and sellers reveal their true values and expiration times when they enter the market, it is difficult to determine what to do with the current agents in the market if one has no knowledge of what agents will appear in the future. For instance, if there is a buyer about to leave the market who values a Playstation at one hundred dollars, and a seller who values a Playstation at fifty dollars who will be present for a while, should the auctioneer initialize a trade, making $50, or wait for a buyer who values a Playstation more? If there is another seller that is at eighty dollars and will expire soon, should the buyer be matched to that one instead? Thus, incomplete information (on behalf of the auctioneer about who will arrive in the future) makes this a challenging optimization problem. What if we add further uncertainty by allowing the buyers and sellers to bid strategically, instead of revealing their true valuations? Then, the auctioneer has to discover how to entice an agent to reveal its own valuation, as well as perhaps some of the information it has about the valuations of others.

Another example of a multi-agent setting is a combinatorial auction. In a combinatorial auction, there is a set of buyers, and a set of items, and the auctioneer decides how to allocate the items to the buyers and what fees to charge them. The difference between a sequence of single-item auctions and a combinatorial auction is that in a combinatorial auction, we do not assume that the valuations are linear. It could be that a group of items, e.g. airplane tickets and a hotel room, could be worth more together than apart. Also, some items, e.g. two free dinners on the same night, could be worth less together than the sum of their separate values. These properties make computing the optimal allocation difficult, even if the auctioneer knows all of the desires of the buyers. As an additional problem, each buyer's desires could be quite complex, and the auctioneer may have difficulty in eliciting enough information about the buyer's desires in a reasonable amount of time.

Other examples of multi-agent settings we will discuss are infinitely repeated bimatrix games, stochastic games, and partially observable stochastic games. In an infinitely repeated bimatrix game, two agents face each other in the same situation, or state, an infinite number of times. In a stochastic game, two or more agents interact in an environment with one or more states, and in each state the actions of each agent may have a different effect on the utility of each agent and on the next state. In a stochastic game, the state is known by each agent. In a partially observable stochastic game, each agent is unaware of the state, but makes observations that are dependent on the underlying state. In these domains, we will study what algorithms the individual agents should use. We will discuss what goals can be achieved by the agents in each of these domains, and develop new algorithms to achieve those goals.

1.2 Definition of Competitive Ratio

This thesis deals with reasoning with incomplete knowledge. In other words, there are important pieces of information that the mechanism or agents do not know, nor do they have a probability distribution over possible values of these pieces of information. The competitive ratio is a measure used in computer science to determine how well you can perform with incomplete knowledge.

As an example of a competitive analysis, consider the following problem. Imagine that you have an umbrella on a rainy day, and various people come and offer you a price for it. Eventually, the rain will stop, and then no one will buy it. However, you will at least still have an umbrella. We will make two assumptions: no one is willing to pay more than 64 dollars for an umbrella (even if they have a nice suit), and the umbrella is worth one dollar to you.

You would be really frustrated if you sold the umbrella for two dollars, and then someone came by and offered you 60 dollars for it. On the other hand, if someone offered you 59 dollars, and you didn't sell it and then saw no one else, you would also be frustrated.

Two avoid such situations, one possible selling strategy is to give the umbrella to anyone who is willing to pay 8 dollars for the umbrella. This creates a balance between the following two scenarios:

1. A first person comes and offers 7.99, and then no one else comes.

2. A first person comes and offers eight dollars, and then a second person comes and offers 64 dollars.

The **worst-case competitive ratio** is:

$$\max_{s \in \text{all scenarios}} \frac{\text{profit of the optimal algorithm in } s}{\text{your profit in } s}.$$

In the first scenario, you could have gotten 7.99. Since you got 1 dollar, your ratio is 7.99. In the second scenario, you could have gotten 64 dollars. Since you got 8 dollars, your ratio is 8. This is in fact the worst case, so the competitive ratio is 8.

Thus, if you knew for certain that no one would bid above N dollars, then the best you could do deterministically would be to choose a threshold of $\sqrt{N}$, for a worst-case competitive ratio of $\sqrt{N}$.

Now, suppose you have a private source of randomness. You know the distribution over this randomness, and are aware that it is not correlated with the scenario. Thus, you consider the ratio:

$$\max_{s \in \text{all scenarios}} \frac{\text{profit of the optimal algorithm on } s}{\text{your expected profit on } s}.$$

It is important that the expectation be taken inside the denominator, instead of over the entire fraction, which would change the result.

A good randomized algorithm would be to choose the threshold T according to:

$$\Pr[T \leq x] = \frac{\ln x - \ln p_{min}}{\ln p_{max} - \ln p_{min}},$$

where $p_{max} = 64$ and $p_{min} = 1$. This is the solution suggested for the one-way trading problem [EYFKT92]. A simple integration shows that regardless of the scenario, the worst-case competitive ratio is $\ln p_{max} - \ln p_{min}$, or approximately 2.3 in the above case.

What if, instead of actually having your umbrella, you know someone who is willing to sell their umbrella for one dollar? How can you make a profit by finding someone willing to buy the umbrella for more? What if there are more buyers and sellers who come and go over time? How can you make a profit? These are some of the questions we answer in Chapter 2.

In Chapter 3, we are using a formalism where some of the scenarios themselves may have randomness: i.e, where the optimal algorithm knows the distribution, but not the value, and you have access to neither, similar to the analysis in [KL04]. Particularly, there is a buyer and a seller, who each have a valuation on an item. However, the scenario will consist of the distribution over the possible valuations for the seller and the distribution over the possible valuations for the buyer. Thus, in this case, the competitive ratio becomes:

$$\max_{s \in \text{all scenarios}} \frac{\text{expected profit of the optimal algorithm on } s}{\text{your expected profit on } s}.$$

1.3 What is a Mechanism?

A mechanism is a way of arbitrating a formal negotiation between a group of people. For instance, an election, a stock market, and an art auction are all examples of mechanisms.

Any mechanism is created by someone or something. Usually, one assumes that the structure of the mechanism is governed by a fixed set of rules, and thus the agents inside the mechanism know both that they cannot affect these rules, and that these rules will not change over time.

Often, a mechanism is designed such that the action of an agent is "obvious" given that agent's desires, or preferences. For instance, consider the following auction for a single item:

1. Each person interested in the item submits a sealed bid.
2. The bids are opened, and the person with the highest bid gets the item for the value of the second highest bid.

This called a Vickrey auction. Regardless of the actions of the other agents, each agent can do no better than submit as its bid its value for the item. This type of mechanism is called **dominant-strategy incentive compatible**. This is a nice property, because it means that each agent can focus on its own preferences.

In contrast, think of a sealed-bid English Auction, where the item goes to the highest bidder, but the price paid is the highest bid. The mechanism designer suggests each agent bid its true value of the item, but some agents may be motivated not to do this. If you value an item at $200 but are pretty sure that no one else values it more than $100, then you might want to shade down your bid. Therefore, such a mechanism is not incentive-compatible.

Another type of incentive-compatibility is illustrated by a reserve-price auction. Imagine there is one buyer and one seller.

1. The seller chooses a reserve price r.
2. The buyer states a value t. If $t \geq r$, then the item is traded at price r: otherwise, it is not traded at all.

As before, the buyer's dominant strategy is to state its true value as t. However, in this mechanism, the mechanism designer is not interested in the seller revealing its true value as r. If $P(r)$ were the probability that a random buyer values the item more than r, the mechanism suggests the seller to use a reserve price maximizing:

$$r^* = \operatorname*{argmax}_r P(r)(r - v_1)$$

where v_1 is the value of the item to the seller. Thus, this mechanism is what is called Bayesian incentive-compatible, as opposed to dominant strategy incentive-compatible, because we are assuming buyers come from a distribution and ask for incentive-compatibility with respect to that distribution. If the seller knows the distribution from which the buyer draws its utility, then it can and will follow the advice of the mechanism: however, it may regret this after the true value of the buyer is revealed. This is similar to the concept of Bayesian Nash equilibrium, where the strategy of each agent is optimal given its beliefs about the other agent.

In addition to incentive-compatibility, another property one would like a mechanism to satisfy is individual rationality: that is, each agent should not expect to lose from participating in the mechanism. For instance, if the broker were to insist that the buyer accept the offer in the reserve-price mechanism, the buyer might no longer want to participate in the mechanism.

1.4 The Revelation Principle

The Vickrey auction is an example of a larger principle in mechanism design called the **revelation principle**: roughly, for any mechanism, one can construct an equivalent mechanism where each agent's optimal behavior is to reveal its preferences. This type of mechanism is a **direct mechanism**. For any set of preferences for the agent, the new mechanism will have an outcome equivalent to the outcome of the old mechanism.

However, the revelation principle depends very heavily on the mechanism designer understanding the market: for instance, in order to construct a mechanism equivalent to the stock market, one must be knowledgeable about all of the values that all of the stocks have to all of the agents in the market. Thus, for many real-world problems, the reduction is intractable. In addition to direct mechanisms, in this thesis we consider two alternatives:

1. Do not worry about whether or not the agents will be truthful in a mechanism, and focus on the algorithmic issues involved in making a profit. Since the mechanisms we consider are close to those that are in existence, we are justified in assuming that people will bid "nearly truthfully".
2. Consider mechanisms where agents have more complicated strategies. For instance, in the mechanisms we present in Chapter 3, we either explicitly or implicitly elicit information about one agent from another. This results in a balancing act, because if the mechanisms are too complicated, then one is making unreasonable assumptions about the knowledge the agents have about the market. However, if they are too simple, then it is difficult for the broker to obtain profit.

1.5 Original Results in Part I: Mechanism Design

Chapter 2 We study the problem of online market clearing where there is one commodity in the market being bought and sold by multiple buyers and sellers whose bids arrive and expire at different times. The auctioneer is faced with an online clearing problem of deciding which buy and sell bids to match without knowing what bids will arrive in the future. For maximizing *profit*, we present a (randomized) online algorithm with a competitive ratio of $\ln(p_{max} - p_{min}) + 1$, when bids are in a range $[p_{min}, p_{max}]$, which we show is the best possible. A simpler algorithm has a ratio twice this, and can be used even if expiration times are not known. For maximizing the *number* of trades, we present a simple greedy algorithm that achieves a factor of 2 competitive ratio if no money-losing trades are allowed. Interestingly, we show that if the online algorithm is allowed to *subsidize* matches — match money-losing pairs if it has already collected enough money from previous pairs to pay for them — then it can be 1-competitive with respect to the optimal offline algorithm that is not allowed subsidy. That is, the ability to subsidize is at least as valuable as knowing the future. We also consider the objectives of maximizing buy or sell volume, and present algorithms that achieve a competitive ratio of $2(\ln(p_{max}/p_{min}) + 1)$, or $\ln(p_{max}/p_{min}) + 1$ if the online algorithm is allowed subsidization. We show the latter is the best possible competitive ratio for this setting. For social welfare maximization we also obtain an optimal competitive ratio, which is below $\ln(p_{max}/p_{min})$. We present all of these results as corollaries of theorems on online matching in an incomplete interval graph.

Some of our algorithms probabilistically select a threshold that is used later in making their decisions. Even though this produces optimal competitive ratios, we show we can use online learning methods to perform nearly as well as the *best* choice of threshold in hindsight, which may be much closer to the offline optimum in certain stationary settings. We also consider incentive compatibility, and develop a nearly optimal incentive-compatible algorithm for maximizing social welfare. Finally, we show how some of our results can be generalized to settings in which the buyers and sellers themselves have online bidding strategies, rather than just each having individual bids.

Chapter 3 Like in Chapter 2, we consider double auctions, but we focus on the motivations of the agents (i.e. we do not assume they will be truthful) in a situation with only one buyer and one seller. The broker wants to maximize his own expected profit, in terms of fees, for the exchange. We show that in this setting, the more the broker can rely on the knowledge the buyer and seller have about the market, the more profit it can make. We analyze the worst-case competitive ratio of the ex ante expected profit of the broker with respect to the ex ante expected profit of the optimal broker aware of the distributions from which the buyer's and seller's values are to be drawn. The best competitive ratio is $\Theta(v_{max})$ if the broker uses a direct mechanism, and $\Theta(\log(v_{max}))$ if it uses a mechanism which for one agent is always truthful. There exists a complex indirect mechanism with a Bayes-Nash equilibrium that obtains as much profit as the optimal broker. We also show a simple, parameterized mechanism that we believe achieves $1 - \epsilon$ of the possible revenue.

Chapter 4 Preference elicitation — the process of asking queries to determine parties' preferences — is a key part of many problems in electronic commerce. For example, a shopping agent needs to know a user's preferences in order to correctly act on her behalf, and preference elicitation can help an auctioneer in a combinatorial auction determine how to best allocate a given set of items to a given set of bidders. Unfortunately, in the worst case, preference elicitation can require an exponential number of queries even to determine an approximately optimal allocation. In Chapter 4 we study natural special cases of preferences for which elicitation can be done in polynomial time via value queries. The cases we consider all have the property that the preferences (or approximations to them) can be described in a polynomial number of bits, but the issue here is whether they can be elicited using the natural (limited) language of value queries. We make a connection to computational learning theory where the similar problem of *exact learning with membership queries* has a long history. In particular, we consider preferences that can be written as *read-once formulas* over a set of gates motivated by a shopping application, as well as a class of preferences we call *Toolbox DNF*, motivated by a type of combinatorial auction. We show that in each case, preference elicitation can be done in polynomial time. We also consider the computational problem of allocating items given the parties' preferences, and show that in certain cases it can be done in polynomial time and in other cases it is NP-complete. Given two bidders with Toolbox-DNF preferences, we show that allocation can be solved via network flow. If parties have read-once formula preferences, then allocation is NP-hard even with just two bidders, but if one of the two parties is additive (e.g., a shopping agent purchasing items individually and then bundling them to give to the user), the allocation problem is solvable in polynomial time.

1.6 Fundamentals of Game Theory

The second part of this thesis discusses the topic of learning in games, that is, the problem of multi-agent interaction from the perspective of the agents rather than the mechanism designer. Before discussing this topic, it is important to understand the fundamentals of game theory. This section contains a brief overview of that field.

1.6.1 Bimatrix Games and Nash Equilibria

In a **bimatrix game**, there are two agents $\{1,2\}$, and each agent i has a set of actions A_i. When a bimatrix game is played, each agent chooses an action $a_i \in A_i$ privately at random. Then, the **joint action** (a_1, a_2) is revealed. The utility of each agent i is represented by a function $u_i : A_1 \times A_2 \to \mathbf{R}$ from the joint action played to a real number representing the "happiness" of the ith agent. An example of bimatrix game is the Prisoner's Dilemma:

Game 1.2: The Prisoner's Dilemma

	D	C
d	-5,-5	-6,0
c	0,-6	-1,-1

In this game, there are two prisoners, 1 and 2, that have been captured committing some minor crime (for which the sentence is one year). The first agent is asked to rat out the second agent for some major crime (with a sentence of five years) they were not caught doing, in exchange for being pardoned on the minor one. The second agent is given the same offer. Now, both prisoners saw such a possibility, so they made a pact to not rat out each other. We assume for the sake of argument that the two agents feel no sympathy for one another, and their utility is the opposite of how many years they will go to jail. When put in separate cells, will they cooperate with each other, and both stay silent, each receiving a one year sentence, or will they defect from their plan and rat each other out, and both receive five years in jail?

The first agent has two actions $\{d, c\}$, either (d)efect or (c)ooperate, indicated on the left side of the table. The second agent has two options, either (D)efect or (C)ooperate, indicated on the top of the table. In each entry of the table is a possible outcome. For instance, if the first agent (d)efects and the second agent (C)ooperates, then the first agent receives a utility of $u_1(d, C) = 0$ (does not go to jail) and the second agent receives a utility of $u_2(d, C) = -6$ (goes to jail for both crimes). This is indicated by the pair $0, -6$ in the top right entry of the table.

Now, consider this from the perspective of the first agent: since the actions of both agents are simultaneous, his action cannot affect the action of the second agent, so the first agent should assume that the second agent's action is fixed. If the second agent will defect, then the first agent achieves $u_1(d, D) = -5$ if it defects and $u_1(c, D) = -6$ if it cooperates. Thus, the first agent would be happier defecting if the second agent defected. Similarly, if the second agent will cooperate, then $u_1(d, C) > u_1(c, C)$. Therefore, the first agent is happier defecting regardless of what happens. Similarly, the second agent is happier defecting regardless of what happens. The pair, (d, D) is called a **Nash equilibrium**, because given the **strategy** (plan of action, possibly involving randomness) of one agent, the strategy of the other agent maximizes expected utility. It is often argued that a Nash equilibrium is the only way that two **rational** agents would play.

Sometimes, there is no pair of deterministic actions that are a Nash equilibrium. The game below is like "Rock-Paper-Scissors", except if an agent loses it receives the same utility as if it had tied.

Game 1.4: Shapley's Game

	R	P	S
r	0,0	0,1	1,0
p	1,0	0,0	0,1
s	0,1	1,0	0,0

For any set S, define $\Delta(S)$ to be the set of all probabilities over S. For a distribution D and a boolean predicate P, we use the notation $\Pr_{x \in D}[P(x)]$ to indicate the probability that $P(x)$ is true given that x

was selected from D, and $D(x)$ to be the probability of x in the distribution D. An agent i may use a **mixed strategy** $\beta_i \in \Delta(A_i)$, where it plays an action at random according to a distribution. $\beta_i(a_i)$ is the probability of choosing a_i while using β_i. Define:

$$u_i(\beta_1, \beta_2) = \sum_{a_1 \in A_1, a_2 \in A_2} \beta_1(a_1)\beta_2(a_2)u_i(a_1, a_2).$$

It is known [Nas50] that for any bimatrix game, there exists a Nash equilibrium $(\beta_1^*, \beta_2^*) \in \Delta(A_1) \times \Delta(A_2)$ such that:

$$\begin{aligned} u_1(\beta_1^*, \beta_2^*) &= \max_{\beta_1 \in \Delta(A_1)} u_1(\beta_1, \beta_2^*) \\ u_2(\beta_1^*, \beta_2^*) &= \max_{\beta_2 \in \Delta(A_2)} u_1(\beta_1^*, \beta_2). \end{aligned}$$

In other words, given the strategy of one agent, the other agent cannot improve its expected utility by changing its strategy. A Nash equilibrium for Shapley's Game is for each agent to choose an action uniformly at random from all its actions. Finally, for each agent i define:

$$\| u_i \| = \max_{(a_1,a_2) \in A_1 \times A_2} u_i(a_1, a_2) - \min_{(a_1,a_2) \in A_1 \times A_2} u_i(a_1, a_2).$$

1.6.2 Correlated Equilibria

Sometimes, when dividing chores, or deciding who has to perform some unpleasant task, people will "draw straws": that is, they each choose a straw from a fist, and whoever gets the shortest straw has to perform the task. Or, they might draw pieces of paper from a hat. Sometimes such an agreement may be *a priori* better for all agents then fighting, or being disorganized, et cetera.

For instance, in Shapley's game, the expected utility of each agent in a Nash equilibrium is 1/3. It would be better if the two agents avoided "ties". One way to do this might be for one agent to play (p)aper, and the other agent to play (s)cissors. They draw straws, and whoever draws the shortest straw plays paper.

Unfortunately, this agreement is somewhat flawed. If the second agent draws the shortest straw, and believes the first agent would play (s)cissors, why wouldn't the second agent play (R)ock? Thus, such an agreement is not self-enforcing, because the agent who draws the shortest straw isn't motivated to comply with the agreement.

Imagine that instead of this **public** randomness, or the **independent private** randomness inherent in mixed strategy Nash equilibria, what if one had a source of **dependent private** randomness. For example, imagine that a **referee**, a third impartial agent, has a bag with six balls, labeled (r, P), (r, S), (p, R),(p, S),(s, R), and (s, P). All the agents know the six balls in the bag. The referee now draws a ball uniformly at random from the bag. It then privately informs the first agent of the first element of the pair on the ball and the second agent of the second element of the pair on the ball.

Observe that the difference between this situation and the one before is that if the first agent is told to play (r)ock, then with equal probability the second agent will have been told to play (S)cissors or (P)aper. Thus, the first agent can do no better than play the action it has been told to play. Observe that if both agents play as they are told, then both agents will have an expected utility of $\frac{1}{2}$.

In general, a **correlated equilibrium** is a joint distribution $D \in \Delta(A_1 \times A_2)$ over the actions of the agents. For any $a_1^* \in A_1$ on which D has positive probability, for any $a_2^* \in A_2$ on which D has positive probability:

$$\begin{aligned} \sum_{a_2 \in A_2} D(a_1^*, a_2)u_1(a_1^*, a_2) &= \max_{a_1 \in A_1} \sum_{a_2 \in A_2} D(a_1^*, a_2)u_1(a_1, a_2) \\ \sum_{a_1 \in A_1} D(a_1, a_2^*)u_1(a_1, a_2^*) &= \max_{a_2 \in A_2} \sum_{a_1 \in A_1} D(a_1, a_2^*)u_1(a_1, a_2). \end{aligned}$$

In other words, if D is the distribution from which the referee draws a joint action, then given that one agent chooses the action recommended by the referee, the other agent can do no better than choose the action recommended by the referee.

Correlated equilibria also have a special significance in learning in multi-stage games. We will discuss this further in the next section.

1.7 Multi-Stage Games

Multi-stage games are an important model in Artificial intelligence and game theory for three reasons. First, they can be used to model the "social consequences" of an action: how the acceptable or unacceptable behavior of one agent can be rewarded or punished by another agent. Secondly, they allow researchers to model how agents could learn about each other by observing each other's actions. Finally, when state is implicitly or explicitly contained inside the model, they allow researchers to model learning how to solve complicated problems where the "consequences" of the actions of an agent are not always immediately apparent.

1.7.1 Repeated Bimatrix Games

Many people find the Nash equilibrium of the Prisoner's Dilemma unsettling. For instance, one could imagine two nations in the grips of a nuclear arms race. Each knows that if it obliterated the other nation, it would be marginally safer. Each would like to strike back if it knew it would be struck. However, the US and the USSR co-existed in such a precarious position for decades without annihilating each other. Is there something wrong with the above analysis?

Observe that to cooperate in the Prisoner's Dilemma is a **strictly dominated action**: that is, regardless of the strategy of the other agent, cooperating never maximizes an agent's expected utility. This implies that cooperating is not in any Nash equilibrium or in any correlated equilibrium.

However, sometimes a situation can be better modeled as a **repeated bimatrix game**: that is, two agents face each other in the same bimatrix game an infinite number of times. For instance, suppose that the US knows that the USSR will retaliate against any nuclear strike today with a strike of its own tomorrow. Also, the US believes there is a high probability that the USSR will not strike tomorrow unless the US strikes today. Thus, it is now in the best interest of the US not to strike today.

Each time a bimatrix game is played is called a **time step**. Formally, one can consider the events in a bimatrix game to form an **infinite history** $h \in H^\infty$, an infinite sequence of joint actions. Define h_i to be the ith joint action of the history h. In a **discounted repeated bimatrix game**, one has a **discount factor** $\gamma \in [0, 1]$, that can sometimes be considered the probability of continuing to the next time step, or as a simple cognitive discounting of rewards that will not be received for some time. Given a discount factor $\gamma \in [0, 1]$ and a history $h \in h^\infty$, the utility of agent i is:

$$u_i^\gamma(h) = \sum_{t=1}^{\infty} \gamma^{t-1} u_i(h_i).$$

The state of the repeated bimatrix game after t time steps can be represented by a finite history $h \in H^t = (A_1 \times A_2)^t$. $H = \bigcup_{t=0}^{\infty} H^t$ is the set of all finite histories, including $\emptyset$, the history of length 0. $|h|$ is the length of history h.

The most important aspect of repeated bimatrix games is that both agents remember the entire history of joint actions up until the last game played and can have their strategy in the current bimatrix game depend upon it. A **behavior** for agent i is a function $\sigma_i : H \to \Delta(A_i)$, a function from the history observed to a strategy to play on the next time step.

In order to consider Nash equilibria in repeated games, we have to define the distribution over histories when two agents play a repeated bimatrix game. Define $h(t)$ to be the first t joint actions of history h. $h_{t,i}$ is the action chosen in the tth time step by agent i. The probability of σ_i playing its part of history $h \in H$

is:

$$P_{\sigma_i}(h) = \prod_{t=1}^{|h|} \sigma_i(h(t-1))(h_{t,i}).$$

In other words, the probability that at each time step t, given the history up until that point, agent i would have chosen the action $h_{t,i}$. The probability that the game begins with h when the behaviors σ_1 and σ_2 are used is $P_{\sigma_1,\sigma_2}(h) = P_{\sigma_1}(h)P_{\sigma_2}(h)$. Define $P^T_{\sigma_1,\sigma_2} \in \Delta(H^T)$ to be the associated probability distribution. The principle can be extended to infinite histories, such that μ_{σ_1,σ_2} is the measure over infinite histories when σ_1 and σ_2 are played, where for all $h' \in H$:

$$\mu_{\sigma_1,\sigma_2}(\{h \in H^\infty : h(t) = h'\}) = P_{\sigma_1,\sigma_2}(h').$$

$h(\sigma_1, \sigma_2)$ is a random variable indicating the history played. Thus, the expected utility of agent i when two agents using the behaviors σ_1 and σ_2 play each other is:

$$u_i^\gamma(\sigma_1, \sigma_2) = E[u_i^\gamma(h(\sigma_1, \sigma_2))].$$

Define Σ_i to be the set of all behaviors for agent i. A Nash equilibrium is a pair of behaviors (σ_1^*, σ_2^*), such that:

$$u_1^\gamma(\sigma_1^*, \sigma_2^*) = \max_{\sigma_1 \in \Sigma_1} u_1^\gamma(\sigma_1, \sigma_2^*).$$

$$u_2^\gamma(\sigma_1^*, \sigma_2^*) = \max_{\sigma_2 \in \Sigma_2} u_2^\gamma(\sigma_1^*, \sigma_2).$$

It is interesting to note that if (β_1^*, β_2^*) is a Nash equilibrium for a single stage of the repeated bimatrix game then the behaviors that always use β_1^* and β_2^* are a Nash equilibrium for the whole repeated bimatrix game. However, there can also be more complicated Nash equilibria that may be more beneficial for all of the agents. For instance, consider the grim trigger behavior, which cooperates until it observes the other agent defect. Then, if both agents use the grim trigger behavior, they will both cooperate ad infinitum. Since defecting at any point for either agent would lower the utility, both agents using a grim trigger behavior is a Nash equilibrium if γ is close to 1. This will yield a discounted utility of $\frac{-1}{1-\gamma}$ for both agents, whereas the equilibrium of both agents always defecting yields a utility of $\frac{-5}{1-\gamma}$. This will be discussed further in Section 6.1.

1.7.2 External Regret

There are four types of regret discussed in this thesis:

- external regret
- internal regret
- external response regret
- internal response regret

We will define the first two here, and define the last two in Chapter 6.

External regret is a measure that can be taken on a history. Imagine two agents are playing Shapley's game, and the history after six time steps is:

$$h = \{(r, R), (p, P), (s, S), (s, P), (r, S), (p, P)\}.$$

One could imagine comparing how well the first agent did to an omniscient agent playing against a second agent who chose the actions $\{R, P, S, P, S, P\}$. However, if the omniscient agent could do anything, then it would get a utility of 1 on every time step. A more fair comparison is to imagine the omniscient agent builds a simple robot, which can perform one action $a_1^* \in A_1$. It is sent down to one time step in the history h

(chosen uniformly at random), and plays its action. Thus, if the omniscient agent built the robot to play p, then in expectation the robot's utility would be $\frac{1}{2}$. In general the utility of the robot would be:

$$\frac{1}{|h|}\sum_{t=1}^{|h|} u_1(a_1^*, h_{t,2}).$$

Now, if the omniscient agent builds a robot with the action with the maximum **a priori** utility (before t is chosen, after the history h is known), then the expected utility of the robot is:

$$\max_{a_1^* \in A_1} \frac{1}{|h|}\sum_{t=1}^{|h|} u_1(a_1^*, h_{t,2}).$$

One can compare this to the utility obtained by the first agent on the same time step:

$$R_1^{ext}(h) = \max_{a_1^* \in A_1} \frac{1}{|h|}\sum_{t=1}^{|h|} \left(u_1(a_1^*, h_{t,2}) - u_1(h_t)\right).$$

This is the **external regret**. A **no-external regret behavior**, is a behavior $\sigma_1 : H \to \Delta(A_1)$ such that for all $\delta > 0$, there exists a time T, such that for every $\sigma_2 : H \to \Delta(A_2)$:

$$\Pr_{h \in \mu_{\sigma_1,\sigma_2}} [\forall t > T, R_1^{ext}(h(t)) < \delta] > \delta.$$

Roughly, as time goes on, the external regret is dropping uniformly, regardless of the second agent's behavior.

1.7.3 Internal Regret

Internal regret is a stronger concept than external regret. Consider the history from the previous section. However, before introducing internal regret, let us consider a more natural concept of regret called swap regret.

Imagine that the omniscient builds another robot: however, instead of having a single action the robot has a function $f : A_1 \to A_1$. Thus, when it arrives at some time step t, before it acts, it gets to see what the first agent did on time step t, and have its own action depend on the action of the first agent. So, when the time step is chosen uniformly at random, the expected utility of the robot is:

$$\frac{1}{|h|}\sum_{t=1}^{|h|} u_1(f(h_{t,1}), h_{t,2}).$$

Thus, if the omniscient builder chooses f to maximize the robot's utility, and this is compared to the utility obtained by the first agent, then the **swap regret** is:

$$R^{swap}(h) = \max_{f} \frac{1}{|h|}\sum_{t=1}^{|h|} \left(u_1(f(h_{t,1}), h_{t,2}) - u_1(h_t)\right).$$

In the history above, the function would be $f(r) = r$, $f(p) = s$, and $f(s) = s$. In internal regret, the omniscient agent chooses a single action a_1^- for the robot to replace, and a single action a_1^* for the robot to play. There is zero contribution to the internal regret on time steps where the first agent did not play a_1^-.

$$R^{int}(h) = \max_{a_1^-, a_1^* \in A_1} \frac{1}{|h|} \sum_{t : h_{t,1} = a_1^-} \left(u_1(a_1^*, h_{t,2}) - u_1(h_t)\right).$$

This is the **internal regret**. In the above example, $a_1^- = p$, $a_1^* = s$. The definition of a **no-internal-regret** behavior is similar to the definition of a no-external-regret behavior.

1.7.4 Internal Regret and Correlated Equilibria

One of the most interesting results in learning in multi-agent systems is that two **no-internal-regret** algorithms will "converge" to the set of correlated equilibria. Particularly, imagine a history h generated after T time steps. Imagine that there was a bag with $|h|$ balls, and the tth ball was labeled with the joint action h_t. If the referee drew a ball from this bag as in Section 1.6.2, then neither agent could significantly improve their utility by choosing a ball different from the one they saw. As T increases, the amount that the agents could improve their utility drops to zero.

Observe that there is a significant difference between the equilibria here and the equilibria discussed in Section 1.6.2. Because the correlated equilibria generated in this process are not necessarily the result of a conscious choice, there is no guarantee that this is a good correlated equilibrium. We will return to this point in Chapter 7.

1.8 Original Results in Learning in Games

Chapter 5 Convex programming involves a convex set $F \subseteq \mathbf{R}^n$ and a convex cost function $c : F \to \mathbf{R}$. The goal of convex programming is to find a point in F which minimizes c. In online convex programming, the convex set is known in advance, but in each step of some repeated optimization problem, one must select a point in F before seeing the cost function for that step. This can be used to model factory production, farm production, and many other industrial optimization problems where one is unaware of the value of the items produced until they have already been constructed. We introduce an algorithm for this domain. We also apply this algorithm to repeated games, and show that it is really a generalization of infinitesimal gradient ascent, and the results in Chapter 5 imply that generalized infinitesimal gradient ascent (GIGA) achieves no-external-regret: in other words, the algorithm almost surely in the limit does almost as well as the best fixed action.

Chapter 6 The concept of regret is designed for the long-term interaction of multiple agents. However, most concepts of regret do not consider even the short-term consequences of an agent's actions: e.g., how other agents may be "nice" to you tomorrow if you are "nice" to them today. For instance, an agent that always defects while playing the Prisoner's Dilemma will never have any internal or external regret. In Chapter 6, we introduce a new concept of regret, called response regret, that allows one to consider both the immediate and short-term consequences of one's actions. Thus, instead of measuring how an action affected the utility on the time step it was played, we also consider the consequences of the action on the next few time steps, subject to the dynamic nature of the other agent's responses: e.g. if the other agent always is nice to us after we are nice to it, then we should always be nice: however, if the other agent sometimes returns favors and sometimes doesn't, we will not penalize our algorithm for knowing when these times are. We develop algorithms for both external response regret and internal response regret, and show how if two agents minimize internal response regret, then they converge to a correlated equilibrium in repeated bimatrix games, stochastic games, and partially observable stochastic games.

Chapter 7 No-regret online learning algorithms are very natural strategies in the context of repeated zero-sum games. E.g., algorithms such as randomized weighted majority, GIGA, and cautious fictitious play guarantee expected payoff comparable to the optimal fixed strategy in hindsight, which is at least the minimax optimum. In a nonzero-sum game, however, achieving good performance depends on both agents being "reasonable". Can an algorithm such as weighted majority guarantee the minimum Nash equilibrium value solely under the assumption that the other agent is using a no-regret algorithm? Can *any* no-regret algorithm? What about weighted majority playing against itself? In Chapter 7, we explore these questions and demonstrate families of games where these conditions are insufficient even to guarantee the minimum correlated equilibrium value. We also show that it is possible for an arbitrary algorithm to obtain the minimum Nash equilibrium value against a no-internal-regret algorithm whose utilities are unknown.

1.9 Summary

In this thesis, we present several new algorithms: some for an auctioneer running a double auction or a combinatorial auction, others for handling online convex programming problems, repeated bimatrix games, or stochastic games. In addition, we clarify some of the issues in auctions and games. In double auctions, there is a direct tradeoff between the complexity of the auction and the profit achievable by the auctioneer. We show how a new type of regret allows for agents to account for how their actions affect the behavior of the other agent. We show that, if the first agent wants to minimize regret, the assumption that the second agent is regret minimizing is insufficient for the first agent to guarantee a reasonable utility.

Part I

Mechanism Design with Incomplete Information

Chapter 2

Algorithms for Online Double Auctions[1]

We study the problem of online market clearing where there is one commodity in the market being bought and sold by multiple buyers and sellers whose bids arrive and expire at different times. The auctioneer is faced with an online clearing problem of deciding which buy and sell bids to match without knowing what bids will arrive in the future. For maximizing *profit*, we present a (randomized) online algorithm with a competitive ratio of $\ln(p_{max} - p_{min}) + 1$, when bids are in a range $[p_{min}, p_{max}]$, which we show is the best possible. A simpler algorithm has a ratio twice this, and can be used even if expiration times are not known. For maximizing the *number* of trades, we present a simple greedy algorithm that achieves a factor of 2 competitive ratio if no money-losing trades are allowed. Interestingly, we show that if the online algorithm is allowed to *subsidize* matches — match money-losing pairs if it has already collected enough money from previous pairs to pay for them — then it can be 1-competitive with respect to the optimal offline algorithm that is not allowed subsidy. That is, the ability to subsidize is at least as valuable as knowing the future. We also consider the objectives of maximizing buy or sell volume, and present algorithms that achieve a competitive ratio of $2(\ln(p_{max}/p_{min}) + 1)$, or $\ln(p_{max}/p_{min}) + 1$ if the online algorithm is allowed subsidization. We show the latter is the best possible competitive ratio for this setting. For social welfare maximization we also obtain an optimal competitive ratio, which is below $\ln(p_{max}/p_{min})$. We present all of these results as corollaries of theorems on online matching in an incomplete interval graph.

Some of our algorithms probabilistically select a threshold that is used later in making their decisions. Even though this produces optimal competitive ratios, we show we can use online learning methods to perform nearly as well as the *best* choice of threshold in hindsight, which may be much closer to the offline optimum in certain stationary settings. We also consider incentive compatibility, and develop a nearly optimal incentive-compatible algorithm for maximizing social welfare. Finally, we show how some of our results can be generalized to settings in which the buyers and sellers themselves have online bidding strategies, rather than just each having individual bids.

2.1 Introduction

Electronic commerce is becoming a mainstream mode of conducting business. In electronic commerce there has been a significant shift to dynamic pricing via *exchanges* (that is, markets with potentially multiple buyers and multiple sellers). The range of applications includes trading in stock markets, bandwidth allocation in communication networks, as well as resource allocation in operating systems and computational grids. In addition, exchanges play an increasingly important role in business-to-business commerce, and several independent business-to-business exchanges have been founded (e.g., ChemConnect). This sourcing trend

[1]This work appeared in [BSZ02].

means that instead of one company buying from multiple suppliers, *multiple* companies will buy from multiple suppliers. In another words, sourcing is moving toward an exchange format.

These trends have led to an increasing need for fast market clearing algorithms [SS01, SS02, SSGL01]. Also, recent electronic commerce server prototypes such as *eMediator* [San02] and *AuctionBot* [WWW98] have demonstrated a wide variety of new market designs, leading to the need for new clearing algorithms.

This chapter analyzes a market for one commodity, for example, DELL stocks, pork bellies, electricity, memory chips, or CPU time. As a running example, let us assume we have a market for Playstations. For simplicity, assume that each buy bid and each sell bid is for a single unit of the commodity. Each bid arrives at a particular point in time and is valid for some given length of time. For example, imagine a seller s arrives at 1:00 PM, interested in selling a Playstation for fifty dollars. A buyer b arrives at 1:30 PM, interested in buying a Playstation for a hundred dollars. An auctioneer (market-maker), an agent who matches buyers and sellers, must decide if it wants to match b and s before 2:30 PM, when b will leave the market. Suppose the auctioneer can pocket the difference when making a match and wants to maximize its profit. What is a good strategy to decide whether or not match b and s without knowing what future buy/sell bids will be? It could be that a second buyer b' will enter the market at 3:00 PM willing to pay \$150 for a Playstation (see Figure 2.1), resulting in more money for the auctioneer. The auctioneer faces the tradeoff of clearing

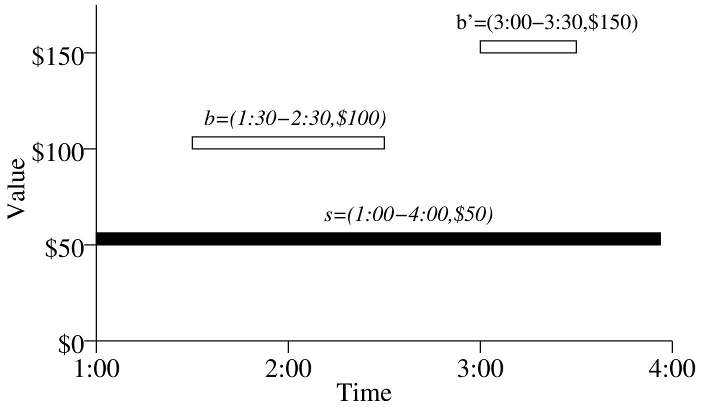

Figure 2.1: In this example, if the auctioneer bought a Playstation from s for 50 dollars at 2:30 and sold it to b for 100 dollars, then it would make a net profit of 50 dollars. But, if the auctioneer waited until 3:30, the auctioneer could make a net profit of 100 dollars.

all possible matches as they arise versus waiting for additional buy/sell bids before matching. Waiting can lead to a better matching, but can also hurt because some of the existing buy/sell bids might expire or get retracted as the bidders get tired of waiting. While the Securities Exchange Commission imposes relatively strict rules on the matching process in securities markets like NYSE and NASDAQ [Dom93], most new electronic markets (for example for business-to-business trading) are not securities markets. In those markets the auctioneer has significant flexibility in deciding which buy bids and sell bids to accept. In this chapter, the focus will be how well the auctioneer can do in those settings, and with what algorithms.

There has been some study using traditional game-theoretic techniques of designing mechanisms that are robust even when the traditional common knowledge assumptions do not apply, such as in [BGM04, BM03, Wil85, Wil87]. Also, there have been a wealth of results using competitive analysis for auctions where the broker is the seller, such as [Har03, LN04, BKRW03].

2.1.1 Problem Definition

The auctioneer can formalize its situation as an online problem in which buy and sell bids arrive over time. When a bid is introduced, the auctioneer learns the bid price and expiration time (though some of the simpler algorithms will not need to know the expiration times). At any point in time, the auctioneer can match a

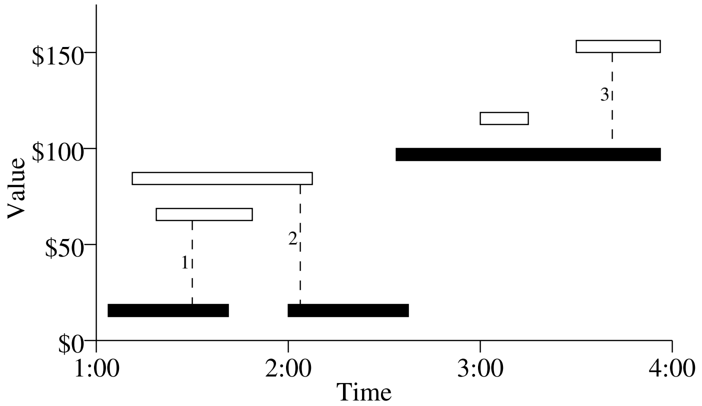

Figure 2.2: Buy bids are represented by white bars, sell bids are represented by black bars, and pairs in the matching by numbered dotted lines. In this example some of the issues facing the auctioneer trying to maximize profit are apparent. Sometimes, it is better to choose a pair like 1 instead of matching the seller to a higher-priced, longer-lasting buy bid, because the longer-lasting bid might be useful later. Also, sometimes it is useful to ignore a buy bid that does not obtain much profit because a more expensive buy bid might come along later, like in pair 3.

live buy bid with a live sell bid, removing the pair from the system. It will be convenient to assume that all bids have integer-valued prices[2](for example, money that cannot be split more finely than pennies) that lie in some range $[p_{min}, p_{max}]$.

Definition 2.1 *A* ***temporal clearing model*** *consists of a set B of buy bids and a set S of sell bids. Each bid $v \in B \cup S$ has a positive price $p(v)$, is introduced at a time $t_i(v)$, and removed at a time $t_f(v)$. A bid v is said to be alive in the interval $[t_i(v), t_f(v)]$. Two bids $v, v' \in B \cup S$ are said to be concurrent if there is some time when both are alive simultaneously.*

Definition 2.2 *A* ***legal matching*** *is a collection of pairs $\{(b_1, s_1), (b_2, s_2), \ldots\}$ of buy and sell bids such that b_i and s_i are concurrent.*

An *offline algorithm* receives knowledge of all buy and sell bids up front. An *online algorithm* only learns of bids when they are introduced. Both types of algorithms have to produce a legal matching, that is, a buy bid and sell bid can only be matched if they are concurrent (see Figure 2.2). As usual in competitive analysis [BEY98], the performance of the online algorithms will be measured against the optimal offline solution for the observed bid sequence.[3]

2.1.2 Objectives

Competitive analysis of auctions (multiple buyers submitting bids to one seller) has been conducted by a number of authors [LN04, GHW01, GHW00, BCG+01]. In this chapter, we present the first competitive analysis of *exchanges* (where there can be multiple buyers and multiple sellers submitting bids). Exchanges are a generalization of auctions — one could view an exchange as a number of overlapping auctions — and they give rise to additional issues. For example, if a seller does not accept a buy bid, some other seller might.

The goal will be to produce online algorithms with optimal competitive ratios for each of the following objective functions:

[2]Technically, it is assumed that the optimal algorithm is incapable of matching two bids which are closer than one unit in value when it is attempting to maximize profit.

[3]This performance measure assumes that actions performed by the auctioneer do not influence the sequence of bids submitted in the future. For instance, one does not worry that if one had acted differently, one might have seen a better sequence of bids. This assumption is relaxed in Section 2.9.

- **Maximize profit.** Each pair of buy and sell bids that are matched produces a profit, which is the difference between the buy price and the sell price. The total profit is the sum of these differences, over all matched pairs. The surplus could be divided in any way among the buyers, sellers, and the auctioneer. If surplus is maximized, there is no way to make all of the parties better off. Offline, the matching that optimizes profit can be found via weighted bipartite matching. We present a (randomized) online algorithm with competitive ratio $\ln(p_{max} - p_{min}) + 1$, which we show is the best possible. A simpler algorithm has ratio twice this, and can be used even if expiration times are not known. These algorithms build on analysis of [EYFKT92] for the one-way-trading problem.[4] We also show how online learning results [LW94, FS96, BB00] can be used to produce algorithms with even stronger guarantees in certain stationary settings.

- **Maximize liquidity.** Liquidity maximization is important for a marketplace for several reasons. The success and reputation of an electronic marketplace is often measured in terms of liquidity, and this affects the (acquisition) value of the party that runs the marketplace. Also, liquidity attracts buyers and sellers to the marketplace; after all, they want to be able to buy and sell.

  We analyze three common measures of liquidity: 1) number of trades, 2) sum of the prices of the cleared buy bids (*buy volume*), and 3) sum of the prices of the cleared sell bids (*sell volume*). Under criterion 1, the goal is to maximize the number of trades made, rather than the profit, subject to not losing money. We show that a simple greedy algorithm achieves a factor of 2 competitive ratio, if no money-losing trades are allowed. This can be viewed as a variant on the on-line bipartite matching problem [KVV90]. Interestingly, we show that if the online algorithm is allowed to *subsidize* matches — match money-losing pairs if it has already collected enough money from previous pairs to pay for them — then it can be 1-competitive with respect to the optimal offline algorithm that is not allowed subsidy. That is, the ability to subsidize is at least as valuable as knowing the future.

  For the problems of maximizing buy or sell volume, we present algorithms that achieve a competitive ratio of $2(\ln(p_{max}/p_{min}) + 1)$ without subsidization. We also present algorithms that achieve a competitive ratio of $\ln(p_{max}/p_{min}) + 1$ with subsidization with respect to the optimal offline algorithm that cannot use subsidies. This is the best possible competitive ratio for this setting.

- **Maximize social welfare.** This objective corresponds to maximizing the good of the buyers and sellers in aggregate. Specifically, the objective is to have the items end up in the hands of the agents that value them the most. We obtain an optimal competitive ratio, which is the fixed point of the equation $r = \ln \frac{p_{max}}{r p_{min}}$. The Greedy algorithm achieves a competitive ratio at most twice this.

Our best algorithms are developed in a more general setting we call the incomplete interval-graph matching problem. In this problem, there are a number of intervals (bids), some of which overlap in time, but only *some* of those may actually be matched (because the auctioneer can only match a buy to a sell, because the prices must be in the correct order, etc.). By addressing this more general setting, one can produce a wide variety of algorithmic results as corollaries.

Throughout most of the chapter, it is assumed that the agents are truthful, that is, they bid their true valuations. Later on in Section 2.8, this assumption is removed.

2.2 An Abstraction: Online Incomplete Interval Graphs

In this section we introduce an abstraction of the temporal bidding problem that will be useful for producing and analyzing optimal algorithms, and may be useful for analyzing other online problems as well.

Definition 2.3 *An incomplete interval graph is a graph $G = (V, E)$, together with two functions t_i and t_f from V to $[0, \infty)$ such that:*

[4] A somewhat related problem is *online difference maximization* [KT99]. However, in that setting, the focus is on the average case (bids arrive in a random order) and the goal is to find a single pair of bids whose difference in *rank* is maximized.

1. *For all $v \in V$, $t_i(v) < t_f(v)$.*
2. *If $(v, v') \in E$, then $t_i(v) \leq t_f(v')$ and $t_i(v') \leq t_f(v)$.*

Call $t_i(v)$ the start time of v, and $t_f(v)$ the expiration time of v. For simplicity, assume that for all $v \neq v' \in V$, $t_i(v) \neq t_i(v')$ and $t_f(v) \neq t_f(v')$.[5]

An incomplete interval graph can be thought of as an abstraction of the temporal bidding problem where the facts that bids come in two types (buy and sell) and have prices attached to them are ignored, and instead there is just a black box "E" that given two bids v, v' that overlap in time, outputs an edge if they are allowed to be matched. The algorithms developed for this generalization will be key subroutines for all the true problems of interest.

Now consider two problems on incomplete interval graphs: the online edge-selection problem and the online vertex-selection problem. In the online *edge-selection* problem, the online algorithm maintains a matching M. The algorithm sees a vertex v at the time it is introduced (that is, at time $t_i(v)$). At this time, the algorithm is also told of all edges from v to other vertices which have already been introduced. The algorithm can now select an edge but only if both endpoints of the edge are alive. Once an edge has been selected, it can never be removed. The objective is to maximize the number of edges in the final matching, $|M|$.

In the online *vertex-selection* problem, the online algorithm maintains a set of vertices W, with the requirement that there must *exist* some perfect matching on W. At any point in time, the algorithm can choose two live vertices v and v' and add them into W so long as there exists a perfect matching on $W \cup \{v, v'\}$. Note that there need not exist an edge between v and v'. The objective is to maximize the size of W. So, the vertex-selection problem can be thought of as a less stringent version of the edge-selection problem in that the algorithm only needs to commit to the *endpoints* of the edges in its matching, but not the edges themselves.

It is easy to see that no deterministic online algorithm can achieve a competitive ratio less than 2 for the edge-selection problem.[6] A simple greedy algorithm achieves this ratio:

Algorithm 2.4 (Greedy) *When a vertex is introduced, if it can be matched, match it (to any one of the vertices to which it can be matched).*

Theorem 2.5 *The Greedy algorithm achieves a competitive ratio of 2 for the edge-selection problem.*

Proof: Consider an edge (v, v') in the optimal matching M^*. Define v to be the vertex which is introduced first, and v' to be the vertex which is introduced second. Then the algorithm will match either v or v'. In particular, if v is not matched before v' is introduced, then v' *will* be matched (either to v or some other vertex). Therefore, the number of *vertices* in the online matching M is at least the number of *edges* in M^*, which means $|M| \geq |M^*|/2$. ∎

For the vertex-selection problem, we show the following algorithm achieves a competitive ratio of 1. That is, it is guaranteed to find (the endpoints of) a maximum matching in G.

Algorithm 2.6 *Let W be the vertices selected so far by the algorithm. When a vertex v is about to expire, consider all the live unmatched vertices v', sorted by expiration time from earliest to latest. Add the first pair $\{v, v'\}$ to W such that there exists a perfect matching on $W \cup \{v, v'\}$. Otherwise, if no unmatched vertex v' has this property, allow v to expire unmatched.*

Theorem 2.7 (Main Theorem) *Algorithm 2.6 produces a set of nodes having a perfect matching M which is a maximum matching in G.*

[5] All our results can be extended to settings without this restriction, and where $t_i(v)$ may equal $t_f(v)$. This is accomplished by imposing an artificial total order on simultaneous events. Among the events that occur at any given time, bid introduction events should precede bid expiration events. The introduction events can be ordered, for example, in the order they were received, and so can the expiration events.

[6] Consider the following scenario: vertex u expires first and has edges to v_1 and v_2. Then, right after u expires, a new vertex w arrives with an edge to whichever of v_1 or v_2 the algorithm matched to u.

The proof of Theorem 2.7 appears in Section 2.11. The core of the proof is to show that if W is the set of selected vertices, then at all times the following invariants hold:

H_1: For any expired, unmatched vertex w, there does not exist any untaken vertex w' such that there is a perfect matching on $W \cup \{w, w'\}$. (An untaken vertex is a vertex that has been introduced but not matched.)

H_2: For any matched vertex w, there does not exist an untaken vertex w' such that there is a perfect matching on $W \cup \{w'\} - \{w\}$ and $t_f(w) > t_f(w')$.

H_3: For any two unexpired vertices $w, w' \in W$, there exists no perfect matching on $W - \{w, w'\}$.

The first invariant says that the algorithm is complete: it lets no matchable vertex expire, and expired vertices do not later become matchable. The second and third invariants say that the algorithm is cautious. The second invariant says that the algorithm favors vertices which expire earlier over those which expire later. The third invariant states that the algorithm only matches vertices that it has to: no subset of the set of vertices chosen has a perfect matching and contains all of the expired vertices[7].

Since an untaken vertex is a vertex which has been introduced but not matched, no untaken vertices exist at the start of the algorithm. Also, W is empty. Therefore, these invariants vacuously hold at the start.

Three events can occur: a vertex is introduced, a vertex expires without being matched, or a vertex expires and is added with some other vertex to W. It is established in Section 2.11 that if all the invariants hold before any of these events, then they hold afterwards as well. If the first invariant holds at the termination of the algorithm, then no pair could be added to the set selected. The augmenting path theorem [Ber57] establishes that the selected set is therefore optimal. A full proof appears in Section 2.11.

2.3 Profit Maximization

In this section we show how the above results can be used to achieve an optimal competitive ratio for maximizing profit.

The first step is to convert the profit maximization problem to an incomplete interval graph problem by choosing some θ to be the minimum profit which the auctioneer will accept to match a pair. So, when translating from the temporal bidding problem to the incomplete interval matching problem, the auctioneer inserts an edge between a concurrent buy bid b and a sell bid s if and only if $p(b) \geq p(s) + \theta$.

The Greedy algorithm (Algorithm 2.4) then corresponds to the strategy: "whenever there exists a pair of bids in the system that would produce a profit at least θ, match them immediately." Algorithm 2.6 attempts to be more sophisticated: first of all, it waits until a bid is about to expire, and then considers the possible bids to match to in order of their expiration times. (So, unlike Greedy, this algorithm needs to know what the expiration times are.) Second, it can choose to match a pair with profit less than θ if the actual *sets* of matched buy and sell bids could have been paired differently in hindsight so as to produce a matching in which each pair yields a profit of at least θ. This is not *too* bizarre since the sum of surpluses is just the sum of buy prices minus the sum of sell prices, and so doesn't depend on which was matched to which.

Define $M^*(G_\theta)$ to be the maximum matching in the incomplete interval graph G_θ produced in the above manner. Then, from Theorem 2.5, the greedy edge-selection algorithm achieves a profit of at least $\frac{1}{2}\theta|M^*(G_\theta)|$. Applying Algorithm 2.6 achieves surplus of at least $\theta|M^*(G_\theta)|$.

So how should the auctioneer choose θ? If θ is deterministically set to 1, then the number of matched pairs will be large, but each one may produce little surplus. If θ is deterministically set any higher than 1, it is possible the algorithm will miss every pair, and have no surplus even when the optimal matching has surplus.

Instead, as in [EYFKT92] and similar to the Classify-and-Randomly-Select approach [LT94, ABF96] (see also [GPS00]), we will choose θ randomly according to an exponential distribution. Specifically, for all

[7] The paraphrasing of this last point is a bit more extreme than the others, but it turns out that if there exists a perfect matching on W and a perfect matching on $W' \subset W$, then there exists two vertices $v, v' \in W - W'$ such that there is a perfect matching on $W - \{v, v'\}$.

$x \in [1, p_{max} - p_{min}]$, let

$$\Pr[\theta \le x] = \frac{\ln(x) + 1}{\ln(p_{max} - p_{min}) + 1},$$

where $Pr[\theta = 1] = \frac{1}{\ln(p_{max} - p_{min}) + 1}$. Observe that this is a valid probability distribution. Let OPT be the surplus achieved by the optimal offline algorithm.

Lemma 2.8 *If θ is chosen from the above distribution, then $\mathbf{E}[\theta|M^*(G_\theta)|] \ge \frac{\text{OPT}}{\ln(p_{max} - p_{min}) + 1}$.*

Corollary 2.9 *The algorithm that chooses θ from the above distribution and then applies Greedy to the resulting graph achieves competitive ratio $2(\ln(p_{max} - p_{min}) + 1)$. Replacing Greedy with Algorithm 2.6 achieves competitive ratio $\ln(p_{max} - p_{min}) + 1$.*

In Section 2.6 we prove a corresponding lower bound of $\ln(p_{max} - p_{min}) + 1$ for this problem.

Proof (of Lemma 2.8): Focus on a specific pair (b, s) matched by OPT. Let $R_\theta(b, s) = \theta$ if $p(b) - p(s) \ge \theta$ and $R_\theta(b, s) = 0$ otherwise. Observe that $\theta|M^*(G_\theta)| \ge \sum_{(b,s)\in \text{OPT}} R_\theta(b, s)$ because the set of pairs of profit at least θ matched by OPT is a legal matching in the incomplete interval graph. So, it suffices to prove that $\mathbf{E}[R_\theta(b, s)] \ge (p(b) - p(s))/(\ln(p_{max} - p_{min}) + 1)$.

To prove this, first observe that for $x > 1$, $\frac{d}{dx} \Pr[\theta \le x] = \frac{1}{x(\ln(p_{max} - p_{min}) + 1)}$. So,

$$\begin{aligned}\mathbf{E}[R_\theta(b, s)] &= \Pr[\theta = 1] + \int_1^{p(b)-p(s)} \frac{x dx}{x(\ln(p_{max} - p_{min}) + 1)} \\ &= \frac{p(b) - p(s)}{\ln(p_{max} - p_{min}) + 1}.\end{aligned}$$

■

One somewhat strange feature of Algorithm 2.6 is that it may recommend matching a pair of buy and sell bids that actually have negative profit. Since this cannot possibly improve total profit, the auctioneer can always just ignore those recommendations (even though the algorithm will think that the auctioneer matched them).

2.4 Liquidity Maximization

The focus of this section is the online maximization of the different notions of liquidity: number of trades, aggregate price of cleared sell bids, and aggregate price of cleared buy bids.

2.4.1 Maximizing the Number of Trades

Suppose that instead of maximizing profit, the auctioneer's goal was to maximize the *number* of trades made, subject to the constraint that each matched pair have non-negative profit. This can directly be mapped into the incomplete interval graph edge-matching problem by including an edge for every pair of buy and sell bids that are allowed to be matched together. So, the greedy algorithm achieves competitive ratio of 2, which is optimal for a deterministic algorithm (see Section 2.6.3).

However, if the online algorithm can subsidize matches (match a pair of buy and sell bids of negative profit if it has already made enough money to pay for them) then the auctioneer can use Algorithm 2.6, and do as well as the optimal solution in hindsight that is not allowed subsidization. Specifically, when Algorithm 2.6 adds a pair $\{b, s\}$ to W, the auctioneer can match b and s together, subsidizing if necessary. The auctioneer knows that it always has enough money to pay for the subsidized bids because of the property of Algorithm 2.6 that its set W always has a perfect matching. The auctioneer is guaranteed to do as well as the best offline algorithm which is not allowed to subsidize, because the offline solution is a matching in the incomplete interval graph.

2.4.2 Maximizing Buy or Sell Volume

A different important notion of liquidity is the aggregate *size* of the trades.

Definition 2.10 *Given a matching M the* ***buy-volume*** *is*

$$\sum_{(b,s)\in M} p(b).$$

The ***sell-volume*** *is*

$$\sum_{(b,s)\in M} p(s).$$

If the auctioneer wishes to maximize buy volume without subsidization, the auctioneer can use an algorithm based on the greedy profit algorithm.

Algorithm 2.11 *Choose a buy price threshold θ at random. Specifically, for all $x \in [p_{min}, p_{max}]$, let*

$$\begin{aligned} \Pr[\theta \le x] &= \frac{\ln(x)+1}{\ln(p_{max}/p_{min})+1} \text{ and let} \\ \Pr[\theta = 1] &= \frac{\ln(p_{min})+1}{\ln(p_{max}/p_{min})+1}. \end{aligned}$$

When a buy bid b is introduced, if $p(b) \ge \theta$, and there exists an untaken, unexpired sell bid that can be matched without subsidy, match them. When a sell bid s is introduced, if there exists an untaken, unexpired buy bid b such that $p(b) \ge \theta$ and the bids can be matched without subsidy, match them.

This algorithm achieves a competitive ratio of $2(\ln(p_{max}/p_{min})+1)$. The proof follows that of Lemma 2.8. If the online algorithm is allowed to use subsidization, then the auctioneer can use Algorithm 2.6 as follows.

Algorithm 2.12 *Choose a buy price threshold θ at random according to the distribution in Algorithm 2.11. Convert the online problem into an incomplete interval graph. For each bid b, insert a vertex with an interval $[t_i(b), t_f(b)]$. If a buy bid b and a sell bid s can be matched without subsidy, and $p(b) \ge \theta$, add an edge between their respective vertices.*

Run Algorithm 2.6 on the constructed graph. If Algorithm 2.6 chooses a buy bid b and a sell bid s, match them. (If $p(b) < p(s)$, then this match involves a subsidy.)

This achieves a competitive ratio of $\ln(p_{max}/p_{min}) + 1$ with respect to the offline algorithm which does not use subsidy. This is the best ratio that can be achieved (the proof is by threat-based analysis similar to that in Section 2.6.1).

Maximizing sell volume is analogous to maximizing buy volume. The best competitive ratio we know without using subsidy is $2(\ln(p_{max}/p_{min})+1)$. The best achievable with subsidy against an offline algorithm not allowed to use subsidy is $\ln(p_{max}/p_{min}) + 1$.

2.5 Maximizing Social Welfare

Maximizing social welfare means maximizing the sum of the valuations of the people who are left with an item, that is, matched buyers and unmatched sellers. If B' is the set of buy bids that were matched, and S' is the set of sell bids that were unmatched, then the term which the auctioneer wishes to maximize is:

$$\sum_{b\in B'} p(b) + \sum_{s\in S'} p(s).$$

Equivalently, if M is the auctioneer's matching, and S is the set of *all* sell bids, then what the auctioneer wishes to maximize is:

$$\sum_{(b,s)\in M} (p(b) - p(s)) + \sum_{s\in S} p(s).$$

Note that the second term cannot be affected by the algorithm (because of the assumption that the bids are predetermined). Furthermore, adding a constant to the offline and online algorithms' objective values can only improve the competitive ratio. Therefore, Corollary 2.9 immediately implies the auctioneer can achieve a competitive ratio of $\ln(p_{max} - p_{min}) + 1$. However, the auctioneer can in fact do quite a bit better, as shown in the following theorem.

Theorem 2.13 *There exists an online algorithm for maximizing social welfare that achieves a competitive ratio r that is the fixed point of the equation:*

$$r \quad = \quad \ln \frac{p_{max}}{r p_{min}}.$$

Furthermore, this is also a lower bound. Note that this competitive ratio is at most $\ln(p_{max}/p_{min})$.

We prove the lower bound in Section 2.6. The basic idea of the algorithm is described here the full description and analysis is in Section 2.13.

The basic idea is to follow the approach used by Lavi and Nisan [LN04] for the case of auctions, and to select a price threshold at random according to a specific distribution. All trades occur at this price: if a buy bid is below the price, it is ignored, and if a sell bid is above the price, it is ignored. Unlike the case of auctions, however, multiple sell bids mean that the auctioneer will also have to decide which unignored sell bids to match. As we will show, one can achieve the optimal competitive ratio by using Algorithm 2.6 together with an appropriate probability distribution on the threshold. In fact, Section 2.13 contains a more general result that applies to suboptimal online matching algorithms as well, such as the Greedy algorithm. This is useful because the Greedy algorithm is incentive-compatible (see Section 2.8.1). Thus the auctioneer can achieve incentive-compatibility while remaining within a factor of 2 of the optimal competitive ratio. For a full analysis, see Section 2.13.

2.6 Lower Bounds on Competitive Ratios

This section establishes that our analysis is tight for these algorithms. Specifically, no algorithm can achieve a competitive ratio lower than $\ln(p_{max} - p_{min}) + 1$ for the profit maximization problem, no algorithm can achieve competitive ratio lower than the fixed point of $r = \ln \frac{p_{max}}{r p_{min}}$ for social welfare, and no *deterministic* algorithm can achieve a competitive ratio better than 2 for the trade volume maximization problem without subsidization. We also show that no randomized algorithm can achieve a competitive ratio better than 4/3 for the trade volume maximization problem without subsidization, though we believe this is a loose bound. Furthermore, on this problem it is impossible to achieve a competitive ratio better than 3/2 without taking into consideration the expiration times of the bids. Also, we prove that the greedy profit-maximizing algorithm does not achieve a competitive ratio better than $2(\ln(p_{max} - p_{min}) + 1)$ (that is, the analysis of this algorithm is tight).

2.6.1 A Threat-Based Lower Bound for Profit Maximization

In this analysis, we prove a lower bound for the competitive ratio of any online algorithm by looking at a specific set of temporal bidding problems. Even if the algorithm knows that the problem is in this set, it cannot achieve a competitive ratio better than $\ln(p_{max} - p_{min}) + 1$. This is very similar to the analysis of the continuous version of the one-way trading problem in [EYFKT92].

For this analysis, assume $p_{min} = 0$. Consider the situation where you have a sell bid at 0 that lasts until the end. First, there is a buy bid at a, and then a continuous stream of increasing buy bids with each new one being introduced after the previous one expired. The last bid occurs at some value y. Define $D(x)$ to be the probability that the sell bid has not been matched before the bid of x dollars expires. Since it is possible there is only one buy bid, if one wanted to achieve a competitive ratio of r, then $D(a) \leq 1 - \frac{1}{r}$. Also, define $Y(y)$ to be the expected profit of the algorithm. The ratio r is achieved if for all $x \in [a, p_{max}]$, $Y(x) \geq x/r$. Observe that $Y'(x) = -D'(x)x$, because $-D'(x)$ is the probability density at x.

Observe that one wants to use no more probability mass on a bid than absolutely necessary to achieve the competitive ratio of r, because that probability mass is better used on later bids if they exist. Therefore, for an optimal algorithm, $D(a) = 1 - \frac{1}{r}$, and $Y(x) = x/r$. Taking the derivative of the latter, $Y'(x) = 1/r$. Substituting, $1/r = -D'(x)x$. Manipulating, $D'(x) = -\frac{1}{rx}$. Integrating:

$$\begin{aligned} D(y) &= D(a) + \int_a^y D'(x)dx \\ &= 1 - \frac{1}{r} - \frac{1}{r} \ln \left| \frac{y}{a} \right| . \end{aligned}$$

For the optimal case, the probability mass should be exhausted just as y approaches p_{max}. Therefore:

$$D(p_{max}) = 1 - \frac{1}{r} - \frac{1}{r} \ln \left| \frac{p_{max}}{a} \right| = 0$$

$$r = \ln \frac{p_{max}}{a} + 1.$$

Thus, setting a to 1, and shifting back by p_{min}, one gets a lower bound of $\ln(p_{max} - p_{min}) + 1$.

2.6.2 Greedy Profit Maximization

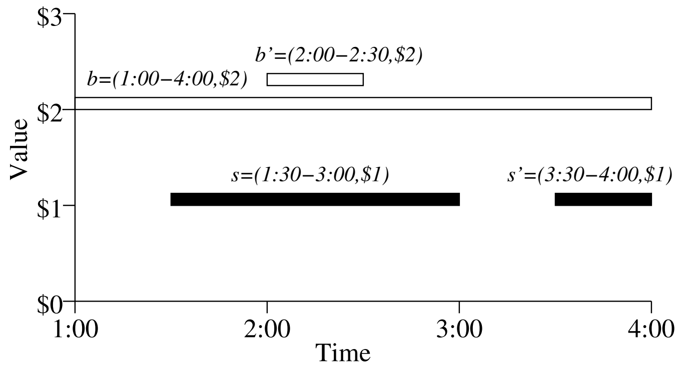

Figure 2.3: On this example, the greedy algorithm achieves a competitive ratio of $2(\ln(p_{max} - p_{min}) + 1)$, by with a very small probability matching b and s.

The following scenario shows that our analysis of the greedy profit algorithm is tight. Imagine that a buy bid for \$2 is introduced at 1:00 (and is good forever), and a sell bid for \$1 is introduced at 1:30 (that is also good forever). At 2:00, another buy bid for \$2 is introduced, which expires at 3:00. At 3:01, another sell-bid for \$1 is introduced. In this scenario, the optimal offline algorithm achieves a profit of \$2 (matching the first buy bid to the last sell bid, and vice versa).

With a probability of $1 - \frac{1}{\ln(p_{max} - p_{min}) + 1}$, $\theta > 1$, and the greedy algorithm ignores all of the bids. Otherwise ($\theta = 1$), the greedy algorithm matches the first two bids for a profit of 1 and then cannot match the second two. Therefore, the expected reward is $\frac{1}{\ln(p_{max} - p_{min}) + 1}$ compared to an optimal of 2.

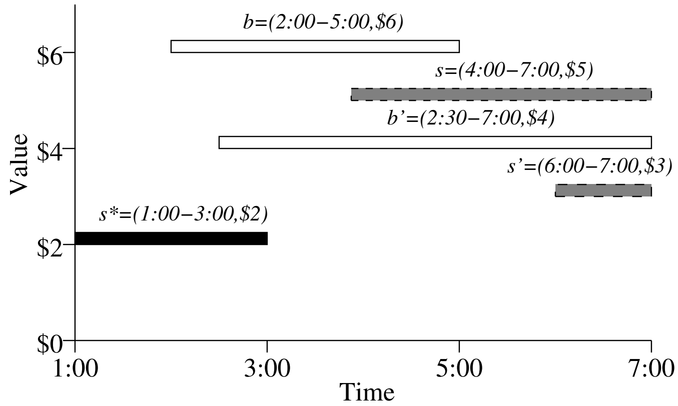

Figure 2.4: Here, either s' or s will arrive, but not both. Therefore, it is impossible to guarantee a competitive ratio higher than 1/2 with a deterministic algorithm.

2.6.3 Lower Bound for Maximizing the Number of Trades

Here we establish that no deterministic algorithm can achieve a competitive ratio lower than 2 for maximizing the number of trades without subsidization. Also, no randomized algorithm can achieve a competitive ratio lower than 4/3. Furthermore, without observing the expiration times, it is impossible to achieve a competitive ratio less than 3/2.

Imagine a sell bid s^* for \$1, is introduced at 1:00 and will expire at 2:00. At 1:01, a buy bid b is introduced for \$3, and will expire at 3:00. At 1:02, a buy bid b' is introduced for \$2, and will expire at 4:00.

There are two possible sell bids that can be introduced: either a sell bid s for \$2.5 at 2:30, or a sell bid s' for \$1.5 at 3:30. Observe that b can match s, and b' can match s'. So if s is to appear, s^* should match b', and if s' is to appear, s^* should match b. But when s^* expires, the online algorithm does not know which one of s and s' will appear. So while a randomized algorithm can guess the correct match to make with a probability of 1/2, the deterministic algorithm must make a decision of which to take before the adversary chooses the example, and so it will choose the wrong match.

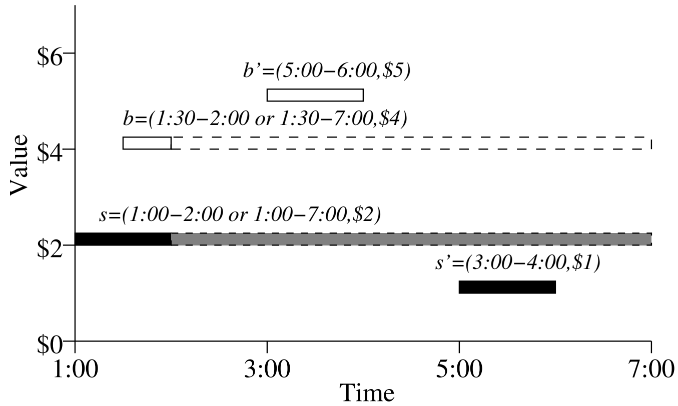

Figure 2.5: Here, b and s will either both expire at 2:00 or both expire at 7:00.

Without observing the expiration times, it is impossible achieve a competitive ratio below 3/2. Imagine that a sell bid is introduced at 1:00 PM for one dollar, and a buy bid is introduced at 1:30 PM for 2 dollars. Should these be matched? Suppose an algorithm A matches them with probability p by 2:00 PM. If $p \leq 2/3$, then they expire at 2:00 PM and no other bids are seen and the expected reward is less than or equal to 2/3 when it could have been 1. If $p > 2/3$, then the bids last all day. Moreover, there is another buy bid from 3:00 PM to 4:00 PM for 4 dollars and another sell bid from 6:00 PM to 6:00 PM for 1 dollar. If the first two bids were unmatched, then there is the possibility of a profit of 2 whereas the expected reward of the algorithm is $(1)p + 2(1-p) \leq 4/3$. Therefore, no algorithm has a competitive ratio better than 3/2 on these two examples. See Figure 2.5.

2.6.4 Lower Bound for Maximizing Social Welfare

Consider the derivation of the price threshold distribution given in Section 2.13. When $\alpha = 1$, then $N^*(T) = N^{alg}(T)$ (see Section 2.13 for definitions). Suppose that the sequence described in Section 2.6.1 is observed. For this sequence, the reward of the algorithm exactly equals $R_T(s)$, where s is the single sell bid. Therefore, the price threshold distribution is optimal for this sequence, and no algorithm[8] can achieve a competitive ratio lower than the fixed point of the equation:

$$r = \ln \frac{p_{max}}{rp_{min}}.$$

2.7 Combining Algorithms Online

The algorithm of Corollary 2.9 begins by picking a threshold θ from some distribution. It turns out that under certain reasonable assumptions described below, the auctioneer can use standard online learning results to do nearly as well as the *best* value of θ picked in hindsight. This does not violate the optimality of the original algorithm: it could be that *all* thresholds perform a log factor worse than OPT. However, one can imagine that in certain natural settings, the best strategy *would* be to pick some fixed threshold, and in these cases, the modified strategy would be within a $(1+\epsilon)$ factor of optimal.

The basic idea of this approach is to probabilistically combine all the fixed-threshold strategies using the Randomized Weighted Majority (also called Hedge) algorithm of [LW94, FS96], as adapted by [BB00] for the case of experts with internal state. In particular, at any point in time, for each threshold θ, one can calculate how well the auctioneer *would* have done had it used this threshold since the beginning of time as a pair $(\mathsf{profit}_\theta, \mathsf{state}_\theta)$, where profit_θ is the profit achieved so far, and state_θ is the set of its current outstanding bids. For example, one might find that had the auctioneer used a threshold of 5, it would have made \$10 and currently have live bids $\{b_1, b_2, s_1\}$. On the other hand, had the auctioneer used a threshold of 1, it would have made \$14 but currently have no live unmatched bids.

In order to apply the approach of [BB00], one needs to be able to view the states as points in a metric space of some bounded diameter D. That is, the algorithm must be able to move from $\mathsf{state}_{\theta_1}$ to $\mathsf{state}_{\theta_2}$ at some cost $d(\mathsf{state}_{\theta_1}, \mathsf{state}_{\theta_2}) \leq D$. This will be true given the following assumption:[9]

Assumption: There is some a priori upper bound B on the number of bids alive at any one time.

We now claim that under this assumption one can view the states as belonging to a metric space of diameter $D \leq Bp_{max}$. Specifically, suppose the overall "master" algorithm tells the auctioneer to switch from threshold θ_1 to θ_2. The auctioneer then conservatively only match buy/sell pairs if they are *both* in $\mathsf{state}_{\theta_2}$ (and have profit at least θ_2). This guarantees that at worst the auctioneer makes B fewer matches than it would have if it were in $\mathsf{state}_{\theta_2}$, and therefore, at worst the auctioneer's profit is Bp_{max} less.

Now someone can plug in the Randomized Weighted-Majority (Hedge) algorithm to get the following theorem.

[8]The competitive ratio of Lavi and Nisan [LN04] is lower than this because they compare their algorithm to the offline Vickrey auction, not the optimal offline algorithm.

[9]This assumption can be weakened a bit and still allow the results to go through.

Theorem 2.14 *Under the assumption above, for any $\epsilon > 0$ (ϵ is given to the algorithm), the auctioneer can achieve an expected gain at least*

$$\max_{\theta} \left[(1-\epsilon)\mathsf{profit}_{\theta} - \frac{2Bp_{max}}{\epsilon} \log N \right],$$

where N is the number of different thresholds.

The proof follows the exact same lines as [BB00] (which builds on [LW94, FS96]). Technically, the analysis of [BB00] is given in terms of losses rather than gains (the goal is to have an expected loss only slightly worse than the loss of the best expert). For completeness, we give the proof of Theorem 2.14 from first principles in Section 2.12.

2.8 Strategic Agents

Up until this point of the chapter, it has been assumed that agents bid truthfully. Each agent has one time interval during which it wants to trade and its valuation stays constant in that interval. This section discusses bidding strategies that are more sophisticated in either of two ways. First, an agent may bid strategically, that is, it might submit a bid that differs from its true valuation. To address this, we present an incentive-compatible algorithm for social welfare maximization. Second, we consider the case of agents whose bid values are allowed to change over time. We show that the Greedy profit maximization algorithm maintains its competitive ratio even in this more difficult setting.

2.8.1 An Incentive-Compatible Mechanism for Maximizing Social Welfare

We design an algorithm that is incentive-compatible in dominant strategies. That is, each agent's best strategy is to bid truthfully regardless of how others bid. Bidding truthfully means revealing one's true valuation. Observe that because in a double auction it is impossible to be efficient in general [MS83], the online VCG mechanism of [FP02] will not work in this domain.

Assume that each bidder wants to buy or sell exactly one item. Also assume that there is one window of time for each agent during which the agent's valuation is valid (outside this window, a seller's valuation is infinite, and a buyer's valuation is negative infinity). For example, the window for a seller might begin when the seller acquires the item, and ends when the seller no longer has storage space. A buyer's window might begin when the buyer acquires the space to store the item. Also assume that within this window, the valuation of the item to the buyer or seller is constant. This is similar to the assumptions made in [Por04] in the online scheduling domain.

First, consider a simpler setting with no temporal aspects. That is, everyone makes bids upfront to buy or sell the item. In the mechanism, the auctioneer first decides a price T at which all trades will occur, if at all. The auctioneer then examine the bids and only consider sell bids below T and buy bids above T. If there are more sell bids than buy bids, then for each buy bid, the auctioneer select a random seller with a valid bid to match it to (this is different from selecting a random sell bid if some sellers submitted multiple bids). The auctioneer does the reverse if there are more buy bids than sell bids.

Now, there are three factors that affect the expected reward of an agent:

1. The agent's valuation for the item. This is fixed before the mechanism begins, and cannot be modified by changing the strategy.

2. The price of a possible exchange. This is set upfront, and cannot be modified by changing the strategy.

3. The probability that the agent is involved in an exchange.

The last factor is the only one that the agent has control over. For a buyer, if its valuation is below the price of an exchange, then the buyer does not wish to exchange, and can submit one bid at its true valuation, and it will have zero probability of being matched.

The more interesting case is when the agent's valuation is above the trading price. Now, the agent wishes to maximize the probability of getting to trade. If the other agents submit at least one fewer buy bid above T than the number of sell bids below T, then the agent is guaranteed a trade. On the other hand, if there are as many or more buy bids than sell bids, then the agent would not be *guaranteed* to purchase an item. Regardless of how many bids the agent submits, it will only have as much of a chance as any other buyer. So, the agent loses nothing by submitting a single bid (at its true valuation).

A similar argument applies to sellers. Therefore, all of the agents are motivated to bid truthfully.[10] An important aspect of this is that the trading price does not depend on the bids.

This technique can be applied to the online problem.

Algorithm 2.15 *Select a price threshold T at random before seeing any bids. Reject any sell bids above T, and any buy bids below T. If at any time, there are more remaining unmatched buy bids than sell bids, then match a buy bid to a random seller (not a random sell bid). If there are more remaining unmatched sell bids than buy bids, then match a sell bid to a random buyer (not a random buy bid). If there are an equal number of remaining unmatched buy and sell bids, match all of them.*

Under this algorithm, each agent is motivated to bid its true price. Furthermore, no agent can benefit from submitting multiple bids or from only bidding for a fraction of its window.

This algorithm is an instance of the Greedy algorithm, because it always matches two bids when it can. For each pair in the optimal matching (where the buy bid is above the price threshold and sell bid is below the threshold), the above algorithm will select at least one bid. If the first bid introduced is not there when the second bid arrives, then it was matched. If the first bid still remains when the second bid arrives, then (since either all buy bids or all sell bids are selected), one bid or the other bid is selected from the pair. This implies that the competitive ratio for the Greedy social welfare maximizing algorithm applies.

2.8.2 Incentive Compatibility for Maximizing Liquidity or Profit

We do not know if there exists an incentive-compatible algorithm that achieves a good competitive ratio for liquidity or profit. Such an algorithm would have to be drastically different from the algorithm described above. Imagine one sets a price of all transactions before any bids are seen (perhaps selecting the price from a distribution). Now suppose that only one buy and sell bid, separated by a dollar, are submitted. Then if and only if the price threshold is between the buy and sell bid will any profit be made. Since the buy bid can be anywhere between p_{max} and $p_{min} + 1$, in the worst case selecting a good price is achieved with probability at most $(p_{max} - (p_{min} + 1))^{-1}$. So, no algorithm of this form can achieve a competitive ratio better than this for either liquidity or profit maximization.

2.9 When Bids Depend on the Algorithm's Actions: Stronger Notions of Competitive Ratio

So far in this chapter we have used the usual framework for competitive analysis, in which the algorithm's performance is compared to the optimal offline algorithm on the *same request sequence*. However, if the event stream (bids) that the algorithm sees may depend on the algorithm's past choices (e.g., think of a bidder who increases his buy bid if he hasn't been matched after a certain amount of time) then this notion does not reflect the ratio of the auctioneer's performance to the best it could have done. In other words, if the auctioneer had behaved differently it might have made a lot more profit because it would have seen a different sequence of bids.[11]

[10] This scheme is vulnerable to coalitions: if for a buyer b and a seller s, $p(b) > p(s) > T$, then b has a motivation to pay s to decrease his offered price to below T, such that b can receive the item. If it is assumed there are no side payments, such problems do not occur.

[11] This is a different (though related) issue than the notion of an oblivious versus adaptive adversary. In the adaptive-adversary model, the bid sequence may depend on the algorithm's actions, but the algorithm's performance is still compared to the optimal in hindsight on the *same* request sequence.

If one allows arbitrary dependence of future bids on the algorithm's actions, any online algorithm will fail miserably under this stronger notion of competitive ratio. To see this, imagine that in the beginning of the sequence, there are several buy bids, and one sell bid. One of the buy bids is special because if it is matched, then there will be many more bids submitted, and if it is not matched, there will be no more bids submitted. The optimal offline algorithm will match this special buy bid and end up making a huge profit, but the online algorithm will not know which one of the buy bids is special.

However, surprisingly there are some natural dependencies that can be handled. Suppose that an agent is interested in buying or selling a single unit, but instead of giving one bid for it, the agent gives several bids over time (at most one bid active at a time) until the agent buys or sells the unit. For example, a buyer might put in a bid for \$100, and then if that bid has not been matched by time t_1, replace that with a bid for \$200. So, if the online algorithm matches the \$100 bid, then it never sees the \$200 one. A natural example of this is plane tickets. Mary wants to go on a flight, but does not want to pay what the flight is worth to her. So she first puts in a low bid, and then gradually increases that bid over time, removing the bid if she does not receive a flight before she has to commit to the trip. Similarly, the airlines might also vary the price of a single seat on a plane over time.

We show that a competitive ratio of $2(\ln(p_{max} - p_{min}) + 1)$ for profit maximization can still be achieved in this setting by using the Greedy algorithm described earlier in this chapter.

The nature of the online graph is different in this model. The auctioneer can still represent the bids as vertices and the matches as edges. If there is a single agent interested in a single unit, only one vertex will be used to represent all of that agent's bids (which differ based on when they are active and on price). This will guarantee that the agent does not buy or sell more than one item. If there is, at some time, a buy bid made by agent b for $pr(b)$ and a sell bid made by agent s for $pr(s)$ where $pr(b) \geq pr(s)$, then an edge from b to s is inserted into the graph. Because a single vertex can represent multiple bids at different prices, there may be different edges between two vertices at different times. Also, two vertices may enter, and at a later time an edge may appear between them.

The algorithm remains the same: the auctioneer chooses a random profit threshold T according to the same distribution as before and only include edges representing trades that achieve profit level at least T. The algorithm selects from the edges greedily as they appear. This is still guaranteed to get a matching at least half as large as optimal. To see this, observe that at least one vertex from every edge in the optimal matching is selected by the algorithm. When an edge arrives, either one of the vertices has already been taken, or both are free. If both are free, they will be matched to each other, or at least one of them will be matched to another edge that happened to be introduced simultaneously. Thus, after an edge is introduced, at least one of its vertices has been selected.

This technique also works for maximizing liquidity. However, maximizing social welfare in this setting is a bit trickier. One problem is that it is not totally clear how social welfare should be defined when the valuations of the participants change with time. If only the *buyers'* valuations change with time, then one natural definition of social welfare for a matching M is

$$\sum_{b \in B'} p_M(b) + \sum_{s \in S'} p(s),$$

where B' is the set of matched buy bids, S' is the set of unmatched sell bids, and $p_M(b)$ is the value of b's bid at the time the match was made. In that case, the Greedy algorithm still achieves its near-optimal competitive ratio. This is perhaps surprising since the online algorithm is at an extra disadvantage, in that even if it magically knew which buyers and sellers to match, it still would need to decide exactly *when* to make those matches in order to maximize social welfare.

2.10 Lemmas on Matchings*

This section contains the proofs of some lemmas required for the proof of the main theorem in Section 2.11. Some standard facts and definitions from graph theory will be used; for a review, see [Koz91]. Paths will be represented as sets of edges, although sometimes it will be easier to write them as a sequence of vertices. $V(P)$ is the set of vertices of a path P. Some well known lemmas will also be used:

Lemma 2.16 *Given $W \subset V$, a perfect matching M on W, and $v, v' \in V - W$, if there exists some augmenting path P with respect to M from v to v', then $M \oplus P$ is a perfect matching on $W \cup \{v, v'\}$.*

Lemma 2.17 *The symmetric difference of two matchings M and M' consists of alternating paths and alternating cycles with respect to M.*

Lemma 2.18 *A vertex $v \in V$ is the endpoint of an alternating path in the symmetric difference of two matchings M and M' if and only if it is in $V(M) \oplus V(M')$.*

Proof: Suppose a vertex is in $V(M) \oplus V(M')$. Then it has an incident edge in either M or M', but not both. Hence, it can only have one incident edge in $M \oplus M'$, making it an endpoint of an alternating path.

Suppose a vertex is at the endpoint of an alternating path in $M \oplus M'$. Then it has one edge in $M \oplus M'$. Observe that if it has an odd number of edges incident to it in $M \oplus M'$, the number of edges in M to it plus the number of edges in M' to it is also odd. But since the latter is bounded by zero and two, it is one. Therefore, the vertex cannot be in both $V(M)$ and $V(M')$. ■

Augmenting paths capture some of the local properties of matchings. But they do not capture how to remove elements from a matching, or how to replace an element. Since these are important concepts in the online setting, we introduce some other types of paths. These paths have properties very similar to those of augmenting paths.

Definition 2.19 *An **abridging path** with respect to a matching M is an alternating path whose first and last edges are in M. A **replacement path** with respect to M is an alternating path whose first edge is in M, and whose last endpoint is not in $V(M)$.*

Lemma 2.20 *Given $v, v' \in W$, and a perfect matching M on W, if there exists some abridging path P with respect to M from v to v', then $M \oplus P$ is a perfect matching on $W - \{v, v'\}$.*

The proof is similar to the proof of Lemma 2.16.

Lemma 2.21 *Given $v \in W$ and $v' \in V - W$, and a perfect matching M on W, if there exists some replacement path P with respect to M from v to v', then $M \oplus P$ is a perfect matching on $W \cup \{v'\} - \{v\}$.*

The proof is similar to the proof of Lemma 2.16.

Lemma 2.22 *Suppose W and W' are subsets of V, and there exists a perfect matching on W and a perfect matching on W', then there exists a partition of $W \oplus W'$ into sets of size two:*

$$\{\{a_1, b_1\}, \{a_2, b_2\}, \ldots, \{a_k, b_k\}\},$$

such that for all $1 \leq i \leq k$, there exists a perfect matching on $W \oplus \{a_i, b_i\}$.

Proof: Define M to be a perfect matching on W and M' to be a perfect matching on W'. Then $M \oplus M'$ consists of a set of alternating paths and cycles. Here, the only concerns are the paths. Since these paths only begin and end at points in $W \oplus W'$, one can partition $W \oplus W'$ where each set is the set of endpoints of a path in $M \oplus M'$. Consider a set in this partition $\{a, b\}$, where P is the path between the two elements of the set. There are three possibilities:

1. Both vertices are in $W' - W$. Then P is an augmenting path, and $M \oplus P$ is a perfect matching on $W \cup \{a, b\} = W \oplus \{a, b\}$.
2. One vertex is in $W - W'$ and one vertex in $W' - W$. In this case, the path P is a replacement path. Without loss of generality, assume $a \in W - W'$. Then $M \oplus P$ is a perfect matching on $W \cup \{b\} - \{a\} = W \oplus \{a, b\}$.

3. Both vertices are in $W - W'$. The first and last edges are not in M', because a and b are not in W'. Also, the first and last edges are in $M \oplus M' \subseteq M \cup M'$. Therefore, the first and last edges are in M, and P is an abridging path. $M \oplus P$ is a perfect matching on $W - \{a, b\} = W \oplus \{a, b\}$.

■

Corollary 2.23 *If $W \subseteq V$ has a perfect matching, but it is not one of the largest sets that has a perfect matching, then there exists $v, v' \in V - W$ such that $W \cup \{v, v'\}$ has a perfect matching.*

2.11 Proof of Theorem 2.7 (Main Theorem)*

Assume W is the set of selected vertices. Recall that if t is the current time, we say that a vertex w is **expired** if $t_f(w) < t$, we say w is unmatched if $w \notin W$, and we say w is untaken if $t_i(w) \leq t$ and $w \notin W$. Define H_1,H_2, and H_3 as follows:

H_1: For any expired, unmatched vertex w, there does not exist any untaken vertex w' such that there is a perfect matching on $W \cup \{w, w'\}$.

H_2: For any matched vertex w, there does not exist an untaken vertex w' such that there is a perfect matching on $W \cup \{w'\} - \{w\}$ and $t_f(w) > t_f(w')$.

H_3: For any two unexpired vertices $w, w' \in W$, there exists no perfect matching on $W - \{w, w'\}$.

Let H denote the conjunction of the three invariants H_1, H_2, and H_3. It will be proven that if these invariants of the algorithm hold before an event occurs, they will hold after the event. The possible events that can occur are a vertex being introduced, expiring without being matched, or expiring and being added.

Lemma 2.24 *If H holds before a vertex $\hat{w}$ is introduced, H holds after the event.*

Proof: The proofs for all three parts of H are below.

H_1: Consider w to be an expired, unmatched vertex. For contradiction, assume that M is a perfect matching on $W \cup \{w, \hat{w}\}$. Define v such that $(\hat{w}, v) \in M$. Observe that $t_f(v) > t_i(\hat{w})$, so v has not expired. Define $M' = M - \{(\hat{w}, v)\}$. Then, M' is a perfect matching on $W - \{v\} + \{w\}$. Since w has expired and v has not expired, $t_f(w) < t_f(v)$. This contradicts the inductive hypothesis H_2.

H_2: Suppose that $w \in W$ and $t_f(w) > t_f(\hat{w})$. Suppose for contradiction there is a perfect matching M for $W \cup \{\hat{w}\} - \{w\}$.

Define v such that $(\hat{w}, v) \in M$. Observe that $t_f(v) > t_i(\hat{w})$, so v has not expired. Neither has w, since $t_f(w) > t_f(\hat{w}) > t_i(\hat{w})$. However, $M - (\hat{w}, v)$ is a perfect matching on $W - \{w, v\}$. This is a contradiction of H_3.

H_3: This cannot have any effect.

■

Lemma 2.25 *If H holds before a vertex w expires without being matched, H holds after the event.*

Proof: Proofs for all three parts of H are below.

H_1: Suppose w' is an untaken vertex. We need to prove that there does not exist a perfect matching on $W \cup \{w, w'\}$. There are two cases:

(a) Assume w' expired. By H_1 (switching the roles of w and w'), we already know there is no perfect matching on $W \cup \{w, w'\}$.

(b) Suppose w' is not expired. Then, if $W \cup \{w, w'\}$ had a perfect matching, then w would have been matched.

H_2,H_3: This cannot have any effect.

■

Lemma 2.26 *If H holds before a vertex v expires and is added with v', H holds after the event.*

Proof: Proof for all three parts of H are below.

H_1: Suppose w is an expired, unmatched vertex, and w' is an untaken vertex. A proof by contradiction is below that there does not exist a perfect matching on $W \cup \{v, v', w, w'\}$.

Observe that there exists a perfect matching on W. If there existed a perfect matching on $W \cup \{v, v', w, w'\}$, then by Lemma 2.22 one of the following conditions holds:

(a) There exists a perfect matching on $W \cup \{v', w\}$. This would contradict H_1.
(b) There exists a perfect matching on $W \cup \{v, w\}$. This would contradict H_1.
(c) There exists a perfect matching on $W \cup \{w, w'\}$. This would contradict H_1.

H_2: Consider an arbitrary vertex w in $W \cup \{v, v'\}$, and an arbitrary untaken vertex w'. Assume that $W \cup \{v, v', w'\} - \{w\}$ is a perfect matching. It must be proven that $t_f(w') \geq t_f(w)$.

(a) $w = v$. Then $W \cup \{v, v', w'\} - \{w\} = W \cup \{v', w'\}$. So by H_1, w' has not expired. Then $t_f(w') \geq t_f(w)$, because w is just expiring.
(b) $w = v'$. Then $W \cup \{v, v', w'\} - \{w\} = W \cup \{v, w'\}$. So by H_1, w' has not expired. Thus if $t_f(w) > t_f(w')$, then w' would have been added instead of w.
(c) $w \in W$. Then by Lemma 2.22 applied to W and $W \cup \{v, v', w'\} - \{w\}$, one of the following conditions holds:
   i. $W \cup \{v\} - \{w\}$ and $W \cup \{v', w'\}$ have perfect matchings. Then by H_1, w' has not expired, and $t_f(w) < t_f(v)$, so w has expired.
   ii. $W \cup \{v'\} - \{w\}$ and $W \cup \{v, w'\}$ have perfect matchings. Then by H_1, w' has not expired, so $t_f(w') \geq t_f(v')$. Also, $t_f(w) \leq t_f(v')$.
   iii. $W \cup \{w'\} - \{w\}$ has a perfect matching. Then by H_2, $t_f(w') \geq t_f(w)$.

H_3: A vertex v expires and is added with v'. Suppose that $w, w' \in W \cup \{v, v'\}$. Assume that $W \cup \{v, v'\} - \{w, w'\}$ has a perfect matching. It must be proven that w or w' has expired. Consider three cases:

(a) $w = v$. Then w has expired.
(b) $w = v'$. Then $W \cup \{v, v'\} - \{w, w'\} = W \cup \{v\} - \{w'\}$. Thus by H_2, $t_f(v) \geq t_f(w')$, so w' has already expired.
(c) $w \in W$. Then by Lemma 2.22 applied to W and $W \cup \{v, v'\} - \{w, w'\}$, one of the following conditions holds:
   i. $W \cup \{v\} - \{w\}$ has a perfect matching. By H_2, $t_f(v) \geq t_f(w)$, so w has expired.
   ii. $W \cup \{v\} - \{w'\}$ has a perfect matching. By H_1, $t_f(v) \geq t_f(w')$, so w' has expired.
   iii. $W - \{w, w'\}$ has a perfect matching. By H_3, either w or w' has expired.

■

Proof (of Theorem 2.7): A proof by induction is below that H holds at the termination of the algorithm. First, observe that H_1, H_2, and H_3 are all properties of introduced vertices, so when there are no vertices introduced, they cannot be violated. This is the base case. Inductively, by Lemmas 2.24, 2.25, 2.26, if these properties hold before an event, they hold afterward as well. Thus, H holds at the termination of the algorithm. Specifically, H_1 implies that there exists no $\{v, v'\} \subseteq V - W$ such that $W \cup \{v, v'\}$ has a perfect matching. By Corollary 2.23, W is one of the largest sets with a perfect matching. ■

2.12 Proof of Theorem 2.14 (Combining Algorithms Online)*

The setting is there are N algorithms ("experts") and the auctioneer can switch from one to another at cost $\leq D$. At each time step (introduction of a bid) the algorithms experience gains between 0 and g_{max}. (we use g_{max} instead of p_{max} because we will be using "p" for probabilities). At any point in time, the overall "master" algorithm will be a probability distribution over the N base algorithms. Specifically, the auctioneer can use the standard multiplicative-weights algorithm [LW94, FS96] as follows:

- Begin with each base algorithm having a weight $w_i = 1$. Let $W = \sum_i w_i$. Our probability distribution over base algorithms will be $p_i = w_i/W$.
- After each event, update the weights according to the rule:

$$w_i \leftarrow w_i(1+\hat{\epsilon})^{g_i/g_{max}},$$

where g_i is the gain of the ith algorithm, and $\hat{\epsilon}$ is defined below. Then update the probability distribution p_i accordingly. Note: if $\vec{p}$ is the old distribution and $\vec{p}'$ is the new distribution, one can do this such that the expected movement cost is at most $\frac{D}{2}L_1(\vec{p},\vec{p}')$ where $L_1(\vec{p},\vec{p}')$ is the L_1 distance between the two distributions.

Theorem 2.27 *If one uses $\hat{\epsilon} = \frac{\epsilon g_{max}}{2D}$, and under the assumption that $D \geq g_{max}$, the above algorithm produces an expected gain at least*

$$\mathrm{OPT}(1-\epsilon) - \frac{2D}{\epsilon}\ln N,$$

where OPT *is the gain of the best of the N base algorithms in hindsight.*

Proof: Suppose the algorithm currently has weights $w_1, \ldots, w_n$, observes a gain vector $g_1, \ldots, g_n$ and then updates the weights to $w_1', \ldots, w_n'$, where $w_i' = w_i(1+\hat{\epsilon})^{g_i/g_{max}}$. Let $W = \sum_i w_i$ and let $W' = \sum_i w_i'$. Let $p_i = w_i/W$ and let $p_i' = w_i'/W'$.

The expected gain of the online algorithm from this event is $\sum_i p_i g_i$, which we will call E_t if this is event t. The expected cost due to moving from probability distribution $\vec{p}$ to distribution $\vec{p}'$ is at most

$$D\sum_{i:p_i'>p_i}(p_i'-p_i) \leq \frac{D}{W}\sum_{i:p_i'>p_i}(w_i'-w_i) \leq \frac{D}{W}(W'-W) = D\left(\frac{W'}{W}-1\right),$$

where the first inequality above uses the fact that $W' \geq W$, and the second inequality uses the fact that $w_i' \geq w_i$ for *all* i, not just the ones such that $p_i' > p_i$.

The next step in the analysis is to upper bound W' as a function of W. To do this, one can use the fact that for $x \leq 1$, $(1+\alpha)^x \leq 1 + x\alpha$. So,

$$W' \leq \sum_i w_i\left(1+\frac{\hat{\epsilon}g_i}{g_{max}}\right) = W\sum_i p_i\left(1+\frac{\hat{\epsilon}g_i}{g_{max}}\right) = W\left(1+\frac{\hat{\epsilon}}{g_{max}}E_t\right).$$

This upper bound on W' is useful in two ways. First, one can now analyze the expected movement cost: $D(W'/W-1) \leq D(\frac{\hat{\epsilon}}{g_{max}}E_t)$. So, the total expected gain for event t is at least $E_t(1-\frac{D\hat{\epsilon}}{g_{max}})$.

Second, one can use this to give an upper bound on the total weight W_{final} at the end of the process:

$$W_{final} \leq N\prod_t\left(1+\frac{\hat{\epsilon}}{g_{max}}E_t\right).$$

This upper bound can be compared to a lower bound

$$W_{final} \geq (1+\hat{\epsilon})^{\mathrm{OPT}/g_{max}},$$

which is simply the final weight of the best of the N algorithms. Taking logs on the upper and lower bounds to W_{final} and using the fact that $\ln(1+x) \in [x - x^2/2, x]$ for $x \in [0,1]$:

$$(\mathrm{OPT}/g_{max})(\hat{\epsilon} - \hat{\epsilon}^2/2) \leq (\mathrm{OPT}/g_{max})\ln(1+\hat{\epsilon}) \leq \ln N + \sum_t \ln(1 + \frac{\hat{\epsilon}}{g_{max}} E_t) \leq \ln N + \frac{\hat{\epsilon}}{g_{max}} \sum_t E_t.$$

So,

$$\sum_t E_t \geq \mathrm{OPT}(1 - \hat{\epsilon}/2) - \frac{g_{max}}{\hat{\epsilon}} \ln N.$$

Finally, using the fact that the total expected gain G_{alg} (with movement costs included) is at least $(1 - \frac{D\hat{\epsilon}}{g_{max}}) \sum_t E_t$, and using the definition $\hat{\epsilon} = \epsilon g_{max}/(2D)$:

$$G_{alg} \geq \mathrm{OPT}(1 - \hat{\epsilon}/2)(1 - \epsilon/2) - (1 - \epsilon/2)\frac{2D}{\epsilon} \ln N \geq \mathrm{OPT}(1 - \epsilon) - \frac{2D}{\epsilon} \ln N.$$

■

2.13 Proof of Bound for Global Welfare (Theorem 2.13)*

Before the algorithm is described, a few definitions must be stated. First, associate utilities from trades with specific sell bids. Define S to be the set of sell bids. Fix an optimal matching M^*. Define $R_{opt} : S \to [p_{min}, p_{max}]$ to be the valuation of the person who received the item from seller s in the optimal matching. If s trades with b, then $R_{opt}(s) = p(b)$. If s does not trade, then $R_{opt}(s) = p(s)$.

For an arbitrary matching M, define $M_{b,\bar{s}}(T)$ to be the number of pairs in M where the buy bid is greater than or equal to T and the sell bid is less than T. Similarly, define $M_{b,s}(T)$ to be the number of pairs in M where the buy bid and sell bid are both greater than or equal to T. Define $M_s(T)$ to be the number of unmatched sell bids greater than or equal to T.

This section focuses on algorithms which work by first selecting a price threshold and then reducing the resulting problem to an incomplete interval graph and applying some algorithm. Suppose that α is the reciprocal of the competitive ratio of the underlying online incomplete interval graph matching algorithm, that is, α is a lower bound on the fraction of potential trades actually harnessed. So, for any optimal online matching algorithm (such as Algorithm 2.6), $\alpha = 1$. For the Greedy algorithm, $\alpha = 0.5$.

Define $N^*(T)$ to be the number of agents (buyers and sellers) that end up with an item in the optimal matching M^*, and have valuation greater than or equal to T. So,

$$N^*(T) = |\{R_{opt}(s) \geq T | s \in S\}| = M^*_{b,\bar{s}}(T) + M^*_{b,s}(T) + M^*_s(T).$$

Let M_{alg} be a matching obtained by an online algorithm. Define $N^{alg}(T)$ to be the number of agents (buyers and sellers) that end up with an item in the matching M_{alg}, and have valuation greater than or equal to T. Then,

$$N^{alg}(T) \geq \alpha M^*_{b,\bar{s}}(T) + M^*_{b,s}(T) + M^*_s(T) \geq \alpha\ N^*(T).$$

The social welfare achieved by the algorithm is at least

$$N^{alg}(T)\ T + (|S| - N^{alg}(T))p_{min} = (N^{alg}(T))(T - p_{min}) + |S|p_{min}.$$

Because $T > p_{min}$, this is an increasing function of $N^{alg}(T)$, so the algorithm achieves a social welfare of at least

$$(\alpha\ N^*(T))(T - p_{min}) + |S|p_{min}.$$

Arithmetic manipulation yields

$$N^*(T)[\alpha T + (1-\alpha)p_{min}] + (|S| - N^*(T))p_{min}.$$

By substitution:

$$|\{R_{opt}(s) \geq T|s \in S\}| \left[\alpha T + (1-\alpha)p_{min}\right] + |\{R_{opt}(s) < T|s \in S\}| \, p_{min}.$$

By defining $R_T(s) = \alpha T + (1-\alpha)p_{min}$ when $R_{opt}(s) \geq T$ and $R_T(s) = p_{min}$ otherwise, the algorithm is guaranteed to achieve a social welfare of at least

$$\sum_{s \in S} R_T(s).$$

So, we have divided the social welfare on a per sell bid basis. In the rest of this section it is described how to design a price threshold distribution so as to maximize the competitive ratio associated with one sell bid in the worst case, similar to that in [EYFKT92]. This will complete the online algorithm for social welfare maximization.

We will attempt to achieve a competitive ratio of r for some arbitrary $r \geq 1$. Define $D(x)$ to be the probability that the threshold is above x. Define $y = R_{opt}(s)$ and $z = R_{alg}(s)$. Observe that if $y = rp_{min}$, then even if $z = p_{min}$, the competitive ratio is achieved. Therefore, $D(rp_{min}) = 1$. Also, there is no need for the threshold to be above p_{max}, so $D(p_{max}) = 0$. Also, define $Y(y)$ to be the expected value of $R_{alg}(s)$ if $R_{opt}(s) = y$. The ratio r is achieved if for all $x \in [rp_{min}, p_{max}]$, $Y(x) \geq x/r$. Note that $Y'(x) = -D'(x)(\alpha x + (1-\alpha)p_{min})$, because $-D'(x)$ is the probability density at x.

Observe that one wants to use no more probability mass on a bid than absolutely necessary to achieve the competitive ratio of r, because that probability mass is better used on later bids if they exist. In other words, the algorithm should hold out as long as possible. Therefore, for an optimal algorithm, $D(rp_{min}) = 0$, and $Y(x) = x/r$. The derivative of the latter is $Y'(x) = 1/r$. Substitution yields $1/r = -D'(x)(\alpha x + (1-\alpha)p_{min})$, so $D'(x) = -\frac{1}{r(\alpha x + (1-\alpha)p_{min})}$. Integration yields

$$\begin{aligned} D(y) &= 1 + \int_{rp_{min}}^{y} D'(x)dx \\ &= 1 - \int_{rp_{min}}^{y} \frac{dx}{r(\alpha x + (1-\alpha)p_{min})} \\ &= 1 - \frac{1}{r\alpha}\left[\ln\left|y + \frac{1-\alpha}{\alpha}p_{min}\right| - \ln\left|rp_{min} + \frac{1-\alpha}{\alpha}p_{min}\right|\right]. \end{aligned}$$

The probability mass should expire as y approaches p_{max}. Therefore,

$$D(p_{max}) = 1 - \frac{1}{r\alpha}\left[\ln\left|p_{max} + \frac{1-\alpha}{\alpha}p_{min}\right| - \ln\left|rp_{min} + \frac{1-\alpha}{\alpha}p_{min}\right|\right] = 0.$$

$$r = \frac{1}{\alpha}\left[\ln\left(p_{max} + \frac{1-\alpha}{\alpha}p_{min}\right) - \ln\left(rp_{min} + \frac{1-\alpha}{\alpha}p_{min}\right)\right].$$

Solving the above equation for r will result in the optimal competitive ratio, that can then be used to calculate the price threshold probability distribution $D(x)$.

So, for Algorithm 2.6,

$$r = \ln p_{max} - \ln rp_{min}$$

For the Greedy algorithm,

$$r = 2\left[\ln(p_{max} + p_{min}) - \ln((r+1)p_{min})\right].$$

In order to better understand this competitive ratio, one can derive an upper bound on it:

$$\begin{aligned} r &\leq \frac{1}{\alpha}\left[\ln\left(p_{max} + \frac{1-\alpha}{\alpha}p_{min}\right) - \ln\left(p_{min} + \frac{1-\alpha}{\alpha}p_{min}\right)\right] \\ &\leq \frac{1}{\alpha}\left[\ln p_{max} - \ln p_{min}\right]. \end{aligned}$$

So, for Algorithm 2.6,

$$r \leq \ln p_{max} - \ln p_{min}.$$

For the Greedy algorithm,

$$r \leq 2\left[\ln p_{max} - \ln p_{min}\right].$$

2.14 Conclusions and Open Questions

In this chapter, we derived bounds on competitive ratios for a number of basic online market clearing problems. These include the objectives of maximizing profit, maximizing liquidity according to several natural measures, and maximizing social welfare. Using the abstraction of online incomplete interval graphs, we obtained algorithms that achieve best-possible competitive ratios for most of these problems, and for the rest we are within a factor of two of the best that is achievable online. For the objective of maximizing number of trades, we demonstrate that by allowing the online algorithm to subsidize matches with profits from previous trades, we can perform as well as the optimal offline algorithm without subsidy. Thus the ability to subsidize trades is at least as powerful as knowing the future, in this context.

It is a somewhat unfortunate fact about competitive analysis that no algorithm can guarantee better than a logarithmic ratio for maximizing profit. This lower bound occurs because the performance of the online algorithm is being compared to an incredible goal: the best performance of any arbitrary algorithm in hindsight. However, if the comparison class is a fixed, finite set of algorithms (such as the class of all fixed-threshold algorithms), and if the data is not too “bursty”, then one can use online learning results to achieve a $1+\epsilon$ ratio, as we demonstrate in Section 2.7.

We also consider the issue of incentive compatibility, and show that the Greedy algorithm is incentive-compatible for the goal of maximizing social welfare. Thus, losing only a factor of 2 in competitive ratio the auctioneer can achieve this desirable property. In fact, we show the Greedy algorithm is quite robust in that it maintains its competitive ratio even if bidders are allowed to modify their bids over time, making the task even harder on the online algorithm.

Open problems

As mentioned in Section 2.8.2, we do not know any incentive-compatible algorithm that achieves good competitive ratio for maximizing liquidity or profit. We know that such an algorithm would need to have a different character than the ones in this chapter.

There is also a gap between the best upper bound (2) and the best lower bound (3/2) for algorithms for the incomplete interval graph vertex-selection problem when *expiration times are unknown.* Algorithm 2.6 achieves an optimal matching but it needs to know the expiration times in advance. Related to this question, we do not know if the Greedy algorithm can be improved upon for the setting of Section 2.9 in which a bidder may submit a bid whose value varies with time.

There has also been recent interest in offline clearing algorithms for *combinatorial* exchanges where a bid can concern multiple distinguishable items (possibly multiple units of each) [San02, SS00, SSGL02]. A bid could state, for example, “I want to buy 20 IBM, buy 50 DELL, sell 700 HP, and get paid $500”. As we transformed the online clearing problem to an online matching problem in an incomplete interval graph, the online problem of maximizing profit or liquidity in a combinatorial exchange can be transformed to the problem of finding a matching on an incomplete interval *hypergraph* online. While we can show with a simple example that a maximum matching cannot be achieved online in the hypergraph problem, it might be possible to achieve a matching with an expected size no less than half of the maximum one.

One direction of future research is to extend these results to settings where there is a broker who can carry inventory of the commodity (sell short and long) rather than simply deciding which bids to match.

Chapter 3

Incentive Compatible Bilateral Trading

Like in Chapter 2, we consider double auctions, but we focus on the motivations of the agents (i.e. we do not assume they will be truthful) in a situation with only one buyer and one seller. The broker wants to maximize his own expected profit, in terms of fees, for the exchange. We show that in this setting, the more the broker can rely on the knowledge the buyer and seller have about the market, the more profit it can make. We analyze the worst-case competitive ratio of the ex ante expected profit of the broker with respect to the ex ante expected profit of the optimal broker aware of the distributions from which the buyer's and seller's values are to be drawn. The best competitive ratio is $\Theta(v_{max})$ if the broker uses a direct mechanism, and $\Theta(\log(v_{max}))$ if it uses a mechanism which for one agent is always truthful. There exists a complex indirect mechanism with a Bayes-Nash equilibrium that obtains as much profit as the optimal broker. We also show a simple, parameterized mechanism that we believe achieves $1 - \epsilon$ of the possible revenue.

3.1 Introduction

Imagine you have a web site, and you want to have people come to your site to buy and sell some item (e.g. Playstations). Suppose that you were unfamiliar with the Playstation market, with the prices for which they were typically bought and sold. What is a good way to make money from the exchange? This is one of the problems studied in the last chapter, but then it was assumed that the agents would bid truthfully. In this chapter, we consider the case that agents bid strategically, based on their knowledge of the distribution the other bidders are drawn from. We assume just a single buyer and a single seller, and consider how an uninformed broker (who does not know the distributions the buyer and seller are drawn from) can design a mechanism that rational buyers and sellers will be motivated to participate in. We will measure how much utility the broker obtains when the agents behave rationally, and compare that to how much utility an informed broker (who does know the distributions) could expect to obtain in the same situation.

Specifically, in this chapter we will consider a single buyer and a single seller attempting to trade a single item through a broker, called a bilateral trade mechanism. Both buyer and the seller have their own private values for the item. Often, we will talk about the **optimal broker**, a broker that knows a lot about the buyer and seller, and given certain restrictions, maximizes its own expected profit. However, we will assume that the brokers we design have minimal knowledge about the buyer and the seller. We will make a minimal assumption that the broker knows a range $[0, v_{max}]$ in which all agents' values for the item will be, but does not know the values or the distributions from which the values were drawn. We will compare our algorithms to the optimal broker that knows the distributions from which the buyer and seller were selected, but not necessarily the values the particular buyer and seller have for the item. We will also make the assumption that the optimal broker can obtain in expectation at least 1 dollar from the buyer and the seller.

We will show how the broker can leverage the knowledge that the buyer and the seller have about the market in order to improve its own revenue. Particularly, we will introduce three mechanisms. The first mechanism is a direct mechanism, and the buyer and seller are both motivated to reveal their true values for the item. This should be easy for them to do, but the broker will not make much profit. The second mechanism is a partially direct mechanism, where the buyer is motivated to reveal its true value. The seller has a more complex task, and thus must know something about the buyer to perform optimally. The broker obtains a reasonable amount in the partially direct mechanism. Finally, in the last mechanism, which is an indirect mechanism, both agents will have to figure out a "reserve price" for the other agent. In this mechanism, both agents will need to determine a complex strategy: however, the broker will obtain almost as much revenue as a broker would who knew from what distributions the buyer and seller were drawn. We show that each of these mechanisms is within a constant factor of the optimal competitive ratio in the mechanisms' respective settings. However, the fundamental reason that these results are interesting is because that the optimal competitive ratios in these settings are all drastically different from each other.

In Section 3.2, we will go over the formalisms required to study this problem, and the definition of competitive ratio used. In Section 3.3, we will introduce the mechanisms we will study in this chapter and give the results about their performance, as well as the optimal performance of each type of mechanism. In the remainder of the chapter, we will prove the results, and then conclude.

3.2 Formal Setting

In this section, we describe the setting formally. We will model the buyer and seller as traditional quasilinear agents, and when the distributions are unknown, restrict ourselves to mechanisms that are ex-interim incentive compatible and ex-interim individually rational for any pair of distributions for the buyer and the seller (see below for definitions). The formalism here is based on that in [Mye81].

3.2.1 Model of the Agents

The standard risk-neutral, quasilinear model will be used for the agents in this chapter. Define y_1 to be the total amount of money received by the seller. Assume that the seller has some type v_1, the seller's value of the item. Assume the seller has a quasilinear preference: that is, the utility of the seller is $y_1 - v_1$ if there is a trade, and y_1 if there is no trade.

The buyer is modeled similarly. Define y_2 to be the total amount of money paid by the buyer. Assume that the buyer has some type v_2, the buyer's value of the item. Assume the buyer has a quasilinear preference: that is, the utility of the buyer is $v_2 - y_2$ if there is a trade, and $-y_2$ if there is no trade.

Without loss of generality, assume that $0 \leq v_1 \leq v_{max}$ and $0 \leq v_2 \leq v_{max}$. Assume that the seller and buyer were each drawn from a distribution: that is, the value v_1 was drawn from some cumulative distribution $F_1 : [0, v_{max}] \rightarrow [0, 1]$, and the value v_2 was drawn from some cumulative distribution $F_2 : [0, v_{max}] \rightarrow [0, 1]$. Define f_1 and f_2 to be the associated probability density functions. For each i, assume that there is some subinterval $[a_i, b_i] \subseteq [0, v_{max}]$, called the **support** of F_i, where f_i is finite and strictly positive, and outside of which f_i is zero. This is so that previous results derived in [Mye81] and [MS83] will be valid.

In addition to such distributions, we will consider distributions where there exists some t_1 such that $\Pr[v_1 = t_1] = 1$ and/or some t_2 such that $\Pr[v_2 = t_2] = 1$. These can be considered as the limit of a sequence of distributions of the above form (like the Dirac δ-function). Often, worst-case analysis will be easiest with such distributions. We will describe all expectations in terms of the density functions, and the discrete analogs of these should be clear. Although before the mechanism begins the value v_1 is known only to the seller, and v_2 is known only to the buyer, it is assumed that F_1 and F_2 are known to the buyer and the seller, but not to the broker. Assume that the buyer and seller are **risk-neutral**, in that at any moment, they choose the action which maximizes their expected gain.

Often we will consider the situation at three key times: **ex ante**, before the agents types are drawn, **ex interim**, after the agents know their own types, but before the mechanism begins, and **ex post**, after the mechanism has completed.

3.2.2 Model of the Mechanisms

A mechanism is basically an algorithm involving input from rational agents. Often, it can be represented by simply stating what each of the agents will do at each step. Let Θ_1 be the set of all strategies for the seller. Let Θ_2 be the set of all strategies for the buyer. Let $\Delta([0, v_{max}])$ be the set of all distributions of the form described above. Formally, a **mechanism** for a bilateral trade can be represented by five functions:[1]

1. $p : \Theta_1 \times \Theta_2 \rightarrow [0, 1]$, the probability that there will be a trade dependent on the strategies of the buyer and the seller,
2. $x_1 : \Theta_1 \times \Theta_2 \rightarrow \mathbf{R}$, the expected[2] amount of money received by the seller,
3. $x_2 : \Theta_1 \times \Theta_2 \rightarrow \mathbf{R}$, the expected payment of the buyer,
4. $\sigma_1 : [0, v_{max}] \times \Delta([0, v_{max}])^2 \rightarrow \Theta_1$, a "suggested" strategy for the seller to use, given its value and the distributions over buyers and sellers,[3] and
5. $\sigma_2 : [0, v_{max}] \times \Delta([0, v_{max}])^2 \rightarrow \Theta_2$, a "suggested" strategy for the buyer to use.

Given $\theta_1 \in \Theta_1$ is used by the seller and $\theta_2 \in \Theta_2$ is used by the buyer, then the seller receives in expectation $x_1(\theta_1, \theta_2)$, the buyer pays in expectation $x_2(\theta_1, \theta_2)$, and the trade occurs[4] with probability $p(\theta_1, \theta_2)$. Thus, the expected values of the agents in this case are:

$$\begin{aligned} U_1(t_1, \theta_1, \theta_2) &= x_1(\theta_1, \theta_2) - t_1 p(\theta_1, \theta_2) \\ U_2(t_2, \theta_1, \theta_2) &= t_2 p(\theta_1, \theta_2) - x_2(\theta_1, \theta_2) \end{aligned}$$

If F_1 is the distribution of the seller, F_2 is the distribution of the buyer, v_1 is the value of the seller, and v_2 is the value of the buyer, then the mechanism suggests the seller will use a strategy $\sigma_1(v_1, F_1, F_2)$ and the buyer will use a strategy $\sigma_2(v_2, F_1, F_2)$. Will the agents be motivated to participate in the mechanism and use these strategies? One can evaluate the expected value of participating in the mechanism for each agent given that the other agent is using the strategy σ_i with a value drawn from F_i:

$$\begin{aligned} U_1(t_1, \theta_1, F_1, F_2) &= \int_0^{v_{max}} U_1(t_1, \theta_1, \sigma_2(s_2, F_1, F_2)) f_2(s_2) ds_2 \\ U_2(t_2, \theta_2, F_1, F_2) &= \int_0^{v_{max}} U_2(t_2, \sigma_1(s_1, F_1, F_2), \theta_2) f_1(s_1) ds_1. \end{aligned}$$

Consider the motivations of the buyer and seller. A mechanism is **ex-interim individually rational** if for all $F_1 \in \Delta([0, v_{max}])$, for all $F_2 \in \Delta([0, v_{max}])$, for all possible v_1 in the support of F_1, and all possible v_2 in the support of F_2, for all $\theta_1 \in \Theta_1$, and all $\theta_2 \in \Theta_2$:

$$\begin{aligned} U_1(v_1, \sigma_1(v_1, F_1, F_2), F_1, F_2) &\geq 0 \\ U_2(v_2, \sigma_2(v_1, F_1, F_2), F_1, F_2) &\geq 0. \end{aligned}$$

Thus, the seller and the buyer are both motivated to participate. A mechanism is **ex-interim incentive-compatible** if for all $F_1, F_2, \theta_1, \theta_2, v_1, v_2$:

$$\begin{aligned} U_1(v_1, \sigma_1(v_1, F_1, F_2), F_1, F_2) &\geq U_1(v_1, \theta_1, F_1, F_2) \\ U_2(v_2, \sigma_2(v_1, F_1, F_2), F_1, F_2) &\geq U_2(v_2, \theta_2, F_1, F_2). \end{aligned}$$

[1] If we were concerned with the ex post results of the mechanism, then we would need to be more precise, considering not just the expected payment and money received but also the correlation between the actual payment and the probability that there will be a trade.

[2] There may be randomness due to either the buyer, the seller, or the mechanism itself.

[3] For example, in a direct mechanism, $\sigma_1(v_1, F_1, F_2) = v_1$.

[4] These expectations are not conditioned on whether the trade occurs. Of course, the actual payments will depend on whether or not a trade occurs.

Thus, if they play the game, the seller and the buyer are motivated to follow the strategies specified by σ_1 and σ_2, respectively. Observe that these properties must hold regardless of the distributions F_1 and F_2. This is stronger than assumptions traditionally made in the economics literature. However, since the broker is unaware of the distributions of the buyer and seller, it must make its mechanism compatible with any distributions.

The broker's **ex-post expected revenue** is:

$$U_0(\theta_1, \theta_2) = x_2(\theta_1, \theta_2) - x_1(\theta_1, \theta_2).$$

The broker's **ex ante expected revenue** is:

$$U_0(F_1, F_2) = \int_0^{v_{max}} \int_0^{v_{max}} U_0(\sigma_1(s_1, F_1, F_2), \sigma_2(s_2, F_1, F_2)) f_2(s_2) f_1(s_1) ds_2 ds_1.$$

In a simple class of mechanisms, called **direct mechanisms**, $\sigma(v_1, F_1, F_2) = v_1$ and $\sigma(v_2, F_1, F_2) = v_2$. The **revelation principle** states, given fixed F_1 and F_2 and an indirect mechanism $(p, x_1, x_2, \sigma_1, \sigma_2)$ there exists a direct mechanism $(p', x_1', x_2', \sigma_1', \sigma_2')$ such that for all v_1 in the support of F_1, and all v_2 in the support of F_2:

$$\begin{aligned} x_1'(v_1, v_2) &= x_1(\sigma_1(v_1, F_1, F_2), \sigma_2(v_2, F_1, F_2)) \\ x_2'(v_1, v_2) &= x_2(\sigma_1(v_1, F_1, F_2), \sigma_2(v_2, F_1, F_2)) \\ p'(v_1, v_2) &= p(\sigma_1(v_1, F_1, F_2), \sigma_2(v_2, F_1, F_2)). \end{aligned}$$

Thus, given the values of the buyer and the seller, the mechanism simply implements the strategies σ_1 and σ_2. In economics this principle allows researchers to focus solely on direct mechanisms. However, *the revelation principle assumes that the distributions of the buyer and seller are known to the broker.* In the model discussed here, this is not the case. Thus, more complex mechanisms must be analyzed for which the revelation principle does not hold.

3.2.3 The Optimal Offline Mechanism and the Competitive Ratio

In this chapter, we will compare our performance to the **optimal offline mechanism**. In other words, the mechanism that, given knowledge of F_1 and F_2, maximizes the ex ante expected utility of the broker, U_0^*. The constraints of incentive compatibility and individual rationality are identical to before. Now, the optimal mechanism for maximizing the broker's revenue can be implemented, as it is described in [MS83].

In order to make it possible to achieve a finite competitive ratio for partial and indirect mechanisms, we will insist that $U_0^*(F_1, F_2) \geq 1$. Given an online mechanism M where the utility of the broker is $U_0(F_1, F_2)$, the worst-case competitive ratio is:

$$CR(M) = \max_{F_1, F_2 \in \Delta([0, v_{max}])^2 : U_0^*(F_1, F_2) \geq 1} \frac{U_0^*(F_1, F_2)}{U_0(F_1, F_2)}$$

This represents how many times more revenue the optimal restricted mechanism achieves than the mechanism we design. We attempt to minimize this number, based on v_{max}.

3.3 Descriptions of the Mechanisms

The key observation of this chapter is that the only parts of the mechanism capable of observing F_1 and F_2 are σ_1 and σ_2. However, σ_1 and σ_2 are implemented by the seller and buyer. The extent to which the agents can, in effect, help implement the mechanism is the degree to which the broker can leverage their knowledge of the market to obtain more profit. This effect can be studied by looking at three different types of mechanisms:

1. Direct mechanisms, where $\sigma_1(v_1) = v_1$ for all $v_1 \in [0, v_{max}]$, and $\sigma_2(v_2) = v_2$ for all $v_2 \in [0, v_{max}]$. Thus, σ_1 and σ_2 are trivial for the buyer and the seller to construct, and they need no knowledge of the other agent's distribution in order to implement the mechanism.

2. Partially direct mechanisms, where either $\sigma_1(v_1) = v_1$ for all $v_1 \in [0, v_{max}]$, or $\sigma_2(v_2) = v_2$ for all $v_2 \in [0, v_{max}]$. Thus, for example, if $\sigma(v_2) = v_2$ for all $v_2 \in [0, v_{max}]$, then the seller only needs to worry about the complexities of the broker and the distribution of the buyer, not the buyer's strategy.

3. Indirect mechanisms, where σ_1 and σ_2 can be arbitrary functions. Thus, both agents must consider the strategy of the other agent when planning their own strategies.

These three types of mechanisms capture the various degrees of strategizing required of the buyer and seller by the broker. For all three types, we have developed mechanisms that have competitive ratios that are within a constant factor of the optimal worst-case competitive ratio. However, the optimal competitive ratio for each of these domains is drastically different. In this section, we will introduce a mechanism for each domain, and prove that two of them are ex-interim incentive compatible and ex-interim individually rational.[5]

3.3.1 Direct Mechanisms

An example of a direct mechanism is a double reserve price mechanism. Observe that the parameter Z is known to all agents in advance.

Mechanism 3.1: Double Reserve Price Mechanism
Parameters: A joint probability distribution over $[0, v_{max}]^2$, $Z \in \Delta([0, v_{max}]^2)$.
Protocol:

1. The seller states a bid t_1.
2. The buyer states an offer t_2.
3. The broker selects z_1, z_2 jointly at random according to Z.
4. If $z_1 \geq t_1$ and $z_2 \leq t_2$, the buyer pays z_2 to the broker, and the broker pays z_1 to the seller, and the item is traded.

Seller's Suggested Strategy: Set $t_1 = v_1$.
Buyer's Suggested Strategy: Set $t_2 = v_2$.

Before considering the choice of Z, let us see why the mechanism is ex interim incentive compatible. If the buyer bids its true value $t_2 = v_2$, is the seller motivated to bid $t_1 = v_1$? Consider the four possible outcomes:

1. $v_2 < z_2$. Regardless of the bid of the seller, there will be no trade.

2. $v_2 \geq z_2$ and $v_1 > z_1$. There will be no trade if the seller is truthful. However, the seller is not interested in trading the item for z_1. Therefore, it can do no better than reveal its true price.

3. $v_2 \geq z_2$ and $v_1 = z_1$. The seller is indifferent to trading the item for z_1.

4. $v_2 \geq z_2$ and $v_1 < z_1$. The seller wants to trade, and will do so if it is truthful.

Thus, regardless of the type of the buyer or the values z_1, z_2, the seller can do no better than reveal its true type. Also, observe that in each case, the seller's value is nonnegative. Thus, it is motivated to participate in the mechanism. The argument that the mechanism is individually rational and incentive-compatible for the buyer is similar.

[5]The indirect mechanism is more complex, and the motivations of the agents in it are discussed in Section 3.4.3

Now, let us consider Z. In order to obtain a competitive ratio linear in v_{max}, choose z_1 uniformly at random from $[0, v_{max} - 1/4]$, and Choose z_2 to be $z_1 + 1/4$.

Theorem 3.2 *If the prices for the double reserve price mechanism are selected in the above fashion, then the competitive ratio is no more than* $16(v_{max} - 1/4)$.

This is not an especially good competitive ratio, but it is not possible to do much better with a direct mechanism.

Theorem 3.3 *It is not possible to achieve a competitive ratio less than* $\frac{\lfloor v_{max} \rfloor + 1}{3}$ *with a direct mechanism.*

The interpretation is clear: no online direct mechanism obtains much revenue. The proof of Theorem 3.2 is in Section 3.4.1, and the proof of Theorem 3.3 is in Section 3.5.1.

3.3.2 Partially Direct Mechanisms

We introduce the following mechanism, where the buyer only needs to reveal its value for the item, but the seller needs to know the prior on the buyer's value to maximize its utility. Assume the function $F_0 : [0, v_{max}] \to [0, 1]$ is known to all agents in advance (a good choice of F_0 is discussed later in this section). Define

$$\tilde{F}_2(x) = 1 - \lim_{\epsilon \to 0^-} F_2(x + \epsilon).$$

This is the probability that $v_2 \geq x$.

Mechanism 3.4: Posting Fee Mechanism
Parameters: A right-continuous function, $F_0 : [0, v_{max}] \to [0, 1]$.
Protocol:

1. The buyer tells the broker in private t_2.
2. The broker selects a posting fee f according to F_0 such that $\Pr[f \leq x] = F_0(x)$.
3. The broker reveals the posting fee.
4. The seller has the option to terminate the mechanism at this point.
5. If the seller chooses to continue, it can state a price q and pay the broker f.
6. If $t_2 \geq q$, then a trade takes place, and the buyer pays the seller q.

Seller's Suggested Strategy: Terminate mechanism if $f > \max_{q \in [0, v_{max}]} \tilde{F}_2(q')(q' - v_1)$.
Otherwise, choose $q \in \mathrm{argmax}_{q' \in [0, v_{max}]} \tilde{F}_2(q')(q' - v_1)$.
Buyer's Suggested Strategy: Set $t_2 = v_2$.

A good choice of F_0 is described later in this section. This mechanism is a little more difficult to analyze, because it could be the case that the seller has a negative utility at the end. However, a risk-neutral seller will be willing to take that risk in order to benefit from the mechanism if its expected utility is nonnegative.

First, we prove that an optimal strategy for the buyer is $t_2 = v_2$, regardless of the strategy of the seller. Consider the four possible outcomes:

1. The seller terminates the auction early. The buyer's bid is then irrelevant.
2. If $t_2 < q$, then no trade takes place. The buyer would not be interested in the item at a price of q, so revealing its true price is optimal.
3. If $t_2 = q$, then the buyer is indifferent to trading.

4. If $t_2 > q$, then a trade takes place. The buyer wants to trade if $v_2 > q$, so revealing its value was the optimal strategy.

Thus, the buyer is motivated to bid its true value. Clearly, its utility is nonnegative.

We will analyze the strategy for the seller from the end to the beginning, so that we can understand the effects of the strategy of the seller near the end of the game on the earlier choices of seller at the beginning. The expected utility of the seller if it accepts the fee and bids q is $\tilde{F}_2(q)(q - v_1) - f$. Thus, in order to maximize its profit, the seller should select a price

$$q^* = \operatorname*{argmax}_{q \in [0, v_{max}]} \tilde{F}_2(q)(q - v_1).$$

Consider when the seller terminates the mechanism early. If the mechanism is terminated early, the seller's expected value is zero. If the seller does not terminate the mechanism, it receives:

$$\max_{q \in [0, v_{max}]} \tilde{F}_2(q)(q - v_1) - f.$$

Define $R(v_1, F_2) = \max_{q \in [0, v_{max}]} \tilde{F}_2(q)(q - v_1)$. There are three possibilities:

1. If $f > R(v_1, F_2)$, then the seller will terminate the mechanism early.
2. If $f < R(v_1, F_2)$, then the seller is motivated to allow the mechanism to continue.
3. If $f = R(v_1, F_2)$, then the seller is indifferent. Assume that in this case the seller allows the mechanism to continue.

Thus, an optimal strategy for the seller is:

1. Terminate the mechanism early if and only if $f > R(v_1, F_2)$.
2. Choose a price $\operatorname{argmax}_{q \in [0, v_{max}]} \tilde{F}_2(q)(q - v_1)$.

This strategy and the truthful strategy for the buyer form a Bayesian-Nash equilibrium by construction. Observe that the expected utility of each agent given its own value for the item is at least zero. One interesting point raised by this mechanism is that the seller may, after the mechanism is completed, wish that it hadn't participated, i.e. the mechanism is not ex-post individually rational. But the seller is still motivated to participate in the mechanism *given all the information it has when the mechanism begins*. Thus, in running such a mechanism, risk-neutral agents would be motivated to participate, so the mechanism is ex-interim incentive compatible.

In order to obtain a good competitive ratio for this mechanism, we use the distribution we developed in Section 2.3.

Choose some $\epsilon \in (0, 1)$. For all $x \in [\epsilon, v_{max}]$, define:

$$F_0(x) = \frac{\ln(x) - \ln \epsilon + 1}{\ln(v_{max}) - \ln \epsilon + 1}.$$

Theorem 3.5 *The posting fee mechanism with the above distribution achieves a competitive ratio of*

$$\frac{\ln(v_{max}) - \ln \epsilon + 1}{1 - \epsilon}.$$

Theorem 3.6 *No partially direct mechanism can achieve a worst-case competitive ratio of less than* $\ln(v_{max}) + 1$.

This bound is fairly tight: the difference is based on the fact that the optimal broker may at times achieve a value below 1, even if its expected revenue is 1. The proof of Theorem 3.5 is in Section 3.4.2, and the proof of Theorem 3.6 is in Section 3.5.2.

3.3.3 Indirect Mechanisms

We introduce two indirect mechanisms. One achieves a competitive ratio of 1, and the other attempts to achieve a lower communication complexity.

Optimal Online Mechanism

One way that a mechanism designer can truthfully extract something that is known to everyone but the mechanism designer is for the mechanism designer to ask everyone, and then compare their answers.

Mechanism 3.7: Optimal Online Mechanism
Parameters: none
Protocol:

1. The buyer states privately to the broker F_1' and F_2'.
2. The seller states privately to the broker F_1'' and F_2''.
3. If $F_1' \neq F_1''$ or $F_2' \neq F_2''$ then the mechanism terminates.
4. Otherwise, the broker runs the optimal mechanism of [MS83] with F_1' and F_2' as inputs for the distributions of the buyer and the seller, maximizes the broker's profit subject to individual rationality and incentive compatibility given $F_1 = F_1'$ and $F_2 = F_2'$.

Seller's Suggested Strategy: State $F_1'' = F_1$, $F_2'' = F_2$, and be truthful during the final step.
Buyer's Suggested Strategy: State $F_1' = F_1$, $F_2' = F_2$, and be truthful during the final step.

Theorem 3.8 *There exists an online indirect mechanism that achieves as much revenue in ex ante expectation as the optimal broker.*

Proof: Observe that since the optimal offline mechanism is individually rational, if both agents reveal the same distributions, the utility is non-negative. Neither agent has anything to gain if it lies about F_1 and F_2 while the other agent tells the truth about F_1 and F_2, because its utility will then be zero. ■

One drawback of this mechanism is that it may be difficult or impossible for both agents to communicate their beliefs to the mechanism. Also, not all Bayesian Nash equilibria for this mechanism achieve an optimal utility. Suppose that the seller's value is 0 with a probability 1 and the buyer's value is 1 with a probability 1. Then the optimal mechanism achieves a profit of 1. However, in the above mechanism, there exists a second Nash equilibrium. Suppose that the agents both state that $F_1(x) = F_2(x) = x$. Then the broker implements a mechanism where the agents trade at a price of $1/2$, and the broker receives no profit. Therefore, through a very weak collusion, the mechanism breaks down. Thus, we present a different mechanism that may do well in this setting.

Profit-Sharing Mechanism

We introduce the following mechanism, where the broker shares a small fraction α of its profit (or loss) with some randomly chosen agent. α is known to all agents in advance.

Mechanism 3.9: α Profit-Sharing Mechanism
Parameters: A fraction $\alpha \in (0,1)$.
Protocol:

1. The broker chooses some $\alpha \in (0,1)$
2. The seller states r_2 and t_1.
3. The buyer states r_1 and t_2.
4. The mechanism flips a fair coin.
5. If the coin is heads, then $z_1 = r_1$ and $z_2 = r_2 - \alpha(r_2 - r_1)$.
6. If the coin is tails, then $z_2 = r_2$ and $z_1 = r_1 + \alpha(r_2 - r_1)$.
7. If $z_1 \geq t_1$ and $z_2 \leq t_2$, then there is a trade, the buyer pays the broker z_2, and the broker pays the seller z_1.

This mechanism attempts to mimic the mechanism in [MS83] for maximizing the broker's revenue. In their mechanism, the broker selects a reserve price z_1 for the seller based on the value of the buyer and a reserve price z_2 for the buyer based on the value of the seller. However, the broker needs to know F_1 and F_2 to properly implement the mechanism in [MS83].

If α was infinitesimally small, then the seller would really care about $r_1 - z_1$. Only when it is indifferent to this direct profit does the seller care about the profit of the broker. However, it is motivated to maximize the profit of the broker when it is indifferent to this direct profit. Thus, the broker is influencing the agents to implement the optimal auction for it. The constraints of incentive compatibility and individual rationality are identical to before. Now, the optimal mechanism for maximizing the broker's revenue can be implemented, as it is described in [MS83].

For indirect mechanisms, we have the following two results that suggest that α profit-sharing mechanisms may be near optimal for sufficiently small α.

Theorem 3.10 *Given that* $\Pr[v_1 = 0] = 1$*, then the α profit-sharing mechanism achieves at least* $1 - \alpha$ *of the optimal revenue, for a competitive ratio of* $\frac{1}{1-\alpha}$.

Theorem 3.11 *If F_1 and F_2 have a positive probability on only a finite number of values, then there exists an α^* such that for all $\alpha \leq \alpha^*$, the α profit-sharing mechanism achieves at least $1-\alpha$ of the optimal revenue.*

A Pareto-optimal equilibrium is an equilibrium where there exists no equilibrium where the buyer and the seller both achieve more revenue in expectation. The open problem is:

Open Problem 3.12 *Does there exist a pair F_1, F_2, and a Pareto-optimal equilibrium σ_1, σ_2, such that the α profit-sharing mechanism receives in ex ante expectation less than $1 - \alpha$ of the optimal broker's revenue?*

We conjecture that no such Bayes-Nash equilibrium exists.

Conjecture 3.13 *If the buyer and the seller choose a Pareto-optimal equilibrium, then the α profit-sharing mechanism receives in ex ante expectation at least $1 - \alpha$ of the optimal broker's revenue.*

3.4 Proofs of Upper Bounds*

The following property of expectations will be useful in analyzing partially direct and partially indirect mechanisms. Define $I(x)$ to be the indicator function, to be 1 if x is true and 0 if x is false.

Lemma 3.14 *For any random variable X where $\mathbf{E}[X] \geq 1$, for all $\epsilon \in [0,1]$,*

$$\mathbf{E}[XI(X \geq \epsilon)] > (1-\epsilon)\mathbf{E}[X]$$

Proof: Observe that

$$\mathbf{E}[XI(X \geq \epsilon)] + \mathbf{E}[XI(X < \epsilon)] = \mathbf{E}[X]$$

However, it is the case that:

$$\mathbf{E}[XI(X < \epsilon)] < \epsilon < \epsilon\mathbf{E}[X]$$

The lemma follows directly. ∎

This lemma means that if one ignores "small profits" less than ϵ, the remaining profit of the optimal broker will be a large portion $(1-\epsilon)$ of its total profit.

3.4.1 Direct Mechanism

Lemma 3.15 *The optimal auction mechanism has an expected revenue of less than*

$$U_0^*(F_1,F_2) \leq \int_0^{v_{max}} \int_0^{v_{max}} (s_2 - s_1) I(s_2 \geq s_1) f_2(s_2) f_1(s_1) ds_2 ds_1$$

Proof: Define

$$\begin{aligned} U_1^*(F_1,F_2) &= \int_0^{v_{max}} U_1^*(s_1, \sigma_1^*(s_1), F_1, F_2) f_1(s_1) ds_1 \\ U_2^*(F_1,F_2) &= \int_0^{v_{max}} U_2^*(s_2, \sigma_2^*(s_2), F_1, F_2) f_2(s_2) ds_2 \end{aligned}$$

The possibility for profit in the mechanism comes from the difference between the value of the buyer and the value of the seller. Observe that

$$\begin{aligned} U_0(F_1,F_2) + \quad & U_1(F_1,F_2) + U_2(F_1,F_2) \leq \\ & \textstyle\int_0^{v_{max}} \int_0^{v_{max}} (s_2 - s_1) I(s_2 \geq s_1) f_2(s_2) f_1(s_1) ds_2 ds_1 \end{aligned}$$

Also, $U_1(F_1,F_2) \geq 0$ and $U_2(F_1,F_2) \geq 0$, because the optimal broker is individually rational. Thus, the bound holds. ∎

Proof (of Theorem 3.2): First, observe from the previous lemma:

$$U_0^*(F_1,F_2) \leq \int_0^{v_{max}} \int_0^{v_{max}} (s_2 - s_1) I(s_2 \geq s_1) f_2(s_2) f_1(s_1) ds_2 ds_1$$

Observe that $U_0^*(F_1,F_2) \geq 1$, so the right side is greater than or equal to 1 as well. By Lemma 3.14 with $\epsilon = 1/2$:

$$U_0^*(F_1,F_2) \leq 2\int_0^{v_{max}} \int_0^{v_{max}} (s_2 - s_1) I(s_2 \geq s_1 + (1/2)) f_2(s_2) f_1(s_1) ds_2 ds_1$$

Given $v_2 \geq v_1 + (1/2)$, there is a probability of $\frac{v_2 - v_1 - (1/4)}{v_{max} - 1/4}$ that $z_1 \geq v_1$ and $z_2 \leq v_2$. Also, $v_2 - v_1 - (1/2) \geq (v_2 - v_1)/2$. Thus, when $v_2 \geq v_1 + (1/2)$, the expected value is greater than $\frac{1}{4}\frac{v_2 - v_1}{2(v_{max} - 1/4)}$, because if $z_1 \geq v_1$ and $z_2 \leq v_2$, then mechanism gets a revenue of $1/4$. Therefore:

$$U_0(F_1,F_2) \geq \int_0^{v_{max}} \int_0^{v_{max}} \frac{s_2 - s_1}{8(v_{max} - 1/4)} I(s_2 \geq s_1 + (1/2)) f_2(s_2) f_1(s_1) ds_2 ds_1.$$

The statement of the theorem follows directly. ∎

3.4.2 Partially Direct Mechanisms

Before analyzing the posting fee mechanism, consider a related mechanism, called a reserve price mechanism.

Reserve Price Mechanism

A reserve price mechanism is of the form:

Mechanism 3.16: Reserve Price Mechanism
Parameters: none
Protocol:

1. The buyer privately sends its type t_2 to the broker.
2. The seller selects a price $q^* = R(v_1, F_2)$.
3. If $t_2 \geq q^*$, then the trade takes place, and the buyer pays q^* to the seller.

By an argument similar to that in Section 3.3.2, the mechanism is incentive compatible and individually rational.

Lemma 3.17 *The reserve-price mechanism maximizes the profit of the seller.*

This is a simple case of the result in Section 6 of [Mye81].

Lemma 3.18 *Given the seller's value is v_1, the expected profit obtained by the seller in the reserve price mechanism is no less than the expected profit of the optimal broker.*

Corollary 3.19

$$U_0(F_1, F_2) \leq \int_0^{v_{max}} R(s_1, F_2) f_1(s_1) ds_1.$$

Proof: To see why this is the case, consider the mechanism for the optimal broker. Suppose that the broker was aware of v_1 without having to extract it from the seller. Then the broker would be capable of achieving the same profit as the seller, by buying the item from the seller at v_1 if a trade were to occur. ■

The Posting Fee Mechanism

Proof (of Theorem 3.5): By the last corollary, the expectation of $R(v_1, F_2)$ is greater than $U_0^*(F_1, F_2)$. Thus:

$$\mathbf{E}[R(v_1, F_2) I(R(v_1, F_2) \geq \epsilon)] \geq (1 - \epsilon) U_0^*(F_1, F_2).$$

Suppose that $R(v_1, F_2) = R$ for some $R \in [\epsilon, v_{max}]$. If the broker chooses a fee f greater than R, it receives no revenue. If the broker chooses a fee f less than or equal to R, it receives a revenue of f. This is exactly the domain for which the distribution in Section 2.3 was designed. Normally, the lower bound of the distribution would be 1. However, ϵ can be easily scaled up to 1. Due to the scaling, the upper bound on the distribution becomes $\frac{v_{max}}{\epsilon}$, the logarithm of which is $\ln v_{max} - \ln \epsilon$. Thus, we can achieve a fraction $\frac{1}{\ln v_{max} - \ln \epsilon + 1}$ of $R(v_1, F_2)$, and at least a fraction $\frac{1-\epsilon}{\ln v_{max} - \ln \epsilon + 1}$ of $U_0^*(F_1, F_2)$ in expectation. ■

3.4.3 Indirect Mechanisms

Lemma 3.20 *In the α profit-sharing mechanism, regardless of the strategies of the agents with respect to r_1 and r_2, it is always a dominant strategy for the seller to declare $t_1 = v_1$ and for the buyer to declare $t_2 = v_2$.*

Proof: This is analogous to the derivation of the incentive compatibility in the double reserve price mechanism. ■

Deterministic Seller

Lemma 3.21 *If* $\Pr[v_1 = 0] = 1$*, and the seller is using the strategy* $t_1 = v_1$*, then in equilibrium the buyer uses the strategy where* $t_2 = v_2$ *and* $r_1 = 0$.

Proof: First, without loss of generality one can assume $t_1 = v_1$ and $t_2 = v_2$ from the previous lemma. Consider the two values of the coin.

1. The coin is tails. Then the buyer's only concern with regards to r_1 is to maximize the chance of a trade. If $r_1 = 0$, then the seller will be willing to trade.
2. The coin is heads. Then $t_1 \leq z_1$ if and only if $r_1 \geq 0$. If the agents trade, then the lower the value of r_1, the more revenue the buyer achieves. Also, the chance of trade is might be increased when r_1 is decreased (so long as it is still nonnegative). Thus, $r_1 = 0$ is an optimal choice.

Therefore, $r_1 = 0$ is the optimal choice. ∎

Proof (of Theorem 3.10): Observe that the optimal broker's revenue is

$$U_0^*(F_1, F_2) = R(0, F_2) = \max_x x\tilde{F}_2(x).$$

By the above lemmas, $r_1 = 0$, $v_1 = t_1$, and $v_2 = t_2$. The seller earns nothing if the coin is heads. If the coin is tails, then the seller's expected profit is $\alpha r_2 \tilde{F}_2(r_2)$. Thus, the seller chooses $r_2 = \text{argmax}_{r\in[0,v_{max}]} \tilde{F}_2(r)$.

Returning to the situation where the coin is heads, the buyer will agree to a trade if $v_2 \geq r_2(1-\alpha)$. Thus, the expected revenue of the broker is:

$$\begin{aligned} U_0(F_1, F_2) &= (1/2)r_2(1-\alpha)\tilde{F}_2(r_2(1-\alpha)) \\ &\quad +(1/2)r_2(1-\alpha)\tilde{F}_2(r_2) \\ &\geq (1-\alpha)r_2\tilde{F}_2(r_2) \end{aligned}$$

From the way the seller selects r_2, one can reason:

$$\begin{aligned} U_0(F_1, F_2) &\geq (1-\alpha)R(0, F_2) \\ &\geq (1-\alpha)U_0^*(F_1, F_2) \end{aligned}$$

∎

Discrete Distributions

In order to understand how the α-profit sharing algorithm works on a discrete distribution, the performance of the optimal broker must first be analyzed.

The optimal broker is basically a combination of two optimal auctions: the optimal auction for some imaginary seller given the distribution of the buyers, and an optimal reverse auction for some imaginary buyer given the distribution of the sellers. One way of defining if the optimal broker decides to trade is if there is some value v such that an imaginary seller with a valuation of v running an optimal auction could trade with the real buyer, and an imaginary buyer running an optimal reverse auction could trade with the real seller. We will go deep into the proof for this in the discrete case in Section 3.6, but for now we will just describe what this means formally.

Assume that the seller has possible values $\{s_1, \ldots, s_m\} = S$, where for all $i \in \{1, \ldots, m-1\}$, $s_i < s_{i+1}$. The buyer has possible values $\{b_1, \ldots, b_n\} = B$, where for all $j \in \{1, \ldots, n-1\}$, $b_j < b_{j+1}$. The probability that the seller has a value $s \in S$ is $f_1(s)$, and the probability that the buyer has a value $b \in B$ is $f_2(b)$. Suppose there is some arbitrary mechanism. By the revelation principle, given that the broker knows the distributions f_1 and f_2, the broker can construct an equivalent mechanism that simply requests the types of the seller and buyer, and has the same probability of trade for every pair of types, and the same distribution over expected payments for every pair of types.

Define $p(s_i, b_j)$ to be the probability of a seller of type i trading with a buyer of type j. Define $x_1(s_i, b_j)$ to be the payment given to the seller. Define $x_2(s_i, b_j)$ to be the payment taken from the buyer. Observe that:

$$U_0 = \sum_{s \in S, b \in B} (x_2(s,b) - x_1(s,b)) f_1(s) f_2(b)$$

is the profit of the broker. Define

$$\begin{array}{cc} \bar{x}_1(s) = \sum_{b \in B} x_1(s,b) f_2(b) & \bar{x}_2(b) = \sum_{s \in S} x_2(s,b) f_1(s) \\ \bar{p}_1(s) = \sum_{b \in B} x(s,b) f_2(b) & \bar{p}_2(b) = \sum_{s \in S} p(s,b) f_1(s) \\ U_1(s) = \bar{x}_1(s) - s\bar{p}_1(s) & U_2(b) = b\bar{p}_1(b) - \bar{x}_2(b) \end{array}$$

The mechanism is ex interim incentive-compatible if for all $s, s' \in S$:

$$U_1(s) \geq \bar{x}_1(s') - s\bar{p}_1(s')$$

and for all $b, b' \in B$:

$$U_2(b) \geq b\bar{p}_2(b') - \bar{x}_2(b').$$

The mechanism is ex interim individually rational if for all $s \in S$, $b \in B$:

$$U_1(s) \geq 0 \text{ and } U_2(b) \geq 0.$$

Thus, the expected gains from trade for agents of any type is nonnegative.

Now, subject to these constraints, we can derive the optimal broker. We begin by thinking about an imaginary seller who wants to run an auction with the buyers. Informally, we will define $(-\infty, C_2(b)]$ to be the set of valuations of sellers who could trade with b as part of an optimal auction. Since we need to prove this is the form of that set, we begin with a little groundwork. Define $F_1(s_i) = \sum_{\hat{i}=1}^{i} f_1(s_{\hat{i}})$, and $F_2(b_j) = \sum_{\hat{j}=1}^{j} f_2(s_{\hat{j}})$. As before, an imaginary seller with valuation $s \in (-\infty, b_n]$ would choose a reservation price in the set:

$$R_2(s) = \operatorname*{argmax}_{v \in B} (1 - F_2(v) + f_2(v))(v - s).$$

This set is nonempty. We can think of a buyer $b \in [s_1, +\infty)$ which could hold a reverse reserve-price auction with a reserve price in the set:

$$R_1(s) = \operatorname*{argmax}_{v \in S} (1 - F_1(v) + f_1(v))(v - s).$$

One can consider the set of all imaginary sellers who could trade with a buyer b:

$$T_2(b) = \{s \in (-\infty, b_n] : b \geq \min_{b' \in R_2(s)} b'\}.$$

Now, observe that sellers with a negative value have been included. Thus, for any b, there is a seller with a sufficiently small value such that it would be willing to trade. Also, there is an upper bound. Thus, one can find the least upper bound, or supremum:

$$C_2(b) = \sup_{s \in T_2(b)} s.$$

Moreover, the set $T_2(b)$ is an interval $(-\infty, C_2(b)]$, as we prove here. Observe that if an imaginary seller s is willing to trade with b, then an imaginary seller with a lower valuation would also be willing to trade. $C_2(b)$ is in $T_2(b)$ because imaginary sellers with valuations arbitrarily close to $C_2(b)$ are in $T_2(b)$, so it must be an optimal choice for $C_2(b)$ to choose a reserve price below b as well[6]. For the seller, define T_1 and C_1 similarly.

$$T_1(s) = \{b \in (-\infty, s_1] : s \leq \min_{s' \in R_2(s)} s'\}.$$

[6]If one fixes the holdout price to be b', the utility of an imaginary seller is a continuous function of s. Thus, if one continuously increases the utility of the imaginary seller from a value s' where it had an optimal utility of b' to a point s'' where it had an optimal utility of b'', there was some point s''' when both b' and b'' were optimal.

$$C_1(s) = \inf_{b \in T_1(s)} b.$$

It is again the case that $T_1(s) = [C_1(s), +\infty)$. It will be helpful to notice that both C_1 and C_2 are increasing functions, because of the nature of reserve price auctions. Now, remember that there should be a trade between s and b if there exists a valuation v such that an imaginary seller with value v could trade with s in the optimal auction, and an imaginary buyer with value v could trade with b in the optimal auction. In other words, s and b should trade if and only if $T_1(s) \cap T_2(b) \neq \emptyset$, i.e. if $C_1(s) \leq C_2(b)$. The optimal broker is of the form:

Mechanism 3.22: Optimal Discrete Mechanism
Parameters: none
Protocol:

1. The seller privately reveals s.
2. The buyer privately reveals b.
3. Compute $C_1(s)$ and $C_2(b)$.
4. If $C_1(s) > C_2(b)$, then no trade occurs.
5. Otherwise, set $x_1 = \max_{t_1 \in S : C_1(t_1) \leq C_2(b)} t$.
6. Set $x_2 = \min_{t_2 \in B : C_2(t_2) \geq C_1(s)}$.
7. Trade the item.

Seller's Suggested Strategy: State $s = v_1$.
Buyer's Suggested Strategy: State $b = v_2$.

Lemma 3.23 *Mechanism 3.22 is incentive compatible.*

Proof: Observe that, given an arbitrary bid s' of the seller, then there exists a value b' such that if the buyer gives a value greater than or equal to b', it will trade at b', and if it gives a value less than b', it will not trade. Thus, regardless of the action of the seller, it is motivated to reveal its true value. Similarly, the seller is motivated to reveals its true value. ■

Theorem 3.24 *The above mechanism achieves the optimal revenue for the broker.*

The proof is in Section 3.6.

For all $s \in S$, define $M_2(s)$ to be the price a buyer pays to trade with a seller of type s, i.e. the minimum $b \in B$ that trades with s. If no buyer trades with s, define $M_2(s)$ to be b_n. Define $M_1(b)$ to be the price a seller pays to trade with a buyer of type b. If no seller trades with b, define $M_1(b)$ to be s_1. Observe that M_1 and M_2 are monotonically increasing functions.

The following lemma helps to clarify the meaning of M_1 and M_2.

Lemma 3.25

$$\begin{aligned} M_1(b) &\in \operatorname*{argmax}_{s' \in S}(C_2(b) - s')(F_1(s')) \\ M_2(s) &\in \operatorname*{argmax}_{b' \in B}(b' - C_1(s))(f_2(b') + 1 - F_2(b')) \end{aligned}$$

Now, this setting is ideal for the α profit-sharing mechanism, because given fixed f_1 and f_2, the broker can make α small enough such that any gains from profit-sharing are dominated by ordinary gains.

Define $r_2 : S \to \mathbf{R}$ such that $r_2(s)$ is the declaration of r_2 of a seller of type s. Define $r_1 : B \to \mathbf{R}$ such that $r_1(b)$ is the declaration of r_1 of a buyer of type b. Assume that the agents bid their values for t_1 and t_2.

So, before continuing, describe how small α must be. First define the diameter of the sets $D(S, B) = \max_{b \in B} b - \min_{s \in S} s$. Assume that this is positive, or else the problem is trivial. Also, Define the value quanta to be smallest difference between values:

$$QV(S, B) = \min_{v, v' \in B \cup S : v \neq v'} |v' - v|.$$

The probability quanta is the smallest probability:

$$QP(S, B, f_1, f_2) = \min\left(\min_{s \in S} f_1(s), \min_{b \in B} f_2(b)\right).$$

Now, define α^*:

$$\alpha^* = \frac{QP(S, B, f_1, f_2) QV(S, B)}{D(S, B)}$$

Basically, D is the largest value in the system, and $QP(S, B, f_1, f_2)QV(S, B)$ is the smallest. So, if $\alpha < \alpha^*$, even $\alpha D < QP(S, B, f_1, f_2)QV(S, B)$.

First, one can establish a simple bound on r_1 and r_2.

Lemma 3.26 *If agents bid their values for t_1 and t_2, then each seller has an optimal choice for r_1 in $[b_1, b_n]$, and each buyer has an optimal choice for r_2 in $[s_1, s_m]$.*

Proof: Observe that if $r_1(s) < b_1$, then s can increase $r_1(s)$ to b_1, and at least as many trades will occur. Also, s might increase its gain from trade. If $r_1(s) > b_n$, then there will only be a trade only if the coin is heads. If the coin is heads, then $r_1(s)$ can be decreased without loss to the buyer. Thus, $r_1(s) = b_n$ is no worse. By a similar argument $r_2(b) \in [s_1, s_m]$. ∎

The next few lemmas leverage the constraint on α. The first three lemmas characterize to some degree how matches are affected by r_1 and r_2.

Lemma 3.27 *If $0 < \alpha < \alpha^*$ and for some $b \in B$ it is the case that $r_1(b) \leq s_i$, and $r_2(s_{i'}) \in [b_1, b_n]$, then $s_{i'}$ will never match b if $i' > i$.*

Proof: If $r_2(s_{i'}) \leq r_1(b)$, then $z_1 \leq s_i < s_{i'}$. If $r_2(s_{i'}) > r_1(b)$, then:

$$z_1 \leq r_1(b) + \alpha(r_2(s_{i'}) - r_1(b))$$

Since $r_2(s_{i'}) \leq b_n$, and $r_1(b) \geq s_1$, then:

$$z_1 \leq r_1(b) + \alpha D$$

Since $s_{i'} - QV(S, B) \geq s_i$:

$$z_1 \leq s_{i'} - QV(S, B) + \alpha D$$

Because $\alpha < \alpha^*$, $QV(S, B) > \alpha D$. Thus, $z_1 < s_{i'}$, and therefore there will never be a trade. ∎

Lemma 3.28 *If $0 < \alpha < \alpha^*$ and for some $s \in S$ it is the case that $r_2(s) \geq b_j$, and $r_1(b_{j'}) \in [s_1, s_m]$, then $b_{j'}$ will never match s if $j' < j$.*

The proof is similar to that above.

Lemma 3.29 *If $0 < \alpha < \alpha^*$, and for all $b \in B$, $r_1(b) \in S$ and ($r_1(b) \leq b$ or $r_1(b) = s_1$), and r_1 is weakly increasing, then for any $i \in \{1, \ldots, m\}$, for any $j \in \{1, \ldots, n-1\}$, if $r_2(s_i) \in [s_i, b_j]$, then for all $j' > j$, s_i is matched to $b_{j'}$ with probability 1 if and only if $r_1(b_{j'}) \geq s_i$.*

Proof: The proof is similar to that above. If $r_1(b_{j'}) \leq b_{j'}$ and $r_2(s_i) \leq b_{j'}$, then $z_2 \leq b_{j'}$. ∎

The following two lemmas will allow a focus on simple behaviors of the two agents.

Lemma 3.30 *If $0 < \alpha < \alpha^*$, and for all $b \in B$, $r_1(b) \in S$, and ($r_1(b) \leq b$ or $r_1(b) = s_1$), and r_1 is weakly increasing, then there is a best response for the seller where $r_2 \in B$.*

Proof: Given some $s_i \in S$, define $j = \max_{j \in 1 \dots n : b_j \leq r_2(s_i)}$. Define $v = r_2(s_i) - b_j$. A proof below shows that s_i does no worse if $v = 0$. By Lemma 3.28, for all $j' < j$, s_i will not match $b_{j'}$, regardless of the value of v. The case where $n = j$, v can be made zero by Lemma 3.26. Define $U(x)$ to be utility of s_i if it selects $r_2 = x$. There are three different situations, based on $r_1(b_j)$:

1. $r_1(b_j) < s_i$. Then, it is the case that $r_1(b_j) \leq s_{i-1}$. Thus by Lemma 3.27, s_i never matches b_j. By Lemma 3.29, s_i will still match the same agents if $r_2(s_i) = b_{j+1}$.

2. $r_1(b_j) > s_i$. Then, it is the case that $r_1(b_j) \geq s_{i+1}$. Observe that if $v > 0$, then s_i can only possibly match b_j and receive the value $r_1(b_j) - s_1$ if the coin is heads, but if $v = 0$, then s_i and b_j always match. Also, the gain from $v > 0$ is only received by the seller if the buyer's type is greater than b_j and the coin is tails. Thus:

$$U(b_j) - U(v + b_j) \geq (1/2) f_2(b_j)(r_1(b_j) - s_i) - \alpha(1/2)(1 - F_2(b_j))(v)$$

   Since

$$\begin{aligned} v &< D \\ r_1(b_j) - s_i &\geq s_{i+1} - s_i \\ &\geq QV(S, B) \\ f_2(b_j) &\geq QP(S, B, f_1, f_2) \\ 1 - F_2(b_j) &\leq 1 \end{aligned}$$

   It is the case that:

$$U(b_j) - U(v + b_j) \geq (1/2) QP(S, B, f_1, f_2) QV(S, B) - \alpha(1/2) D$$

   Because $\alpha < \alpha^*$:

$$U(b_j) - U(v + b_j) > 0$$

   Thus, the seller can use $v = 0$.

3. $r_1(b_j) = s_i$. There are three cases:

   (a) If $(b_{j+1} - b_j)(1 - F_2(b_j)) > f_2(b_j)(b_j - s_i)$, then $U(b_{j+1}) > U(v + b_j)$.

   (b) If $(b_{j+1} - b_j)(1 - F_2(b_j)) < f_2(b_j)(b_j - s_i)$, then $U(b_j) \leq U(v + b_j)$.

   (c) If $(b_{j+1} - b_j)(1 - F_2(b_j)) = f_2(b_j)(b_j - s_i)$, then $U(b_j) = U(v + b_j) = U(b_{j+1})$.

   Thus, in these cases, the seller can use b_{j+1} or b_j.

In any situation, there is always a value for $r_1 \in B$ that is just as good or better than any arbitrary value. ■

Lemma 3.31 *If $0 < \alpha < \alpha^*$, and for all $s \in S$, $r_2(s) \in B$, and ($r_2(s) \geq s$ or $r_2(s) = b_n$), and r_2 is weakly increasing, then there is a best response for the buyer where $r_1 \in S$.*

The proof is the mirror image of the one above.

Lemma 3.32 *If $0 < \alpha < \alpha^*$, and the buyer uses a strategy $r_1(b) = M_1(b)$, then the seller is motivated to use the strategy $r_2(s) = M_2(s)$.*

Proof: Consider the motivations of a single seller of type $s \in S$. Observe that M_1 is an increasing function and ($M_1(b) \leq b$ or $M_1(b) = s_1$), so that by Lemma 3.30, the optimal strategy for s is to bid $t_1 = s$ and $r_1 \in B$. Consider two cases:

1. There exists a $b \in B$ where $M_1(b) = s$. Then, by the definition of M_2:

$$M_2(s) = \min_{b \in B: M_1(b) = s} b$$

Now, s will not desire to select any $b \in B$ where $b < M_2(s)$, because s and b will never trade. It will never set $r_2(s)$ greater than any b where $M_1(b) > s$, because it would not risk losing $M_1(b) - s$ profit. If $M_1(b_n) = s$, define $b' = b_n$, and otherwise define $b' = \min_{b \in B: M_1(b) > s} b$. Thus, if the seller is of type s, it will play according to:

$$r_2(s) \in \operatorname*{argmax}_{v \in B: M_2(s) \leq v \leq b'} (f_2(v) + (1 - F_2(v)))(v - s) \tag{3.1}$$

Now, by Lemma 3.25:

$$M_2(s) \in \operatorname*{argmax}_{v \in B} (f_2(v) + (1 - F_2(v)))(v - C_1(s)).$$

In order to compare these, we bound the maximum value obtainable in (3.1):

$$\max_{v \in B: M_2(s) \leq v \leq b'} (f_2(v) + (1 - F_2(v)))(v - s) = \max_{v \in B: M_2(s) \leq v \leq b'} (f_2(v) + (1 - F_2(v)))(v - s)$$

Thus, $r_2(s)$ can equal $M_2(s)$.

2. For all $b \in B$, $M_1(b) \neq s$. Then $M_2(s) = \min_{b \in B: M_1(b) > s} b$. s will never set $r_2(s)$ above $M_2(s)$, because it would not want to lose $M_1(M_2(s)) - s$ profit. s will never set $r_2(s)$ below $M_2(s)$, because it can never match with $b < M_2(s)$ regardless of where it sets $r_2(s)$. Thus, it is optimal again for r_2 to select $M_2(s)$.

■

Lemma 3.33 *If $0 < \alpha < \alpha^*$, and the seller uses a strategy $r_2(s) = M_2(s)$, then the buyer is motivated to use the strategy $r_1(b) = M_1(b)$.*

The proof is the mirror image of the one above.

Lemma 3.34 *If $0 < \alpha < \alpha^*$, the seller uses a strategy $r_2(s) = M_2(s)$, and the buyer uses the strategy $r_1(b) = M_1(b)$, then the two agents trade if and only if they would have traded in the optimal mechanism.*

Proof: For any $b \in B$, $s \in S$, if $M_1(b) \geq s$ and $M_2(s) \leq b$, then clearly there will be a match. If not, then by Lemma 3.27 and Lemma 3.28, they will not be matched. ■

Proof (of Theorem 3.11): From Lemma 3.32 and Lemma 3.33, if $0 < \alpha < \alpha^*$, the agents bid according to the optimal auction. Also, from Lemma 3.34, they will trade if and only if they would have traded in the optimal auction. Because an α fraction of the profit of the broker is returned to the buyer or the seller, the broker receives $1 - \alpha$ of the optimal broker's profit. ■

3.5 Proofs of Lower Bounds*

This section contains proofs for lower bounds of the achievable competitive ratios for direct mechanisms and partially direct mechanisms. These lower bounds require the utilization of the incentive compatibility, individual rationality, and the ignorance of the distribution of the mechanism that is being analyzed.

3.5.1 Direct Mechanisms

In [Mye81], the revenue equivalence principle for a seller's auction states that any two ex interim incentive compatible mechanisms for the buyers that have the same allocation rule have the same revenue for the seller (to within an additive constant over all types of buyers). Here, we prove a revenue equivalence principle for an online direct mechanism in the double auction setting if it is incentive compatible over arbitrary distributions of the buyer and the seller. In [MS83], they find an optimal mechanism for maximizing the broker's revenue if the distributions of the buyer and seller are known. Many of the techniques they use are applicable here.

Theorem 3.35 *If an online direct mechanism is incentive compatible, then for all t_1,t_2, $p(t_1,t_2)$ is weakly decreasing in t_1 and weakly increasing in t_2, and:*

$$\begin{aligned} U_1(t_1,t_2) &= U_1(v_{max},t_2) + \int_{t_1}^{v_{max}} p(s_1,t_2)ds_1 \\ U_2(t_1,t_2) &= U_2(t_1,0) + \int_0^{t_2} p(t_1,s_2)ds_2 \end{aligned}$$

Corollary 3.36 *The utility of the broker is:*

$$\begin{aligned} U_0(t_1,t_2) &= (t_2-t_1)p(t_1,t_2) \\ &\quad - \left[U_1(v_{max},t_2) + \int_{t_1}^{v_{max}} p(s_1,t_2)ds_1\right] \\ &\quad - \left[U_2(t_1,0) + \int_0^{t_2} p(t_1,s_2)ds_2\right] \\ &\leq (t_2-t_1)p(t_1,t_2) \\ &\quad - \int_{t_1}^{v_{max}} p(s_1,t_2)ds_1 \\ &\quad - \int_0^{t_2} p(t_1,s_2)ds_2 \end{aligned}$$

Proof: This proof follows quite closely the beginning of Theorem 1 of [MS83]. Basically, the only distributions that really matter are the deterministic distributions. Assume that, at the beginning, there exists a t_1 such that $\Pr[v_1 = t_1] = 1$ and a t_2 such that $\Pr[v_2 = t_2] = 1$. The incentive compatibility constraint can be written in the following form. For any t_1':

$$U_1(t_1,t_2) \geq x_1(t_1',t_2) - t_1 p(t_1',t_2)$$

Similarly:

$$U_1(t_1',t_2) \geq x_1(t_1,t_2) - t_1' p(t_1,t_2)$$

From these two equations, it follows that:

$$(t_1'-t_1)p(t_1,t_2) \geq U_1(t_1') - U_1(t_1') \geq (t_1'-t_1)p(t_1',t_2)$$

If $t_1' > t_1$, then $p(t_1,t_2) \geq p(t_1',t_2)$, so $p(\cdot,t_2)$ is decreasing. Since p_1 is decreasing, it is Riemann integrable, and so the above equation implies that

$$\frac{\partial U_1(t_1,t_2)}{\partial t_1} = -p(t_1,t_2)$$

at almost every t_1. Thus:

$$U_1(t_1,t_2) = U_1(v_{max},t_2) + \int_{t_1}^{v_{max}} p(s_1,t_2)ds_1$$

A similar argument for the buyer shows that:

$$U_2(t_1, t_2) = U_2(t_1, 0) + \int_0^{t_2} p(t_1, s_2) ds_2$$

■

Define $n = \lfloor v_{max} \rfloor$. Consider the n pairs of distributions, $\{(F_1^1, F_2^1), \ldots, (F_1^n, F_2^n)\}$. For (F_1^i, F_2^i), it is the case that the seller is deterministically $i-1$, and the buyer is i. Observe that the optimal broker achieves a value of 1. If p is the algorithm, define

$$p_1 = \min_{i \in \{1 \ldots n\}} p(i-1, i).$$

From Lemma 3.36, observe that the competitive ratio of i is no more than $\frac{1}{p_1}$. We now show a recursive bound that will show an upper bound on p_1.

Lemma 3.37 *Define p_t such that:*

$$p_t = \min_{i \in \{t \ldots n\}} p(i-t, i).$$

If the mechanism is ex-post weakly budget balanced then for all $t > 1$:

$$p_t \geq \frac{t+1}{3} p_1$$

Proof: Define

$$p_t^* = \begin{cases} p_1 & \text{if } t = 1 \\ \frac{t+1}{3} p_1 & \text{otherwise.} \end{cases}$$

This lemma can be proven by recursion. Assume that the theorem holds for all $t' < t$. Now, observe that, for any integer $a \in \{t, n\}$:

$$U_0(a-t, a) \quad \leq \quad (t) p(t_1, t_2) - \int_{a-t}^{v_{max}} p(s_1, a) ds_1 - \int_0^a p(a-t, s_2) ds_2$$

Since p is nonnegative, decreasing in its first argument, and increasing in its second argument, this can be bounded above by:

$$\begin{aligned} U_0(a-t, a) \quad &\leq \quad tp(a-t, t) - \int_{a-t}^{a-1} p(s_1, a) ds_1 - \int_{1+a-t}^{a} p(a-t, s_2) ds_2 \\ &\leq \quad tp(a-t, t) - \sum_{s_1=1+a-t}^{a-1} p(s_1, a) - \sum_{s_2=1+a-t}^{a-1} p(a-t, s_2). \end{aligned}$$

Moreover, by the definition of $p_{t'}$, this can be bounded above by:

$$U_0(a-t, a) \leq tp(a-t, t) - 2 \sum_{t'=1}^{t-1} p_{t'}.$$

Now, observe that if the mechanism is ex-post weakly budget balanced, the expected utility of the broker must be nonnegative regardless of the types of the agents. Thus

$$0 \leq tp(a-t, t) - 2 \sum_{t'=1}^{t-1} p_{t'}$$

$$p(a-t, t) \geq \frac{2}{t} \sum_{t'=1}^{t-1} p_{t'}$$

By the inductive hypothesis

$$p(a-t,t) \geq \frac{2}{t}\sum_{t'=1}^{t-1} p^*_{t'}.$$

Since a is any integer in $\{t,\ldots,n\}$:

$$p_t \geq \frac{2}{t}\sum_{t'=1}^{t-1} p^*_{t'}.$$

A bit of algebra can be used to derive the lemma. ■

Proof (of Theorem 3.3): From the above lemma, $p(0,n) \geq \frac{n+1}{3}p_1$. This implies that $p_1 \leq \frac{3}{n+1}$, and the competitive ratio is no less than $\frac{\lfloor v_{max}\rfloor+1}{3}$. ■

3.5.2 Partially Direct Mechanisms

A proof follows for a mechanism that is truthful for the seller. The case where the mechanism is truthful for the buyer is analogous. Since the goal is a lower bound, make the broker's job easier by giving it more information. Fix the seller's price to be zero. Inform the broker not only that the seller's price is zero, but that it is always zero. Also, inform the broker that the buyer has a single, deterministic value, but not what that value is. Thus, after being given the type of the buyer v_2, the mechanism is aware of F_1, F_2, v_1, and v_2. Therefore, the revelation principle holds, and the focus can now be on direct mechanisms. In fact, a result from [Mye81], rewritten for the simpler case here, will be useful.

Lemma 3.38 *If the mechanism is incentive compatible for the buyer, then:*

1. *p is increasing.*
2. *$tp(t) - x_2(t) = (0p(0) - x_2(0)) + \int_0^t p(s)ds$.*

Using this lemma, we can prove the following result.

Lemma 3.39 *Suppose there is an indirect mechanism M. Given that the mechanism designer is aware of the fact that the seller's valuation is 0 with probability 1, and the fact that there exists a value t_2 such that* $\Pr[v_2 = t_2] = 1$*, but not the value t_2, the mechanism designer can construct a reserve price mechanism that achieves the same expected revenue as M for the broker regardless of F_2 or v_2.*

Proof: The revelation principle holds, because the broker is aware of F_1, F_2, v_1, and v_2 after the it has received the buyer's bid v_2. Therefore, there exists an optimal online mechanism that is a direct mechanism. One can consider this mechanism to consist of three functions: $x_1 : [0, v_{max}] \to \mathbf{R}$, $x_2 : [0, v_{max}] \to \mathbf{R}$ and $p : [0, v_{max}] \to \mathbf{R}$. $x_1(t)$ is the expected payment from the broker to the seller if the buyer reports t. $x_2(t)$ is the expected payment from the buyer to the broker if the buyer reports t. $p(t)$ is the probability that the buyer receives the item.

In order to be individually rational for the seller, $x_1(t^*) - 0p(t^*) \geq 0$. Therefore, $x_1(t^*) \geq 0$. Thus, since x_1 needs to be minimized in order to maximize the broker's profit, $x_1 = 0$ is optimal.

Use Lemma 3.38, rewriting the result as:

$$x_2(t) = tp(t) + x_2(0) - \int_0^t p(s)ds.$$

If x_2 is maximized, the broker's revenue is maximized. Observe that by the individual rationality of the buyer, $0p(0) - x_2(0) \geq 0$, $x_2(0) \leq 0$, so for the maximum revenue, $x_2(0) = 0$, and:

$$x_2(t) = tp(t) + x_2(0) - \int_0^t p(s)ds.$$

Consider a different mechanism: the broker selects a reserve price z for the buyer according to the distribution where $\Pr[z \leq t] = p(t)$. It is valid to consider p as a distribution because it is increasing. This mechanism has a revenue of:

$$R(t) = \int_0^t s \Pr[z = s] ds$$

Using integration by parts where $u(s) = s$ and $v(s) = \int_0^s \Pr[z = r] ds = p(s)$:

$$R(t) = tp(t) - \int_0^t p(s) ds$$

Thus, a holdout price mechanism where $\Pr[z \leq t] = p(t)$ is the optimal mechanism. ∎

Proof (of Theorem 3.6): Observe that the problem is only easier if $v_1 = 0$ with probability 1, and that the mechanism is aware of F_1. By Lemma 3.39, one can assume that the optimal mechanism is a randomly chosen holdout price.

Now, observe that the optimal distribution for selecting a holdout price is identical to the selection of a threshold in Section 2.3. Thus, one cannot do better. ∎

3.6 The Optimal Discrete Mechanism*

This section contains a proof that Mechanism 3.22 is indeed the optimal offline discrete broker. The proof is similar to that found in [MS83]. First we describe two functions V_1 and V_2 which are similar (though not always identical to) the functions C_1 and C_2 described in Section 3.4.3. Section 3.6.1 describes how V_1 and V_2 can be useful in describing the maximum utility obtainable by the broker given a rule for when to trade. Section 3.6.2 compares the V_i to the C_i, in order to prove that the mechanism in Section 3.4.3 is indeed optimal.

3.6.1 The Optimal Profit Given a Trading Rule p

Consider the general formulation of a discrete mechanism introduced in Section 3.4.3.

Lemma 3.40 *If a mechanism (p, x_1, x_2) is incentive-compatible, then $\bar{p}_1$ is decreasing and for all $s_i \in S$:*

$$U_1(s_i) \geq U_1(s_m) + \sum_{\hat{i}=i+1}^{m} (s_{\hat{i}} - s_{\hat{i}-1})\bar{p}_1(s_{\hat{i}}).$$

Proof: Observe that the incentive-compatibility of (p, x_1, x_2) implies that for any $s_i \in S \backslash s_m$:

$$\begin{aligned} U_1(s_i) &= \bar{x}_1(s_i) - s_i\bar{p}_1(s_i) \geq \bar{x}_1(s_{i+1}) - s_i\bar{p}_1(s_{i+1}) \\ U_1(s_{i+1}) &= \bar{x}_1(s_{i+1}) - s_{i+1}\bar{p}_1(s_{i+1}) \geq \bar{x}_1(s_i) - s_{i+1}\bar{p}_1(s_i) \end{aligned}$$

Arithmetic yields:

$$s_{i+1}(\bar{p}_1(s_i) - \bar{p}_1(s_{i+1})) \geq \bar{x}_1(s_i) - \bar{x}_2(s_{i+1}) \geq s_i(\bar{p}_1(s_i) - \bar{p}_1(s_{i+1})).$$

Thus, since $s_{i+1} > s_i$, $\bar{p}_1$ is decreasing. Arithmetic also yields:

$$U_1(s_i) \geq U_1(s_{i+1}) + (s_{i+1} - s_i)\bar{p}_1(s_{i+1})$$

The lemma follows by induction. ∎

Lemma 3.41 *If a mechanism (p, x_1, x_2) is incentive-compatible, then $\bar{p}_2$ is increasing and for all $b_i \in B$:*

$$U_2(b_i) \geq U_2(b_1) + \sum_{\hat{i}=1}^{i-1} (b_{\hat{i}+1} - b_{\hat{i}}) \bar{p}_2(b_{\hat{i}}).$$

The proof is similar to that of the previous lemma.

For all $i \in \{1, \ldots, m\}$, define:

$$V_1(s_i) = \begin{cases} s_i + \frac{F_1(s_{i-1})(s_i - s_{i-1})}{f_1(s_i)} & \text{if } i > 1 \\ s_1 & \text{if } i = 1 \end{cases}$$

For all $j \in \{1, \ldots, n\}$, define:

$$V_2(b_j) = \begin{cases} b_j - \frac{(1 - F_2(b_j))(b_{j+1} - b_j)}{f_2(b_j)} & \text{if } j < n \\ b_n & \text{if } j = n \end{cases}$$

These functions are closely related to C_1 and C_2. If the distribution F_2 is **regular**, $V_2 = C_2$. Particularly, $V_2(b_i)$ is the point where a seller is indifferent between a reserve price of b_i and a reserve price of b_{i+1}. We will return to this relationship later.

While C_1 and C_2 are useful functions for determining the optimal auction, V_1 and V_2 are useful in their own right in describing the optimal utility obtainable by a broker given a function $p(s_i, b_j)$.

Theorem 3.42 *Given (x_1, x_2, p) that is incentive-compatible and individually rational, the expected utility of the broker is less than:*

$$U_0 \leq \sum_{i=1}^{m} \sum_{j=1}^{n} f_1(s_i) f_2(b_j) \left(V_2(b_j) - V_1(s_i) \right) p(s_i, b_j).$$

Moreover, if $\bar{p}_1$ is decreasing and $\bar{p}_2$ is increasing, then there exists an x_1' and x_2' such that (x_1', x_2', p) is incentive-compatible, individually rational, and achieves the upper bound.

Proof: Observe that, for all $s_i \in S$, $b_j \in B$

$$U_0(s_i, b_j) = p(s_i, b_j)(b_j - s_i) - U_1(s_i, b_j) + U_2(s_i, b_j).$$

Taking the expected value of both sides yields

$$U_0 = \sum_{i=1}^{m} \sum_{j=1}^{n} f_1(s_i) f_2(b_j) p(s_i, b_j)(b_j - s_i) - \sum_{i=1}^{m} f_1(s_i) U_1(s_i) - \sum_{j=1}^{n} f_2(b_j) U_2(b_j).$$

By Lemma 3.40 and Lemma 3.41:

$$\begin{aligned}
U_0 &\le \sum_{i=1}^{m}\sum_{j=1}^{n} f_1(s_i)f_2(b_j)p(s_i,b_j)(b_j-s_i)\\
&\quad -U_1(s_m)-\sum_{i=1}^{m-1} f_1(s_i)\sum_{\hat{i}=i+1}^{m}(s_{\hat{i}}-s_{\hat{i}-1})p_1(s_{\hat{i}})\\
&\quad -U_2(b_1)-\sum_{j=2}^{n} f_2(b_j)\sum_{\hat{j}=1}^{j-1}(b_{\hat{j}+1}-b_{\hat{j}})p_2(b_{\hat{j}})\\
U_0 &\le \sum_{i=1}^{m}\sum_{j=1}^{n} f_1(s_i)f_2(b_j)p(s_i,b_j)(b_j-s_i)\\
&\quad -U_1(s_m)-\sum_{i=2}^{m} F_1(s_{i-1})(s_i-s_{i-1})p_1(s_i)\\
&\quad -U_2(b_1)-\sum_{j=1}^{n-1}(1-F_2(b_j))(b_{j+1}-b_j)p_2(b_j)
\end{aligned}$$

By individual rationality, assume that $U_1(s_m)\ge 0$ and $U_2(b_1)\ge 0$.

$$\begin{aligned}
U_0 &\le \sum_{i=1}^{m}\sum_{j=1}^{n} f_1(s_i)f_2(b_j)p(s_i,b_j)(b_j-s_i)\\
&\quad -\sum_{i=2}^{m} F_1(s_{i-1})(s_i-s_{i-1})\sum_{j=1}^{m} f_2(b_j)p(s_i,b_j)\\
&\quad -\sum_{j=1}^{n-1}(1-F_2(b_j))(b_{j+1}-b_j)\sum_{i=1}^{n} f_1(s_i)p(s_i,b_j)\\
U_0 &\le \sum_{i=1}^{m}\sum_{j=1}^{n} f_1(s_i)f_2(b_j)p(s_i,b_j)(b_j-s_i)\\
&\quad -\sum_{i=2}^{m} \frac{F_1(s_{i-1})}{f_1(s_i)}(s_i-s_{i-1})\sum_{j=1}^{m} f_1(s_i)f_2(b_j)p_1(s_i,b_j)\\
&\quad -\sum_{j=1}^{n-1}\frac{1-F_2(b_j)}{f_2(b_j)}(b_{j+1}-b_j)\sum_{i=1}^{n} f_1(s_i)f_2(b_j)p(s_i,b_j)
\end{aligned}$$

Using the definition of V_1 and V_2, one can obtain the upper bound. If one chooses $x_1(s_m)$ and $x_2(b_1)$ such that $U_1(s_m)=U_2(b_1)=0$, and chooses x_1 and x_2 such that the bounds in Lemma 3.40 and Lemma 3.41 are tight, then the result here is tight as well. ∎

As the above lemma shows how to measure the optimal profit achievable with a given p, the lemma below shows how for some p (where the trade is deterministic when the buyer and seller are known) to choose a price function that achieves this profit for the broker.

Lemma 3.43 *Given a mechanism (x_1,x_2,p), such that p is decreasing in its first argument and increasing* *in its second argument, for all* $s_i \in S$, $b_j \in B$, $p(s_i, b_j) \in \{0, 1\}$, *and:*

$$\begin{aligned} x_1(s_i, b_j) &= \begin{cases} \max_{s_i : p(s_i, b_j)=1} s_i & \textit{if } p(s_i, b_j) = 1 \\ 0 & \textit{if } p(s_i, b_j) = 0 \end{cases} \\ x_2(s_i, b_j) &= \begin{cases} \min_{s_i : p(s_i, b_j)=1} b_j & \textit{if } p(s_i, b_j) = 1 \\ 0 & \textit{if } p(s_i, b_j) = 0, \end{cases} \end{aligned}$$

Then:

$$U_0 = \sum_{i=1}^{m} \sum_{j=1}^{n} f_1(s_i) f_2(b_j) \left(V_2(b_j) - V_1(s_i)\right) p(s_i, b_j).$$

Proof: Observe that if for any $b_j \in B$, $p(s_m, b_j)$ trades, then $x_1(s_m, b_j) = 0$. Thus, $U_1(s_m) = 0$. By induction it can be proven that the above mechanism is tight according to Lemma 3.40. First, we can prove that for all $s_i < s_m$, for all $b_j \in B$, $U_1(s_i, b_j) = U_1(s_{i+1}, b_j) + (s_{i+1} - s_i) p(s_{i+1}, b_j)$. There are three cases:

1. $p(s_i, b_j) = p(s_{i+1}, b_j) = 0$. Then $x_1(s_i, b_j) = x_1(s_{i+1}, b_j) = 0$, and $U_1(s_i, b_j) = U_1(s_{i+1}, b_j) = 0$, so $U_1(s_i, b_j) = U_1(s_{i+1}, b_j) + p(s_{i+1}, b_j)(s_{i+1} - s_i)$.
2. $p(s_i, b_j) = p(s_{i+1}, b_j) = 1$. Then $x_1(s_i, b_j) = x_1(s_{i+1}, b_j)$. Thus,

$$\begin{aligned} U_1(s_i, b_j) &= x_1(s_{i+1}, b_j) - s_i p(s_{i+1}, b_j) \\ U_1(s_i, b_j) &= x_1(s_{i+1}, b_j) - s_{i+1} p(s_{i+1}, b_j) + (s_{i+1} - s_i) p(s_{i+1}, b_j) \\ U_1(s_i, b_j) &= U_1(s_{i+1}, b_j) + (s_{i+1} - s_i) p(s_{i+1}, b_j). \end{aligned}$$

3. $p(s_i, b_j) = 1$, and $p(s_{i+1}, b_j) = 0$. Then $x_1(s_{i+1}, b_j) = 0$, and $x_1(s_i, b_j) = s_i$. Thus:

$$\begin{aligned} U_1(s_i, b_j) &= x_1(s_i, b_j) - s_i p(s_i, b_j) \\ U_1(s_i, b_j) &= s_i - s_i 1 \\ U_1(s_i, b_j) &= 0 \\ U_1(s_i, b_j) &= U_1(s_{i+1}, b_j) + (s_{i+1} - s_i) p(s_{i+1}, b_j) \end{aligned}$$

Now, for all $s_i < s_m$:

$$\begin{aligned} U_1(s_i) &= \sum_{j=1}^{n} U_1(s_i, b_j) f_2(b_j) \\ &= \sum_{j=1}^{n} U_1(s_{i+1}, b_j) f_2(b_j) + (s_{i+1} - s_i) p(s_{i+1}, b_j) f_2(b_j) \\ &= U_1(s_{i+1}, b_j) + (s_{i+1} - s_i) p_1(s_{i+1}) \end{aligned}$$

By induction, this function U_1 is tight. Similarly, one can prove that U_2 is tight. ∎

Thus, it is clear that if $V_1 = C_1$ and $V_2 = C_2$, then this mechanism is optimal, because then $p(s_i, b_j) = 1$ when $V_2(b_j) - V_1(s_i) \geq 0$, and $p(s_i, b_j) = 0$ when $V_2(b_j) - V_1(s_i) < 0$. However, this is not always the case.

3.6.2 Comparing V_i and C_i

This section is similar to the proof in [Mye81], except we develop it here for the discrete case. If a function F_2 is regular, then V_2 is increasing: that means for each buyer's valuation b_j is there is a valuation $s \in \mathbf{R}$ for an imaginary seller which can choose b_j as an optimal reserve price,like in Figure 3.1. If this is the case, then $C_2 = V_2$. However, not all distributions are regular.

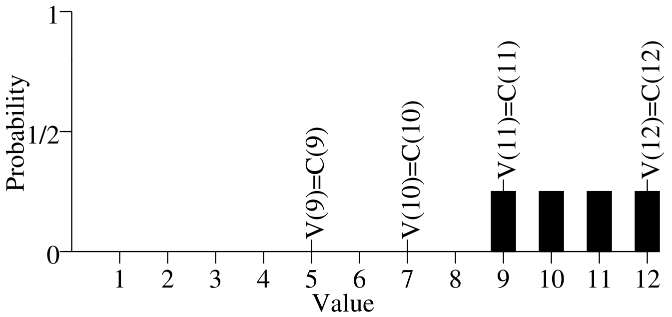

Figure 3.1: Here, there is a uniform distribution and V_2 is increasing. Consider an imaginary seller with valuation s that wants to maximize its expected profit. If $s < 5$, it should use a holdout price of 9. If $s \in (V(i), V(i+1))$, it should use a holdout price of $i+1$. If $s = V(i)$, it can use a holdout price of i or $i+1$.

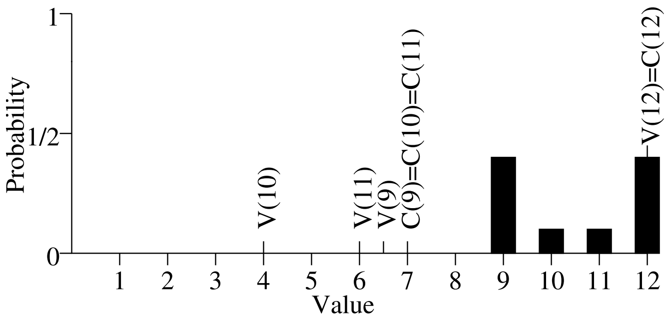

Figure 3.2: Observe that there is a high probability on the lowest bid. In this case, there is no imaginary seller who would choose 10 as a holdout price. If $s < 7$, the seller should use 9 as a holdout price. If $s > 7$, the seller should use 12. If $s = 7$, the seller could use either 9, 11, or 12 as a holdout price (but not 10).

Define $\tilde{F}_2(b_j) = 1 + f_2(b_j) - F_2(b_j)$. Observe that for any two values $b_j, b_{j'} \in B$ where $b_j \neq b_{j'}$, there exists a unique valuation $s_{j,j'}$ for an imaginary seller such that:

$$\tilde{F}_2(b_j)(b_j - s_{j,j'}) = \tilde{F}_2(b_{j'})(b_{j'} - s_{j,j'}).$$

Thus, for a seller with this valuation b_j and $b_{j'}$ are equally useful as holdout prices. Solving for $s_{j,j'}$, we obtain:

$$s_{j,j'} = \frac{\tilde{F}_2(b_j)b_j - \tilde{F}_2(b_{j'})b_{j'}}{\tilde{F}_2(b_j) - \tilde{F}_2(b_{j'})}.$$

These valuations are very useful in determining in whether a valuation b_j of a buyer would be ever useful as a holdout price. Observe that b_1 is useful for a seller with a very low valuation, and b_n is useful if the valuation is higher than b_{n-1}. Particularly, if and only if there exists j' and j'' such that $j' < j < j''$ and $s_{j,j'} > s_{j,j''}$, then b_j is never **useful**, i.e. useful as a holdout price. This is because below $s_{j,j'}$, $b_{j'}$ is a better holdout price than b_j. Above $s_{j,j''}$, $b_{j''}$ is a better holdout price than $b_{j''}$.

What is the maximum seller that will trade with b_j when $j < n$? First, define $b_{j'}$ to be the highest useful bid that is less than or equal to b_j. Define $b_{j''}$ to be the lowest useful bid that is strictly greater than b_j. Observe that by definition, there are no bids between $b_{j'}$ and $b_{j''}$ that are useful, therefore at $s_{j',j''}$, $b_{j'}$ and $b_{j''}$ are both optimal. Thus, $C_2(b_j) = s_{j',j''}$. If b_j and b_{j+1} are useful, then $C_2(b_j) = s_{j,j+1} = V_2(b_j)$.

Now consider the function $H : B \to \mathbf{R}$ defined as:

$$H_2(b_j) = \sum_{j'=1}^{j-1} V_2(b_{j'})f_2(b_{j'}).$$

Using the definition of V_2, the above can be rewritten as:

$$H_2(b_j) = \sum_{j'=1}^{j-1} \tilde{F}_2(b_{j'})b_{j'} - \tilde{F}_2(b_{j'+1})b_{j'+1}.$$

Thus, telescoping yields:

$$H_2(b_j) = \tilde{F}_2(b_1)b_1 - \tilde{F}_2(b_j)b_j.$$

Similarly, we can define:

$$G_2(b_j) = \sum_{j'=1}^{j-1} C_2(b_{j'})f_2(b_{j'}).$$

It is interesting to observe that G_2 is in a sense the convex hull of H_2.

Lemma 3.44 *For all useful values b_j, $G_2(b_j) = H_2(b_j)$.*

Proof: This can be proved by induction. First, observe that $H_2(b_1) = G_2(b_1) = 0$. Now, assume that for some useful $b_{j'}$, b_j was the highest useful value strict less than b_j. By the inductive hypothesis, $H_2(b_j) = G_2(b_j)$. Also, as described above,

$$H_2(b_{j'}) - H_2(b_j) = \tilde{F}_2(b_j)b_j - \tilde{F}_2(b_{j'})b_{j'}.$$

Since for all j'' where $j < j'' < j'$, j'' is not useful, then $C_2(j'') = C_2(j)$:

$$G_2(b_{j'}) - G_2(b_j) = \sum_{j'''=j}^{j'-1} C_2(b_j)f_2(b_{j'''}) = (\tilde{F}_2(b_j) - \tilde{F}_2(b_{j'}))C_2(b_j).$$

Thus, since $C_2(b_j) = s_{j,j'}$:

$$G_2(b_{j'}) - G_2(b_j) = (\tilde{F}_2(b_j) - \tilde{F}_2(b_{j'}))\frac{\tilde{F}_2(b_j)b_j - \tilde{F}_2(b_{j'})b_{j'}}{\tilde{F}_2(b_j) - \tilde{F}_2(b_{j'})} = H_2(b_{j'}) - H_2(b_j).$$

Thus, the difference between H_2 on two useful points equals the difference between G_2 on two useful points, completing the proof. ■

Lemma 3.45 *For all $b_j \in B$, $G_2(b_j) \leq H_2(b_j)$.*

Proof: Since the previous lemma establishes this for useful b_j, let us assume that b_j is not useful. Suppose $b_{j'}$ is the highest useful bid less than or equal to b_j. Then:

$$\begin{aligned}
G_2(b_j) &= G_2(b_{j'}) + \sum_{j''=j'}^{j-1} C_2(b_{j''})f_2(b_{j''}) \\
&= G_2(b_{j'}) + \sum_{j''=j'}^{j-1} C_2(b_j)f_2(b_{j''}) \\
&= G_2(b_{j'}) + C_2(b_j)(\tilde{F}_2(b_{j'}) - \tilde{F}_2(b_j)) \\
H_2(b_j) &= H_2(b_{j'}) + b_{j'}\tilde{F}_2(b_{j'}) - b_j\tilde{F}_2(b_j) \\
&= G_2(b_{j'}) + b_{j'}\tilde{F}_2(b_{j'}) - b_j\tilde{F}_2(b_j)
\end{aligned}$$

Suppose $b_{j''}$ is the lowest useful bid strictly greater than b_j. Then $C_2(b_j) = s_{j',j''}$. Because b_j is not useful, $s_{j,j'} > s_{j',j''}$, i.e. by the time the seller's valuation is high enough such that b_j is better than $b_{j'}$, $b_{j''}$ is better than both of them.

$$\begin{aligned}
G_2(b_j) &< G_2(b_{j'}) + s_{j,j'}(\tilde{F}_2(b_{j'}) - \tilde{F}_2(b_j)) \\
G_2(b_j) &< G_2(b_{j'}) + \frac{\tilde{F}_2(b_j)b_j - \tilde{F}_2(b_{j'})b_{j'}}{\tilde{F}_2(b_j) - \tilde{F}_2(b_{j'})}(\tilde{F}_2(b_{j'}) - \tilde{F}_2(b_j)) \\
G_2(b_j) &< G_2(b_{j'}) + \tilde{F}_2(b_{j'})b_{j'} - \tilde{F}_2(b_j)b_j \\
G_2(b_j) &< H_2(b_j)
\end{aligned}$$

■

Finally, for an increasing function $p_2(b_j)$, define $p'_2(b_j) = p_2(b_j) - p_2(b_{j-1})$ if $j > 1$, and $p'_2(b_1) = p_2(b_1)$. Similarly, all of these results hold for a seller as well. Specifically, define:

$$\begin{aligned}
b_{i,i'} &= \frac{F_1(s_i)s_i - F_1(s_{i'})s_{i'}}{F_1(s_i) - F_1(s_{i'})} \\
H_1(s_i) &= \sum_{i'=i+1}^{n} V_1(s_i)f_1(s_i) = F_1(s_m)s_m - F_1(s_i)s_i \\
G_1(s_i) &= \sum_{i'=i+1}^{n} C_1(s_i)f_1(s_i) \\
p'_1(s_i) &= \begin{cases} p_1(s_i) - p_1(s_{i+1}) & \text{if } i < m \\ p_1(s_m) & \text{if } i = m \end{cases}
\end{aligned}$$

A useful seller's value s_i is an optimal holdout price in a reverse auction for some buyer. As for buyers, one can show that:

Lemma 3.46 *For all useful values s_i, $G_1(s_i) = H_1(s_i)$. For all $s_i \in S$, $G_1(s_i) \geq H_1(s_i)$.*

The proof is similar to those above.

There is a simple observation about sums to be made here, similar to integration by parts.

Fact 3.47 *Given u, w, U, and W are real-valued functions on the integers between 1 and n, $U(j) = \sum_{j'=1}^{j-1} u(j')$, and $W(j) = \sum_{j'=1}^{j} w(j')$, then $\sum_{j=1}^{n} u(j)W(j) = U(n)W(n) - \sum_{j=1}^{n} U(j)w(j)$.*

Proof (of Theorem 3.24): First, we know from Theorem 3.42 that given a $p(s_i, b_j)$ that is decreasing in its first argument and increasing in its second:

$$U_0 \leq \sum_{i=1}^{m}\sum_{j=1}^{n} f_1(s_i)f_2(b_j)\left(V_2(b_j) - V_1(s_i)\right)p(s_i, b_j). \tag{3.2}$$

For the mechanism described in Section 3.11 $p = p^*$, where:

$$p^*(s_i, b_j) = \begin{cases} 1 & \text{when } V_2(b_j) \geq V_1(s_i) \\ 0 & \text{when } V_2(b_j) < V_1(s_i). \end{cases} \tag{3.3}$$

Observe that $p(s_i, b_j)$ might not be decreasing in its first argument if V_1 was not increasing, and $p(s_i, b_j)$ might not be increasing in its second argument if V_2 was not decreasing. Thus, we rewrite Equation (3.2) in order to consider the effects of using C_1 and C_2:

$$\begin{aligned} U_0 &\leq \sum_{i=1}^{m}\sum_{j=1}^{n} f_1(s_i)f_2(b_j)\left(C_2(b_j)-C_1(s_i)\right)p(s_i,b_j) \\ &\quad +\sum_{j=1}^{n}\left(V_2(b_j)f_2(b_j)-C_2(b_j)f_2(b_j)\right)p_2(b_j) \\ &\quad -\sum_{i=1}^{m}\left(V_1(s_i)f_1(s_i)-C_1(s_i)f_1(s_i)\right)p_1(s_i) \end{aligned}$$

If we use Fact 3.47, where $U = H_2 - G_2$ and $V = p_2$, then:

$$\sum_{j=1}^{n}(V_2(b_j)f_2(b_j)-C_2(b_j)f_2(b_j))p_2(b_j) = (H_2(b_n)-G_2(b_n))p_2(b_n)-\sum_{j=1}^{n}\left(H_2(b_j)-G_2(b_j)\right)p_2'(b_j).$$

Similarly:

$$\sum_{i=1}^{n}(V_1(s_i)f_1(s_i)-C_1(s_i)f_2(s_i))p_1(s_i) = (H_1(s_1)-G_1(s_1))p_1(s_1)-\sum_{i=1}^{n}\left(H_1(s_i)-G_1(s_i)\right)p_1'(s_i).$$

Observe that since b_n and s_1 are useful:

$$\begin{aligned} \sum_{j=1}^{n}(V_2(b_j)f_2(b_j)-C_2(b_j)f_2(b_j))p_2(b_j) &= \sum_{j=1}^{n}\left(G_2(b_j)-H_2(b_j)\right)p_2'(b_j). \\ \sum_{i=1}^{n}(V_1(s_i)f_1(s_i)-C_1(s_i)f_2(s_i))p_1(s_i) &= \sum_{i=1}^{n}\left(G_1(s_i)-H_1(s_i)\right)p_1'(b_i). \end{aligned}$$

Observe that if p_2 is increasing and p_1 is decreasing, then p_2' and p_1' are both non-negative. Since $G_1(s_i) \geq H_1(s_i)$, and $H_2(b_j) \geq G_2(b_j)$, then:

$$\begin{aligned} \sum_{j=1}^{n}(V_2(b_j)f_2(b_j)-C_2(b_j)f_2(b_j))p_2(b_j) &\leq 0 \\ \sum_{i=1}^{n}(V_1(s_i)f_1(s_i)-C_1(s_i)f_2(s_i))p_1(s_i) &\geq 0. \end{aligned}$$

This implies that:

$$U_0 \leq \sum_{i=1}^{m}\sum_{j=1}^{n} f_1(s_i)f_2(b_j)\left(C_2(b_j)-C_1(s_i)\right)p(s_i,b_j).$$

By Equation (3.3), and the fact that $p(s_i, b_j) \in [0,1]$:

$$U_0 \leq \sum_{i=1}^{m}\sum_{j=1}^{n} f_1(s_i)f_2(b_j)\left(C_2(b_j)-C_1(s_i)\right)p^*(s_i,b_j).$$

It remains to be proven that Mechanism 3.22 achieves this utility. Observe that if $G_2(b_j) \neq H_2(b_j)$, then b_j is not useful, and therefore $j > 1$ and $C_2(b_j) = C_2(b_{j-1})$, thus $p_2^*(b_j) = p_2^*(b_{j-1})$ and $p_2^*(b_j) - p_2^*(b_{j-1})$. Therefore:

$$\sum_{j=1}^{n}(V_2(b_j)f_2(b_j)-C_2(b_j)f_2(b_j))p_2^*(b_j) = 0.$$

Similarly,

$$\sum_{i=1}^{n}(V_1(s_i)f_1(s_i) - C_1(s_i)f_2(s_i))p_1^*(s_i) = 0.$$

Thus, if $p = p^*$:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} f_1(s_i)f_2(b_j)\left(V_2(b_j) - V_1(s_i)\right)p^*(s_i,b_j) = \sum_{i=1}^{m}\sum_{j=1}^{n} f_1(s_i)f_2(b_j)\left(C_2(b_j) - C_1(s_i)\right)p^*(s_i,b_j).$$

Thus, since p_1^* is decreasing and p_2^* is increasing, there is an x_1 and x_2 that make it an optimal trading rule. By Lemma 3.43, the optimal rule is that described in Mechanism 3.22. ■

3.7 Future Work

Clearly, work could be done to make these upper and lower bounds on the competitive ratios closer to each other for the direct and partially direct mechanisms. However, the most interesting areas of future research are:

1. extensions of these mechanisms to the situation where sellers and buyers are bidding over time, and
2. discovering if the α-profit sharing mechanism is $\frac{1}{1-\alpha}$ competitive for all pairs of distributions F_1, F_2 and all $\alpha \in (0,1)$.

The extensions would be simple. For instance, a direct mechanism posts z_1 and z_2 at the beginning, and then greedily matches buyers above z_2 with sellers below z_1. A partially direct mechanism can charge a posting fee for bids and offers, but can allow agents to accept bids and offers for free. Finally, the broker can allow agents to give their own best estimate at the best reserve price for them to be matched with. The broker would want allow this reserve price to change over time.

The first may require some research to be completed in economics first. What type of equilibria exist in these type of mechanisms? The latter is perhaps the more tractable problem, but it is still difficult. The first step is to prove it for discrete distributions where the buyer and seller only have three or four values each. This can most likely be solved by a computer, and would give more evidence to its general applicability.

3.8 Conclusions

If the broker is not knowledgeable about a market, then it must rely on the knowledge of the buyer and seller. If the mechanism simply requests the type of an agent, then the agent is incapable of communicating a wealth of useful information. This is formalized in the following three results.

1. The optimal competitive ratio for a direct mechanism is $\Theta(v_{max})$.
2. The optimal competitive ratio for a partially direct mechanism is $\Theta(\ln v_{max})$.
3. The optimal competitive ratio for an indirect mechanism is $\Theta(1)$.

However, in these later mechanisms, the broker is trusting that the agents understand the market. If they do not, then catastrophic outcomes can result. These algorithms are simple enough so that they can be extended to scenarios where buyers and sellers submit their offers and bids over time. An interesting area of future work is to consider the incentive-compatibility constraints of the online mechanism.

Chapter 4

Preference Elicitation Algorithms[1]

Preference elicitation — the process of asking queries to determine parties' preferences — is a key part of many problems in electronic commerce. For example, a shopping agent needs to know a user's preferences in order to correctly act on her behalf, and preference elicitation can help an auctioneer in a combinatorial auction determine how to best allocate a given set of items to a given set of bidders. Unfortunately, in the worst case, preference elicitation can require an exponential number of queries even to determine an approximately optimal allocation. In this chapter, we study natural special cases of preferences for which elicitation can be done in polynomial time via value queries. The cases we consider all have the property that the preferences (or approximations to them) can be described in a polynomial number of bits, but the issue here is whether they can be elicited using the natural (limited) language of value queries. We make a connection to computational learning theory where the similar problem of *exact learning with membership queries* has a long history. In particular, we consider preferences that can be written as *read-once formulas* over a set of gates motivated by a shopping application, as well as a class of preferences we call *Toolbox DNF*, motivated by a type of combinatorial auction. We show that in each case, preference elicitation can be done in polynomial time. We also consider the computational problem of allocating items given the parties' preferences, and show that in certain cases it can be done in polynomial time and in other cases it is NP-complete. Given two bidders with Toolbox-DNF preferences, we show that allocation can be solved via network flow. If parties have read-once formula preferences, then allocation is NP-hard even with just two bidders, but if one of the two parties is additive (e.g., a shopping agent purchasing items individually and then bundling them to give to the user), the allocation problem is solvable in polynomial time.

4.1 Introduction

We now switch from the study of auctions involving a single type of item to **combinatorial auctions** in which there are N items and a bidder's value for a subset may be quite different (more or less) than the sum of the bidder's values of the individual items in the subset themselves. The focus of this chapter will be on **preference elicitation**, the problem of how an auctioneer can efficiently elicit the information it needs about the bidders' preferences in order to make the optimal allocation of the items to the bidders. In most auctions where multiple (say, n) distinct items are being sold, a bidder's valuation for the items is not additive. Rather, the bidder's preferences generally exhibit **complementarity** (a bundle of items is worth more than the sum of its parts—e.g., a flight to Hawaii and a hotel room in Hawaii) and **substitutability** (a bundle is worth less than the sum of its parts—e.g., a flight to Hawaii and a flight to the Bahamas on the same day). Combinatorial auctions where agents can submit bids on **bundles** (sets of items), are economically efficient auction mechanisms for this setting. The computational problem of determining the winners, given the bids, is a hard optimization problem that has recently received significant research attention.

An equally important problem, which has received less attention, is that of obtaining enough preference

[1]This work appeared previously in [ZBS03].

information from the bidders so that there is a basis for allocating the items (usually the objective is to maximize the sum of the bidders' valuation functions). There are $2^n - 1$ bundles, and each agent may need to bid on all of them to fully express its preferences. This can be undesirable because there are exponentially many bundles to evaluate, and furthermore determining one's valuation for any given bundle can be computationally or cognitively expensive. So in practice, when the number of items for sale is even moderate, the bidders will not bid on all bundles. Instead, they may wastefully bid on bundles that they will not win, and they may suffer reduced economic efficiency by failing to bid on bundles they would have won.

Recently, an approach has been proposed where the auctioneer is enhanced by **elicitor** software that incrementally elicits the bidders' preferences by querying them (based on the preference information elicited so far) [CS01, CS02b, CS02a]. In the worst case, an exponential amount of communication is required to allocate the items even approximately optimally, if the bidders can have general preferences [NS02]. (This holds even when bidders can dispose of extra items for free, that is, their valuation functions are monotone.) However, experimentally, preference elicitation appears to help quite a bit [HS02, HS04]. Nonetheless, the amount of querying can be prohibitively large when the bidders have general (monotone) preferences.[2]

An analogous issue arises with shopping agents. Consider the following scenario. Alice goes to her software agent and asks it to help her purchase a vacation. In order to act on her behalf, the agent first needs to find out Alice's preferences (how much is a trip to Hawaii worth compared to a trip to the Bahamas, does it substantially increase the value to her if she can get some entertainment booked in advance, etc.). Then, after scouring the Internet, the agent needs to solve the computational problem of deciding on the best vacation package—the one that maximizes Alice's valuation minus the cost of the trip. In this scenario, there is no auctioneer. Rather, the elicitor is the buyer's helper. Again, the amount of querying can be prohibitively large when the buyer has general (monotone) preferences.

In this chapter we study natural classes of preferences that we show can be elicited in polynomial time with value queries, and yet are rich enough to express complementarity and substitutability. We consider a setting in which there is a universe of n items, and a user has different valuations over different subsets of these items. The auctioneer (or software agent) can get information about these preferences using **value queries**: proposing a bundle and asking the user for her valuation of that bundle. This is the same as the notion of a **membership query** in computational learning theory.

The restrictions that we place on the preferences will imply that the preferences *can* be compactly represented. This means that preference elicitation would be *easy* if the elicitor was allowed to ask "give me a computer program for (an approximation to) your preferences". But, in practice, a human will not be able to express her preferences by uttering a formula that completely captures them. Value queries are a standard and much more natural communication protocol.

We begin by arguing that *read-once formulas* over a certain set of gates (defined below) can express many natural preference functions. We then show that if the user's preferences *can* be expressed as a read-once formula of this form, then preference elicitation can be done in polynomial time using value queries (this builds upon work of Angluin et al. [AHK93] and Bshouty et al. [BHHK94, BHH95] for learning over classes of gates motivated from learning theory).[3]

More precisely, we assume the bidder's preferences can be described as a read-once formula over $\{0,1\}^n$ (an input $x \in \{0,1\}^n$ corresponds to the bundle containing the items i such that $x_i = 1$) using gates $\{\text{SUM}, \text{MAX}, \text{ALL}\}$, with real-valued multipliers on the inputs. A read-once formula is a function that can be represented as a tree ("read-once" means that even the initial inputs may only have out-degree 1). A SUM node sums the values of its inputs; a MAX node takes the maximum value of its inputs; an ALL node sums its inputs *unless* one of the inputs is zero, in which case the output is 0. For convenience, we will sometimes view inputs as $x \in \{0,1\}^n$ and sometimes as $x \subseteq \{1, \ldots, n\}$. Each input i also has a positive

[2] Ascending combinatorial auctions [BdVSV01, PU00, WW00] can be viewed as a special case of the preference elicitation framework where the queries are of the form: "Given these prices on items (and possibly also on bundles), which bundle would you prefer the most?".

[3] The reason for so much work on read-once formulas is that Angluin [Ang88] shows that reconstructing read-twice functions from membership (value) queries alone can require an exponential number of queries, even if the function is simply an OR of ANDs (a monotone DNF).

real-valued multiplier v_i (representing the intrinsic value of that item). For example, a legal function on 3 inputs might be $\mathrm{ALL}(2x_1, \mathrm{MAX}(10x_2, 4x_3))$, which gives value 12 to the input 110, 6 to the input 101, and 0 to the input 011.

Read-once formulas of this type allow for many natural preferences. For example, suppose items are flights and hotel rooms in different locations (e.g., input $x_{i,j,0}$ represents the ith flight to location j, and $x_{i,j,1}$ represents the ith hotel room in location j) and a couple want to take just one trip. Then for each location j the couple could compute $\mathrm{ALL}(\mathrm{MAX}\{v_{i,j,0}x_{i,j,0}\}_i,\ \mathrm{MAX}\{v_{i,j,1}x_{i,j,1}\}_i)$, and then at the root of the tree the couple would take a MAX over the different destinations.[4]

We then generalize our results by considering a broader class of gates. Let MAX_k output the sum of the k highest inputs, and $\mathrm{ATLEAST}_k$ output the sum of its inputs if there are at least k positive inputs, and 0 otherwise. Finally, we consider $\mathrm{GENERAL}_{k,l}$, a parameterized gate capable of representing all the above types of gates. We show that read-once preferences including all of these gates can be elicited in a polynomial number of queries. We also give positive results for the setting where preferences are *approximately* read-once formulas with MAX and SUM gates.

In addition to the elicitation problem, we study the computational allocation problem of determining how the items should be divided among agents with read-once preferences. We show that once the preference function f is known, the problem of finding the subset of items x that maximizes $f(x) - g(x)$, where g is a *linear* cost function (or equivalently, maximizing $f(x) + g(\{1, \ldots, n\} - x)$, where g is a linear valuation function), can be done in polynomial time. This is natural for the case of a shopping agent that buys items individually on the web for a user with valuation function f. However, if g is a general read-once formula, then we show this optimization is NP-complete.

Finally, we consider the class of preferences that can be expressed as monotone polynomials. E.g., $f(x) = ax_1x_2 + bx_2x_3x_4 + cx_3x_4$. We call this class *Toolbox DNF* because it captures settings where each agent has a set of tasks to accomplish (one per term in the polynomial), each task requiring a specific set of tools (the variables in the term) and each having its own value (the coefficient on that term). For example, the tools may be medical patents, and producing each medicine requires a specific set of patents. The value of a set of items to the agent is the sum of the values of the tasks that the agent can accomplish with those items. We show that *Toolbox DNF* preferences can be elicited in a polynomial number of value queries, and that given two agents with *Toolbox DNF* preferences, the items can be optimally allocated in polynomial time using network flow.

More broadly, in the combinatorial auctions literature, the issue of preference elicitation is often put as "The problem is easy if preferences are linear, and hard if preferences are arbitrary. What if preferences are *somewhat* linear?" Our answer is that if one defines "somewhat linear" as read-once (with certain types of gates) or as Toolbox DNF, then preference elicitation with value queries is easy, while still allowing preferences that exhibit interesting behavior.

4.2 Eliciting read-once formulas over {SUM, MAX, ALL}

Define a read-once formula to be **canonical** if no internal node in the tree has a child with the same label. It is not hard to see that for the node functions $\{\mathrm{SUM}, \mathrm{MAX}, \mathrm{ALL}\}$ We can assume without loss of generality that the formula is canonical. In particular, for these gates, if a node and its child are of the same type we can just merge them. As a warmup, consider the easier problem of eliciting when there are no ALL gates.

Lemma 4.1 *One can elicit read-once formulas over* $\{\mathrm{SUM}, \mathrm{MAX}\}$ *using* $O(n^2)$ *value queries.*

Proof: Let $S = \{1, \ldots, n\}$. The elicitor will ask two sets of questions:

1. For every $a \in S$, what is $f(\{a\})$? This is n questions.

2. For every pair $a, b \in S$ what is $f(\{a, b\})$? This is $n(n-1)/2$ questions.

[4] Notice that the multiplicative values at the leaves may not be uniquely specified: if there is only one flight and one hotel in a particular destination, and their combined value is \$1000, then this can be arbitrarily split into values $v, 1000 - v$ for the flight and hotel in the formula.

Notice that an item $a \in S$ is in the tree if and only if $f(\{a\}) > 0$. Call the set of items in the tree T. Second, notice that if the least common ancestor (LCA) of two inputs a and b is a SUM node, then $f(\{a,b\}) = f(\{a\}) + f(\{b\})$, whereas if the LCA is a MAX node, then $f(\{a,b\}) = \mathrm{MAX}(f(\{a\}), f(\{b\}))$. The elicitor can therefore use the answers to its queries to construct a graph G with the vertices representing the items in T, and an edge between two vertices if their least common ancestor is a MAX gate.

The elicitor now determines the structure of the tree root-down. Notice that if the root node of the formula is a SUM gate, then there will be no edges in G between any two vertices in different subtrees of the root and the graph will be disconnected. On the other hand, if the root node is a MAX gate, then every item in the first subtree will be connected to every item in the other subtrees. Since there are at least two subtrees, this means that if the root is a MAX, then the graph *is* connected.

So, if the graph is disconnected, the elicitor places a SUM gate at the root, partitions the items into subtrees according to the connected components of the graph, and then recursively solves to discover the structure of each subtree. Formally, the elicitor is using the fact that if f_i is the ith subtree and S_i are the items in subtree f_i, then for any $S' \subseteq S_i$, $f(S') = f_i(S')$. On the other hand, if the graph is connected, the elicitor places a MAX gate at the root, partition the items into subtrees according to the connected components of the *complement* of the graph, and then recursively solve for the subtrees. Here the elicitor is using the fact that in the complement of the graph, there is an edge between two nodes if and only if the lowest common ancestor is a SUM node, so the elicitor can use the same argument as before.

Finally, the elicitor sets the leaf multipliers v_i to the values given by the n unary queries asked in step 1. ■

We now proceed to our first main theorem.

Theorem 4.2 *One can elicit read-once formulas over* $\{\mathrm{SUM}, \mathrm{MAX}, \mathrm{ALL}\}$ *gates using* $O(n^2)$ *value queries.*

The high-level outline of the proof is as follows. First, notice that if one thinks of a value greater than zero as being *true* and a value equal to zero being *false*, then MAX and SUM act as OR, and ALL acts as AND. The elicitor can now apply an algorithm of Angluin et al. [AHK93] that exactly learns (elicits) read-once formulas of AND and OR gates using membership (value) queries to determine the AND/OR structure. Next the elicitor expands each OR gate back into a tree of MAX and SUM using the algorithm of Lemma 4.1. One complication is that to apply that technique here the elicitor needs to deal with two issues: (1) the elicitor only gets to observe the output of the full tree, not the specific subtree, and (2) the elicitor can only directly manipulate the inputs to the full tree, not the inputs to the specific subtree. Finally, the elicitor needs to find a consistent set of value multipliers for the inputs.

Definition 4.3 *Given a read-once formula f consisting of* SUM, MAX, *and* ALL *gates, define the* ***Boolean image*** *of f to be the function g where for all sets of items S, $g(S)$ is true if and only if $f(S) > 0$.*

As noted above, the Boolean image of a read-once formula f over $\{\mathrm{SUM}, \mathrm{MAX}, \mathrm{ALL}\}$ is equivalent to a read-once monotone Boolean formula with an AND gate wherever f has an ALL gate, and an OR gate wherever f has a MAX or SUM gate. This direct mapping may produce a non-canonical tree (because of a SUM node beneath a MAX node, for instance). The canonical Boolean image is the tree in which all subtrees of OR nodes have been merged together.

Definition 4.4 *Given a canonical real-valued read-once formula f and its canonical Boolean image g, define* $\mathrm{bool}(u)$ *for a node u in f to be its associated node in g. For a given node v in g, let $r(v)$ be the highest node in* $\mathrm{bool}^{-1}(v)$ *(the one closest to the root of f).*

Theorem 4.5 (Angluin et al. [AHK93]) *There is an algorithm that exactly identifies any monotone read-once formula over {OR, AND} using $O(n^2)$ queries.*

The elicitor can use Theorem 4.5 to elicit the canonical Boolean image of f, but the elicitor now needs to expand each OR node back into a subtree of {MAX, SUM}. The next two lemmas show how to do this.

Observe that the only test required to determine the structure of a {MAX, SUM} tree is a test of equality of value between two sets. Lemma 4.6 will describe how to perform such a test when the function one is interested in is not the root. Lemma 4.7 shows how to treat the inputs to this OR node like items to elicit the label of their least common ancestor.

Lemma 4.6 *Suppose one has g, the canonical Boolean image of f, and v is a node of g. Let $u = r(v)$. Then, for two sets of items S_a and S_b, one can determine if $u(S_a) = u(S_b)$ in two queries.*

Proof: Suppose that S' is the set of items that are descendants of v, and $Z = S - S'$ is all other items. Without loss of generality, the elicitor can assume $S_a, S_b \subseteq S'$. Notice that it is possible $f(S_a) = f(S_b)$ but $u(S_a) \neq u(S_b)$ if v has an ancestor labeled AND. It is also possible that $f(S_a \bigcup Z) = f(S_b \bigcup Z)$ but $u(S_a) \neq u(S_b)$ if u had a MAX gate as a parent.

Thus, the elicitor constructs a set R that is the items $x \in Z$ such that the LCA of v and x is an AND node. This is identical to the set of all items in Z that have an ALL node as a LCA with u. This means that for every ALL node that is an ancestor of u, its children that are not ancestors of u have positive output. Also, for every MAX node that is an ancestor of u, its children that are not ancestors of u have zero value. Thus, if c is the sum of the values of the children of the ancestors of u on input R, then $u(S_a)+c = f(S_a \bigcup R)$ or $u(S_a) = f(S_a \bigcup R) = 0$. Thus, $u(S_a) = u(S_b)$ if and only if $f(S_a \bigcup R) = f(S_b \bigcup R)$. ∎

Lemma 4.7 *Suppose one has g, the canonical Boolean image of f, and suppose v_1 and v_2 are siblings in g with an OR parent. Then one can determine whether the LCA of $r(v_1)$ and $r(v_2)$ is a* MAX *or* SUM *in four queries.*

Proof: Let v_3 be the parent of v_1 and v_2, and let $u_i = r(v_i)$. Observe that no node on the path between u_1 and u_3 is an ALL gate, and no node on the path between u_2 and u_3 is an ALL gate. Also, the LCA of u_1 and u_2 is a descendent of u_3. Suppose that S_i is the set of items that are descendants of v_i. The LCA of u_1 and u_2 is a MAX node if and only if $u_3(S_1 \bigcup S_2) = u_3(S_1)$ or $u_3(S_1 \bigcup S_2) = u_3(S_2)$. Using Lemma 4.6, this can be tested using four queries.[5] ∎

Lemma 4.8 *Suppose one has g, the canonical Boolean image of f, and an internal node v in g with k children. Then, one can determine the subtree corresponding to* $\mathrm{bool}^{-1}(v)$ *in $2k(k-1)$ queries, and how the leaves of that subtree map to the children of v.*

Proof: If v is an AND gate, then $\mathrm{bool}^{-1}(v)$ is an ALL gate. So, the elicitor simply constructs one ALL gate with children that are leaves labeled with the children of v.

If v is an OR gate, define $u = r(v)$. Define $v_1, \ldots, v_k$ to be the children of v, and define $u_i = r(v_i)$. Using Lemma 4.7, the elicitor can determine whether the LCA of each pair u_i, u_j is a MAX or SUM in 4 queries for a total of $4k(k-1)/2 = 2k(k-1)$ queries. The elicitor can now apply the graph decomposition technique from the proof of Lemma 4.1 to compute the whole subtree. ∎

Lemma 4.9 *Given the structure of a read-once formula f over* $\{\mathrm{MAX}, \mathrm{SUM}, \mathrm{ALL}\}$*, one can determine a consistent set of values (multipliers) on the leaves using at most $3n$ value queries, where n is the number of items.*

The proof is in Section 4.7.

Proof (of Theorem 4.2): The overall algorithm works as follows. Apply the algorithm from [AHK93] to get the Boolean image of g. Use Lemma 4.8 to find the fine structure of f. Observe that overall all of the internal nodes have less than $2n$ total children, so the total number of queries is less than $4n(2n-1)$. Then use Lemma 4.9 to find the weights. ∎

[5] In reality, one can use less than four queries per test.

4.3 Eliciting read-once preferences over more general operators

The elicitor can also elicit preferences with more general gates that we call ATLEAST$_k$, MAX$_k$, and GENERAL$_{k,l}$. An ATLEAST$_k$ node returns the sum of its inputs if it receives a positive value on at least k inputs. Otherwise, it returns zero. This is a generalization of the ALL node. A MAX$_k$ node returns the sum of the k highest-valued inputs. A GENERAL$_{k,l}$ gate returns the sum of the l highest-valued inputs if and only if at least k inputs are positive, otherwise it returns zero. We restrict k to be less than or equal to l. Every read-once gate discussed in this chapter is a specific instance of a GENERAL$_{k,l}$ gate.

For instance, imagine that on a vacation to the Bahamas, Alice wanted entertainment. If she got to go out on at least three nights, then the trip would be worthwhile. Otherwise, she would rather stay home. Each night, she takes the maximum valued entertainment option. Then there is an ATLEAST$_3$ node combining all of the different nights.

In a different situation, imagine that Joe wants a more relaxing vacation in Hawaii, where he does not want to go out more than three nights. In this case, a MAX$_3$ gate will be useful. For each night, he chooses the best possible entertainment given to him. Then, he takes the best three nights of entertainment.

Finally, imagine that Maggie wants a moderately active vacation, and is interested in going to Paris for a week, and wants at least three but no more than four nights of entertainment. Then a GENERAL$_{3,4}$ gate will describe her preferences.

Theorem 4.10 *A read-once* GENERAL$_{k,l}$ *function can be learned in polynomial time.*

The proof sketch is in Section 4.9.

4.4 Allocation with read-once preferences

Suppose there are two parties with preference functions f and g and the elicitor wants to maximize social welfare, that is, find the allocation $(A, S - A)$ that maximizes $f(A) + g(S - A)$. In this section we show that if one of these functions (say, g) is additive (that is, the value of a bundle is the sum of the values of the items in the bundle) and the other (say, f) is read-once, then the elicitor can find the optimal allocation in polynomial time. However, if both f and g are read-once formulas—even containing just ALL, MAX, and SUM gates—then the allocation problem is NP-hard.

Another way to think of maximizing social welfare when g is linear is to think of g as a “cost” function, and the elicitor is maximizing $f(A) - g(A)$. For example, a software agent wants to find the set of items A that maximizes the difference between the value of A to its user and the cost of A, when items are each purchased separately.

4.4.1 Read-once valuation function and additive valuation function

Theorem 4.11 *Given one party with a known* MAX-SUM-ALL *read-once valuation, and another party with a known additive valuation, there exists a polynomial-time algorithm for maximizing their joint welfare.*

Proof: The idea is that the elicitor recursively learns two things about each node u of f: what subset S' of the items that are descendants of u maximizes $u(S') - g(S')$, and what subset S'' of the items that are descendants of u maximizes $u(S'') - g(S'')$ given the restriction that u is positive.

Observe that this is trivial if u is a leaf. If not, assume that $u_1, \ldots, u_k$ are the children of u, and that S_i are the items that are descendants of u_i. Define $S'_i \subseteq S_i$ to be the items that maximize $u_i(S'_i) - g(S'_i)$, and $S''_i \subseteq S_i$ to be the items that maximize $u_i(S''_i) - g(S''_i)$ given that $u_i(S''_i)$ is positive.

If u has an ALL label, then $S'' = \bigcup_{i=1}^{k} S''_i$. If $u(S'') - g(S'') > 0$, then $S' = S''$. Otherwise, $S' = \emptyset$. This justifies the need for maintaining both sets.

If u has a SUM label, then $S' = \bigcup_{i=1}^{k} S'_i$. Define j to be the index of the child of u that loses the least from being positive. More formally,

$$j = \operatorname*{argmin}_{i \in \{1 \ldots k\}} \left(u_i(S'_i) - g(S'_i)\right) - \left(u_i(S''_i) - g(S''_i)\right).$$

Then the elicitor finds that $S'' = S''_j \bigcup \left(\bigcup_{i \neq j} S'_j \right)$.

If u has a max label, then define a to be the index of the best S'_i, or more formally:

$$a = \operatorname*{argmax}_{i \in \{1 \ldots k\}} u(S'_i) - g(S'_i).$$

Then $S' = S'_a$. Define b to be the index of the best S''_i, or more formally:

$$b = \operatorname*{argmax}_{i \in \{1 \ldots k\}} u(S''_i) - g(S''_i).$$

Then $S'' = S''_b$.

Each of these sets can be found in polynomial time, so the runtime is polynomial. ■

Theorem 4.12 *Given one party with a known* GENERAL$_{k,l}$ *read-once valuation, and another party with a known additive valuation, there exists a polynomial-time algorithm for maximizing their joint welfare.*

The proof is similar to the one above.

4.4.2 NP-hardness of allocation among two parties

Theorem 4.13 *Given two agents with known valuations which are* MAX-SUM-ALL *read-once functions, it is NP-hard to find an allocation whose welfare is more than* $\frac{1}{2}$ *of the welfare of the optimal allocation.*

The proof is in Section 4.10.

Notice that achieving welfare of *at least* half the optimal is easy because the elicitor can simply give all the items to the agent who values the total the most. This shows that it is NP-hard to do any better.

4.5 Learning preferences that are *almost* read-once

In this section, we will consider the setting where a person's valuation function is a δ-approximation of a read-once function: given that the person's valuation function is f, there exists a read-once function f' such that for all sets S, then $|f(S) - f'(S)| < \delta$. This algorithm works only for the case of read-once formulas over $\{\mathrm{MAX}, \mathrm{SUM}\}$.

Observe that, for any class of polynomial-size circuits (like those described in the previous sections), the set of preference functions that can be represented has a low topological dimension in comparison to the topological dimension of the class of all preference functions. However, the set of approximate $\{\mathrm{MAX}, \mathrm{SUM}\}$ functions has the same dimension as the set of all preference functions.

Theorem 4.14 *Given black-box access to f, a δ-approximation of a read-once function consisting of* MAX *and* SUM *nodes, a function g can be learned in $n(n-1)/2$ queries such that for any set of items S', it is the case that $|g(S') - f(S')| < 6\delta|S'| + \delta$.*

Proof: Define f' to be the read-once function such that $|f - f'| < \delta$, and let $v_a = f'(\{a\})$. The key behind this algorithm is that the elicitor will throw away all items a where $v_a < 4\delta$, because they will interfere with the LCA test. In order to achieve this, the elicitor throws away a if $f(\{a\}) < 5\delta$. Observe that if $v_a \geq 6\delta$, then a will not be thrown away.

We now argue that the test for LCA is correct if all the items a have a value $v_a \geq 4\delta$ or more. Consider the following test for the LCA of a and b: a and b have a max node as an LCA if and only if $f(\{a, b\}) \leq f(\{a\}) + 2\delta$ or $f(\{a, b\}) \leq f(\{b\}) + 2\delta$.

Assume that the LCA is a max node. Then $f'(\{a,b\}) = f'(\{a\})$ or $f'(\{a,b\}) = f'(\{b\})$. Without loss of generality, assume that $f'(\{a,b\}) = f'(\{a\})$. Then:

$$\begin{array}{rcl} f(\{a,b\}) & < & f'(\{a,b\}) + \delta \\ f(\{a,b\}) & < & f'(\{a\}) + \delta \\ f(\{a,b\}) & < & (f(\{a\}) + \delta) + \delta \\ f(\{a,b\}) & < & f(\{a\}) + 2\delta \end{array}$$

Assume that the LCA is a SUM node. Then $f'(\{a,b\}) = f'(\{a\}) + f'(\{b\}) > f'(\{a\}) + 4\delta$. This implies that $f(\{a,b\}) > f(\{a\}) + 2\delta$. Similarly, $f(\{a,b\}) > f(\{b\}) + 2\delta$.

Given this LCA test, the elicitor can learn the structure of the tree as in Lemma 4.1. The elicitor associates the value $f(\{a\})$ to the leaf with item a, even though this value might be off by almost δ. Now, observe that if the elicitor throws away items not in S', then this does not affect the difference between $f'(S')$ and $g(S')$. Also, incorrectly calculating the value of an item not in S' does not affect this difference. Now, for each item in S', the elicitor can get an error at most less than 6δ, because the elicitor threw out an item of value just below 6δ. Also, there is a difference of less than δ between f' and f. ∎

4.6 Toolbox DNF

We now consider another natural class of preferences that we call Toolbox DNF, that can be elicited in polynomial time via value queries. In **Toolbox DNF**, each bidder has an explicit list of m bundles $S_1, \ldots, S_m$ called **minterms**, with values $v_1, \ldots, v_m$ respectively. The value given to a generic set S' is assumed to be the *sum* of values of the S_i contained in S'. That is,

$$v(S') \quad = \quad \sum_{S_i \subseteq S'} v_i.$$

These preferences are natural if, for example, the items are tools or capabilities and there are m tasks to perform that each require some subset of tools. If task i has value v_i and requires set of tools S_i, then these preferences represent the value that can be attained by any given collection of tools. We show that Toolbox-DNF can be elicited in time polynomial in the number of items and the number of minterms.

Theorem 4.15 *Toolbox DNF can be elicited using $O(mn)$ value queries.*

Proof: The elicitor will find the minterms one at a time in an iterative fashion. The elicitor can clearly test for the presence of *some* minterm by simply testing if $v(S) > 0$. The elicitor can then repeatedly remove elements to find a minimal positive set S_1 in n queries. That is, $v(S_1) > 0$ but for all $x \in S_1$, $v(S_1 - \{x\}) = 0$. This will be the elicitor's first minterm, and the elicitor will set $v_1 = v(S_1)$.

At a generic point in time, the elicitor will have found minterms $S_1, \ldots, S_i$ with associated values $v_1, \ldots, v_i$. Let v^i be the Toolbox-DNF given by the terms found so far; that is,

$$v^i(S') = \sum_{S_j \subseteq S' : j \leq i} v_j.$$

Define $\bar{v}^i = v - v^i$. Observe that $\bar{v}^i$ is also a toolbox function, namely, $\bar{v}^i(S') = \sum_{S_j \subseteq S' : j > i} v_j$. Also, the elicitor can "query" $\bar{v}^i$ at any set S' by querying $v(S')$ and subtracting $v^i(S')$. Thus, the elicitor can find a minterm of $\bar{v}^i$ using the same procedure as above, and this will be a new minterm of v. The elicitor simply continues this process until at some point the elicitor finds that $\bar{v}^m(S) = 0$ and the elicitor is done. ∎

Theorem 4.16 *Given two agents with Toolbox-DNF preferences having m_1 and m_2 minterms respectively, optimal allocation can be done in time polynomial in n and $m_1 + m_2$.*

Proof: Construct a node-weighted bipartite graph with one node on the left for each of the first agent's minterms, and one node on the right for each of the second agent's minterms. Give node i a weight v_i (where v_i is defined with respect to the associated agent). An edge is drawn between a node on the left and a node on the right if the associated minterms share any items.

Notice that any independent set I on this graph represents a legal allocation. The first agent gets all items in minterms associated with nodes in I on the left, and the second agent gets all items in minterms associated with nodes in I on the right. This is legal because that if an item is in a node on the left and a node on the right, then there is an edge between these nodes, and only one of them would be in I. If remaining items are thrown away, then the total welfare of this allocation is equal to the total weight of independent set I.

Similarly, any legal allocation corresponds to an independent set of the same total weight. If the allocation gives set A_1 to agent 1, and A_2 to agent 2, then just pick the nodes on the left whose bundles are in A_1, and the nodes on the right whose bundles are in A_2.

Thus, the allocation problem reduces to finding a maximum weight independent set in a bipartite graph, which can be solved via maximum flow. Specifically, it is a standard result that the complement of I, which is a minimum-weight vertex cover, corresponds directly to the minimum cut in an associated flow-network. ∎

4.7 Learning the Weights*

There are some situations where there are multiple consistent values for the leaves of the function f. For example, if a plane ticket and hotel are fed into an ALL gate, and together they are worth \$1000, then any pair of values $(x, 1000 - x)$ are consistent for the leaves. The algorithm will be guaranteed to learn *some* consistent set of values.

The elicitor will use a recursive technique to learn the tree. At each node, the elicitor will either learn the function f' at that node, or a function f'' such that there exists some $c \in \mathbf{R}$ such that for all S'', if $f'(S'') > 0$ then $f'(S'') = f''(S'') + c$, otherwise $f''(S'') = 0$. We call such an f'' a *shift* of function f', even though the shift is only on the inputs where $f' > 0$. In the case that we learn a shift f'', the function that we learn will be "shiftable" in the sense that it will be easy to make an additive increase or decrease of the value of the function on all positive inputs by recursively shifting the values of the subfunctions. This will be done as a postprocessing step. An example of a function that is not shiftable is a SUM node with two leaves as children.

For easier recursive application, we will allow leaves to have positive and negative values, and find a function where each node computes a real value and a boolean value. If the boolean value is false, the real value will be zero. Leaves will return *true* and their associated real value if the item is in the input, $false$ and zero otherwise. SUM nodes will return the OR of the boolean values of their children and the sum of their real values. MAX nodes will return the OR of their children and the highest real value of a *true* child if this is true, otherwise zero. This means that if one child has a negative real value and is *true*, and one child has a zero real value and is $false$, MAX will return the negative value. ALL will return the AND of its children and the sum of its children's real values. The result of the tree is the real value returned by the root. Observe that this is identical to the original if the values of the leaves are all positive.

Definition 4.17 *A shiftable node is defined recursively.*

1. *A leaf is shiftable.*
2. *A* SUM *node is never shiftable.*
3. *A* MAX *node is shiftable if all of its children are shiftable.*
4. *An* ALL *node is shiftable if at least one of its children is shiftable.*

Define a tree to be shiftable if its root is shiftable, and a function to be shiftable if the representative tree is shiftable.

Lemma 4.18 *Suppose that f is a read-once function that is shiftable. The elicitor can construct a new read-once function f' such that $f' = f + c$ for all inputs where f returns a true value.*

Proof: The elicitor performs this construction recursively, only modifying the values of the leaves. For a leaf node, it is sufficient to add c to its value. This is the base case.

If the tree is shiftable and not a leaf, the read-once function has a MAX or ALL label at the root.

Assume the root is a MAX node. In this case, the elicitor knows that the value of the tree, if it is *true*, will be the value of one of the subfunctions $f_1, \ldots, f_k$. Observe that all subfunctions are shiftable. Thus, by the inductive hypothesis, the elicitor can change each subfunction such that $f_i' = f_i + c$.

Assume the root is an ALL node. In this case, the elicitor knows that the value of the tree, if it is *true*, will be the value of the sum of the subfunctions $f_1, \ldots, f_k$. Now, there must exist some subfunction f_i that is shiftable. Therefore, the elicitor can change the value of this subfunction such that $f_i' = f_i + c$, and leave the remaining subfunctions unchanged. ∎

Lemma 4.19 *Given the structure of a read-once function f and a node N, suppose that f' is the true function associated with N. One can determine values for the leaves that are descendants of N such that the resulting function f'' is a shift of f', using at most $(3/2)n$ queries, where n is the number of nodes that are descendants of N. If f' is not shiftable, then we guarantee that $f'' = f'$.*

Proof: This is proven inductively on the depth of the tree. This holds for a leaf. Now consider trees of larger depth. Define S' to be the items that are descendants of N. The first step is to show that the elicitor can construct a set R similar to that of Lemma 4.6. Define R to be the set of all items that have an ALL gate as an LCA with N and are not descendants of N.

Thus, in an argument similar to Lemma 4.6, there exists a $c \in \mathbf{R}$ such that for all $S'' \subseteq S'$, $f(S'' \cup R) = f'(S'') + c$.

Now, define $N_1, \ldots, N_k$ to be the children of N. Define f_i' to be the function associated with each N_i, and S_i to be the items that are descendants of N_i. Recursively for each subtree, we can discover a f_i such that there exists a $c_i \in \mathbf{R}$ where for all S, $f_i(S) = f_i'(S) + c_i$. In order to obtain these functions, the elicitor requires $(3/2)(n - k)$ queries.

Suppose that N has a MAX label. Then if one of the subtrees, say i, is not shiftable, then $f_i(S_i) = f_i'(S_i) = f'(S_i)$, so $f_i(S_i) + c = f(S_i \cup R)$. Thus, with one query the elicitor can discover c. Then, for each subtree f_j that is shiftable, the elicitor can find c_j. It is true that $f_j'(S_j) + c = f(S_j \cup R)$, and so $c_j = f_j(S_j) - f_j'(S_j) = f_j(S_j) - (f(S_j \cup R) - c)$. Thus, the elicitor can find a function equal to f_j' when N_j is true by finding $f_j'' = f_j - c_j$. This requires overall less than or equal to k queries.

However, if N has a MAX label and all the subtrees are shiftable, then the tree is shiftable. However, the elicitor can still construct a set of functions f_i'' such that for all i, for all $S'' \subseteq S'$ such that N_i is true, $f_i''(S'') = f_i'(S'') + c$. This can be done by defining $c_i' = f_i(S_i) - f(S_i \cup R)$, and then $f_i'' = f_i - c_i'$. Observe that $f_i'' = f_i - (f_i - (f_i' + c)) = f_i' + c$ when the input makes N_i true.

Suppose that N has an ALL label. Then the elicitor directly makes $f'' = \mathrm{ALL}_{i=1}^k f_i'$. N is *true* if and only if all N_i are true, and in this case $\sum_{i=1}^k f_i' = \sum_{i=1}^k (f_i + c_i) = (\sum_{i=1}^k f_i) + (\sum_{i=1}^k c_i)$. Thus, if all of the subtrees of N_i are not shiftable, then $f'' = f'$.

The most complex case is when N has a SUM label. In this case, the elicitor can learn the value of each subtree exactly. The elicitor measures $f(S_1 \cup S_2 \cup R)$, $f(S_1 \cup R)$, and $f(S_2 \cup R)$. Observe that

$$\begin{aligned} f(S_1 \cup S_2 \cup R) - f(S_2 \cup R) &= f'(S_1 \cup S_2) - f'(S_2) \\ &= f_1'(S_1). \end{aligned}$$

If $k > 2$, then for each $i > 2$ the elicitor measures $f(S_1 \cup S_i \cup R)$. Thus, for each $1 \le i \le k$ such that the ith subtree is shiftable, we can calculate $c_i' = f_i(S_i) - f_i'(S_i)$, and construct $f_i'' = f_i - c_i'$. This requires at most $k + 1 \le 3k/2$ queries. ∎

Proof (of Lemma 4.9): One can elicit values such that there exists a c such that if the root is true, $f'(S') = f(S') + c$ using $(3/2)(m-1)$ queries, where m is the number of nodes in the tree (which can be no more than twice the number of items). If f is not shiftable, the elicitor is done. If f is shiftable, one can use a single query on the set of all items S to find $c = f'(S) - f(S)$. Then one can compute $f' - c$. ■

4.8 Learning a MAX_l Tree*

Assume that the tree is canonical, that no MAX_1 node v has a MAX_1 child v', because the children of v' can be made children of v. Also, no node is labeled MAX_0, because such nodes and their descendants can be removed from the tree without affecting the output.

A large number of definitions are listed below, grouped for ease of reference. S', S'' are sets of items and v, v' are nodes.

- Define r to be the root node.
- Define $T = \{a \in S : f(a) > 0\}$.
- Define $L(v)$ to be l if v is labeled MAX_l.
- Define $C(v)$ to be the children of v.
- Define $D(v)$ to be the items that are descendants of v.
- Define $DP(v) = \{D(v') : v' \in C(v)\}$.
- Define $C(v, S') = \{v' : v' \in C(v), D(v') \cap S' \neq \emptyset\}$.
- A set S' is **dependent** if there exists some $a \in S'$ such that $f(S') = f(S' \backslash \{a\})$.
- A set is **independent** if it is not dependent.
- A set S' is a **minimal dependent set** if it is dependent and there exists no dependent proper subset $S'' \subset S'$.
- A set S' is **pairwise independent** if no subset of S' of size 2 is dependent.
- A node v is a **witness** for a set S' if $L(v) < |C(v, S')|$.
- A set S' **represents** v if $L(v) + 1 = |S'| = |C(v, S')| + 1$.
- Given a graph G which has the items in T as nodes, G is **legal** if for all $a, b \in T$ that are connected, $LCA(a, b) \neq r$.
- Given a graph G of T, define $R(G)$ to be the collection of sets such that for each set $S' \in R(G)$, S' contains exactly one item from each connected component in G.

The elicitor will attempt to find $DP(r)$ and $L(r)$ using a polynomial number of value queries[6]. The elicitor will do this by constructing a legal graph and then adding edges to it. The intermediate outcome of the algorithm depends on $L(r)$:

- $L(r) = 1$: The elicitor will quickly find that $L(r) = 1$ and be able to use an applicable trick.
- $L(r) = |C(r)|$: $DP(r)$ will be the connected components of the graph.
- $1 < L(r) < C(r)$. The elicitor will eventually find a set S' that represents r, and use it to find $DP(r)$.

[6]In the remaining discussion, the phrases “The elicitor can” or “one can” will indicate one can find some set, partition, or value with a polynomial number of value queries.

The elicitor will not need to know $L(r)$ in advance, the elicitor will find it in the course of the algorithm. Below there are some useful lemmas about witnesses, representatives and dependent sets. The algorithm follows directly.

Fact 4.20 *1. A set S' is dependent if and only if it has a witness.*

2. If S' is dependent and $S' \subseteq S''$, then S'' is dependent.

3. If S' is independent and $S' \supseteq S''$, then S'' is independent.

4. A minimal dependent set S' represents some node v.

5. A set S' that represents some node v is a minimal dependent set.

6. A set $\{a, b\} \subseteq T$ is dependent if and only if $L(LCA(a, b)) = 1$.

7. The subset of a pairwise independent set is pairwise independent.

8. If S' is pairwise independent, then it has no witnesses v where $L(v) = 1$.

Lemma 4.21 *The elicitor can determine if $L(r) = 1$. If $L(r) = 1$, the elicitor can find $DP(r)$. If $L(r) > 1$, the elicitor can find a legal graph G such that for all $S' \in R(G)$, S' is pairwise independent.*

Proof: The elicitor constructs a graph with nodes in T, connecting two items $\{a, b\}$ if and only if they are a dependent set. By an argument similar to that of Lemma 4.1, if G is connected, then the root is a MAX_1 node. In this case, the connected components of the complement of G is $DP(r)$, and $L(r) = 1$.

If G is not connected, then the root is not a MAX_1 node. Thus, $L(r) > 1$. For all edges (a, b) in the graph G, $L(LCA(a, b)) = 1$, and therefore $LCA(a, b) \neq r$.

Consider an arbitrary $S' \in R(G)$. If $\{a, b\} \subseteq S'$, then there is no edge between a and b. Thus, $\{a, b\}$ is independent, and S' is pairwise independent. ■

The elicitor will be constructing more legal graphs by adding edges.

Fact 4.22 *If all $S' \in R(G)$ are pairwise independent, and G' has all of the edges of G, then all $S' \in R(G)$ are pairwise independent.*

The primary reason that dependent sets are useful is because they are easily found.

Fact 4.23 *Given a set S'', one can find a minimal dependent set S' such that $S' \subseteq S''$.*

Lemma 4.24 *Given a pairwise independent set S'', a minimal dependent set $S' \subseteq S''$ that represents some unknown node v, and an element $a \in S''$, one can determine if $a \in D(v)$. Specifically, $a \in D(v)$ if and only if there exists some $b \in S'$ such that $S' \backslash \{b\} \cup \{a\}$ is dependent.*

Corollary 4.25 *One can find $D(v) \cap S''$.*

Corollary 4.26 *Given a legal graph G, a pairwise independent set $S'' \in R(G)$, and a minimal dependent set $S' \subseteq S''$, one can find if S' represents r.*

Proof: If $a \in D(v)$, then either:

- For all $b \in S'$, $LCA(a, b) = v$. Then for all $b \in S'$, $(S \backslash \{b\}) \cup \{a\}$ represents v and is dependent.
- There exists some $b \in S'$ where $LCA(a, b) \neq v$. In this case, $(S \backslash \{b\}) \cup \{a\}$ represents v and is dependent.

If $a \notin D(v)$, then take an arbitrary $b \in S'$, and define $S''' = S' \backslash \{b\}$. The elicitor will prove that $S''' \cup \{a\}$ does not have a witness.

First, observe that $L(v) = |C(v, S')| - 1 = |C(v, S''')| = |C(v, S''' \cup \{a\})|$. Observe that S''' is independent. For all $v' \neq v$, $C(v', S''') \leq 1$. Thus, $C(v', S''' \cup \{a\}) \leq 2$. However, since $S''' \cup \{a\} \subseteq S''$ is pairwise independent, it has no witness where $L(v') = 1$, so it cannot have any witness. ∎

Lemma 4.27 *Given a pairwise independent set S'', a minimal dependent set $S' \subseteq S''$, an element $a \in D(v) \cap (S'' \backslash S')$, and an element $b \in S'$, then $LCA(a, b) \neq v$ if and only if $(S' \backslash \{b\}) \cap \{a\}$ is dependent and for all $c \in S' \backslash \{b\}$, $(S' \backslash \{c\}) \cap \{a\}$ is independent.*

Corollary 4.28 *Given a legal graph G, a pairwise independent set $S'' \in R(G)$, and a minimal dependent set $S' \subseteq S''$ that represents v, one can find $DP(v)$.*

The proof is similar to that above.

Lemma 4.29 *The elicitor can learn* MAX_l *tree using a polynomial number of value queries.*

The algorithm is as follows:

1. Begin with a legal graph G such that all $S' \in R(G)$ are pairwise independent.
2. Choose an $S' \in R(G)$.
3. If S' is independent, then $L(r) = |C(r)|$, and the components of the graph G are $DP(r)$.
4. If S' is dependent, then find a minimal dependent set $S'' \subseteq S'$.
5. If S'' represents the root, then $L(r) = |S''| - 1$, and one can find $DP(r)$.
6. If S'' does not represent the root, then for all pairs $a, b \in S''$, add an edge (a, b) in G. Continue from step 2.

This algorithm will terminate because $|S''| > 1$, and therefore the elicitor is always adding edges to the graph in step 6.

4.9 Proof Sketch for $\text{GENERAL}_{k,l}$ Preferences*

Observe that the boolean image of a $\text{GENERAL}_{k,l}$ node is a $THRESH_k$ node, a node that is true if at least k inputs are true, and false otherwise. So, the boolean image of a read-once $\text{GENERAL}_{k,l}$ function is a read-once $THRESH_k$ function. An algorithm to learn such functions is discussed in [BHHK94]. Like learning a $\{\text{SUM}, \text{MAX}, \text{ALL}\}$ function, when the elicitor is learning a $\text{GENERAL}_{k,l}$ function, the elicitor can begin by learning the boolean image.

$THRESH_1$ nodes represent one or many $\text{GENERAL}_{1,l}$ nodes, just like OR nodes can represent one or many MAX or SUM nodes. Also, $THRESH_k$ nodes, when $k > 1$, represents exactly one $\text{GENERAL}_{k,l}$ node for some $l \geq k$.

The elicitor can calculate the output of a specific node to within an additive factor, by forming a set like in Lemma 4.6. Using equality tests, one can learn the MAX_l structure corresponding to a $THRESH_1$ node. Thus, with an equality test, the elicitor can learn the structure of the original function.

As before, the weights can be computed in polynomial time. The elicitor will recursively learn a function for each node that may be off by a fixed constant value on sets where the original function is positive. Again, the elicitor can think about shifting subfunctions as necessary. The concept of a shiftable node is defined recursively.

1. A leaf is shiftable.
2. Suppose an internal node is labeled $\text{GENERAL}_{k,l}$ and has m children.

(a) If $k = l < m$ and all the children are shiftable, the node is shiftable.
(b) If $k = l = m$ and there is a shiftable child, the node is shiftable.
(c) Otherwise, the node is not shiftable.

The tree is learned recursively from the bottom. One can learn a shiftable tree within a constant factor on all positive inputs and a tree that is not shiftable exactly.

The elicitor can recursively learn the function for each node, perhaps being off by a constant value on the positive values. As before, if a node is not shiftable, the elicitor can learn the function exactly. If $k < l$, then it is analogous to a SUM gate. If $k = l = m$, then it is analogous to the ALL gate. If $k = l < m$, then it is loosely analogous to the MAX gate. This is the hardest case where the elicitor solves a simple system of linear equalities in order to determine how to much to shift each subfunction.

4.10 Proving Approximate NP-Hardness*

Begin with a SAT instance, which will be translated into an instance of optimally allocation. The maximum global welfare will be 2 if the instance is satisfiable, and 1 otherwise. Thus, in order to get more than 1/2 of the optimal welfare, an algorithm must solve an NP-complete problem. Let $\{C_1, \dots C_k\}$ denote the clauses of the given formula, and let $X = \{x_1, \dots, x_n\}$ denote the variables.

The plan is that Agent I's valuation will be 1 if the allocation represents an assignment to the variables and 0 otherwise. Agent II's valuation can be equal to 1 if the allocation represents a satisfying assignment, and will be 0 if it does not represent a satisfying assignment. If the allocation does not represent an assignment, the value is between zero and 1 inclusive.

A variable x_i will be represented by two sets of items: $P_i = \{p_i^1, \dots, p_i^k\}$ and $N_i = \{n_i^1, \dots, n_i^k\}$. If Agent I has all the items in set N_i, then x_i is "positive". If Agent I has all the items in set P_i, then the variable is "negative". If every variable is either positive or negative, then the allocation is called "legal". The function:

$$f_i = \text{MAX}\left(\text{ALL}_{j=1}^k p_i^j, \text{ALL}_{j=1}^k n_i^j\right)$$

is k if x_i is positive or negative, zero otherwise. Agent I's valuation is:

$$v_I = \frac{1}{nk}\text{ALL}_{i=1}^n f_i.$$

Observe that $v_I = 1$ if each variable is positive or negative, and $v_I = 0$ otherwise.

Now, the set D_j will be used to represent the clause C_j. D_j is defined as follows: p_i^j is in D_j if and only if x_i occurs positively in C_j. n_i^j is in D_j if and only if x_i occurs negatively in C_j. So, given that the allocation is legal, observe that:

$$\max_{a \in D_j} a$$

is 1 only if there is a variable occurring positively in j which is not negative according to the allocation, or a variable occurring negatively in j which is not positive according to the allocation. Also, this function is never more than 1. The valuation of Agent II is:

$$v_{II} = \frac{1}{k}\text{ALL}_{j=1}^k \text{MAX}\{a : a \in D_j\}.$$

This function is never greater than 1. If the allocation is legal but does not correspond to a satisfying assignment, the value is zero.

Suppose that there is a satisfying assignment. Then for each variable x_i, if it is positive give all p_i^j to Agent II and all n_i^j to Agent I, and if it is negative give all n_i^j to Agent II and all p_i^j to Agent I. This allocation has a value 2.

Thus, if the assignment is not legal, the first agent has value 0 and the total value is less than or equal to 1. If the assignment is legal but not a satisfying assignment, the total value is 1. The optimal global welfare is 2 if there is a satisfying assignment, and 1 otherwise.

4.11 Conclusions and future work

Elicitation of real-valued preferences is a key capability in many allocation problems, especially in electronic commerce. When multiple items are to be allocated, preference elicitation is difficult because the parties' preferences over items are generally not additive due to complementarity and substitutability. In this chapter we studied the elicitation of real-valued preferences over subsets of items, showing that *read-once* preferences over a natural set of gates (a restriction that is still powerful enough to capture complementarity and substitutability), and toolbox DNF preferences, can be elicited in a polynomial number of value queries. Such elicitation can be used by a shopping agent to elicit its user's preferences. It can also be used to elicit bidders' preferences in a combinatorial auction (answering the value queries truthfully can be made an *ex post equilibrium* by using Clarke pricing [CS01]—assuming the agent has read-once preferences). We also showed that if the party's preferences are close to read-once, then a good approximation of the preferences can be elicited quickly.

We also studied the complexity of the computational problem of allocating the items given the parties' preferences. We showed that this is NP-hard even with just two parties with read-once valuations. However, in the natural setting where only one of the parties has a read-once valuation and the other has an additive valuation function, the allocation problem is solvable in polynomial time.

There are several interesting avenues for future research along these lines. For one, in these multi-item allocation problems, what is the limit on the generality of preferences that can be elicited in polynomial time via value queries? Second, when there are multiple parties, one agent's preferences can be used to decide what information needs to be elicited from another party in order to determine an optimal (or approximately optimal) allocation of items. This has been the driving motivation in the work on preference elicitation in combinatorial auctions [CS01, CS02b, CS02a, HS02, HS04], but our work in this chapter did not yet capitalize on this extra power. In the future, it would be interesting to harness this power, together with the possibilities that preference restrictions open, to design effective goal-driven preference elicitation algorithms.

Part II

Learning in Games

Chapter 5

Online Convex Programming[1]

Convex programming involves a convex set $F \subseteq \mathbf{R}^n$ and a convex cost function $c : F \to \mathbf{R}$. The goal of convex programming is to find a point in F which minimizes c. In online convex programming, the convex set is known in advance, but in each step of some repeated optimization problem, one must select a point in F before seeing the cost function for that step. This can be used to model factory production, farm production, and many other industrial optimization problems where one is unaware of the value of the items produced until they have already been constructed. We introduce an algorithm for this domain. We also apply this algorithm to repeated games, and show that it is really a generalization of infinitesimal gradient ascent, and that generalized infinitesimal gradient ascent (GIGA) achieves no-external-regret: in other words, the algorithm almost surely in the limit does almost as well as the best fixed action.

5.1 Introduction

Imagine a farmer who decides what to plant each year. She has certain restrictions on her resources, both land and labor, as well as restrictions on the output she is allowed to produce. How can she select which crops to grow without knowing in advance what the prices will be? Here we present an algorithm based on gradient descent that will earn her almost as much as she could given that she knew all prices in advance but produced the same amount of each crop year after year. This is an example of an online convex programming problem.

Online convex programming is a generalization of the well-studied experts problem [FS99, LW89]. Imagine that a decision maker has n experts, each of which has a plan at each step with some cost. At each time step, the decision maker selects a probability distribution over experts. If $x \in \mathbf{R}^n$ is defined such that x_i is the probability that he selects expert i, then the set of all probability distributions is a convex set. Also, the cost function on this set is linear, and therefore convex.

This chapter presents an algorithm for general convex functions based on gradient descent. The algorithm applies gradient descent in $\mathbf{R}^n$, and then moves back to the set of feasible points. There are three beneficial properties of this algorithm. The first is that gradient descent is a simple, natural algorithm that is widely used, and studying its behavior is of intrinsic value. Secondly, this algorithm is more general than the experts setting, in that it can handle an arbitrary sequence of convex functions, which had not been solved. Finally, in online linear programs, this algorithm can in some circumstances perform better than an experts algorithm. While the bounds on the performance of most experts algorithms depends on the number of experts, these bounds are based on other criteria, which may sometimes be better. This relationship is discussed further in Section 5.4, and further comments on related work can be found in Section 5.5. The main theorem is stated and proven in Section 5.2.1.

The algorithm that motivated this study was infinitesimal gradient ascent [SKM00], which is an algorithm for repeated games. First, this result shows that infinitesimal gradient ascent is a no-regret behavior [FL98],

[1]This work previously appeared in [Zin03].

and secondly it shows that GIGA, a nontrivial extension developed here of infinitesimal gradient ascent to games with more than two actions, is a no-regret behavior. GIGA is defined in Section 5.3.2 and is proven to be a no-regret algorithm in Section 5.3.3.

Also, Bansal et al. [BBCM03] use the results of this chapter in the oblivious routing domain.

5.2 Online Convex Programming

Definition 5.1 *A set of vectors $S \subseteq \mathbf{R}^n$ is **convex** if for all $x, x' \in S$, and all $\lambda \in [0,1]$, $\lambda x + (1-\lambda)x' \in S$.*

Definition 5.2 *For a convex set F, a function $f : F \to \mathbf{R}$ is **convex** if for all $x, y \in F$, for all $\lambda \in [0,1]$,*

$$\lambda f(x) + (1-\lambda) f(y) \geq f(\lambda x + (1-\lambda) y).$$

If one were to imagine a convex function $\mathbf{R}^2 \to \mathbf{R}$, where the function described the altitude, then the function would look like a valley.

Definition 5.3 *A **convex programming problem** consists of a convex feasible set F and a convex cost function $c : F \to \mathbf{R}$. The **optimal solution** is the solution that minimizes the cost.*

An example of a convex programming problem is that of a farmer who knows the restrictions on labor and land before she begins, and also knows the demand for her goods in advance.

Suppose that the farmer is not aware of the demand for her products before she begins. She knows that the corresponding cost function is convex, but she is unaware of its actual values. After the year is over and she has sold her items, she then becomes aware of her profit, and can use this to plan the next year. This is an instance of an online convex programming problem.

Definition 5.4 *An **online convex programming problem** consists of a feasible set $F \subseteq \mathbf{R}^n$ and an infinite sequence $\{c^1, c^2, \ldots\}$ where each $c^t : F \to \mathbf{R}$ is a convex function.*

*At each time step t, an **online convex programming algorithm** selects a vector $x^t \in F$. After the vector is selected, it receives the cost function c^t.*

Because all information is not available before decisions are made, online algorithms do not reach "solutions", but instead achieve certain goals. See Section 5.2.1.

Define $\| x \| = \sqrt{x \cdot x}$ and $d(x, y) = \| x - y \|$. Throughout the remainder of the chapter we will make seven assumptions:

1. The feasible set F is **bounded**. There exists $N \in \mathbf{R}$ such that for all $x, y \in F$, $d(x, y) \leq N$.
2. The feasible set F is **closed**. For all sequences $\{x^1, x^2, \ldots\}$ where $x^t \in F$ for all t, if there exists a $x \in \mathbf{R}^n$ such that $x = \lim_{t \to \infty} x^t$, then $x \in F$.
3. The feasible set F is **nonempty**. There exists an $x \in F$.
4. The cost is **differentiable**. For all t, c^t is differentiable [2].
5. The cost derivatives are **bounded**. There exists an $N \in \mathbf{R}$ such that for all t, for all $x \in F$, $\| \nabla c^t(x) \| \leq N$.
6. The cost derivatives are **computable**. For all t, there exists an algorithm, given x, which produces $\nabla c^t(x)$.
7. The projection onto F is **computable**. For all $y \in \mathbf{R}^n$, there exists an algorithm which can produce $\operatorname{argmin}_{x \in F} d(x, y)$. We define the projection $P(y) = \operatorname{argmin}_{x \in F} d(x, y)$.

Given this machinery, we can describe our algorithm.

[2] Although we make the assumption that c^t is differentiable, the algorithm can also work if there exists an algorithm that, given x, can produce a vector g such that for all y, $g \cdot (y - x) \leq c^t(y) - c^t(x)$.

Algorithm 5.5 ***Greedy Projection*** *Select an arbitrary $x^1 \in F$ and a sequence of learning rates $\eta_1, \eta_2, \ldots \in \mathbf{R}^+$. In time step $t \geq 1$, after receiving a cost function c^t, select the next vector x^{t+1} according to:*

$$x^{t+1} = P\left(x^t - \eta_t \nabla c^t(x^t)\right).$$

The basic principle at work in this algorithm is quite clear if we consider the case where $F \subseteq \mathbf{R}^2$ and the sequence $\{c^1, c^2, \ldots\}$ consists of a single repeated cost function c. In this case, our algorithm is operating in a valley that does not change over time. The boundary of the feasible set is the edge of the valley. By proceeding along the direction opposite the gradient, we walk down into the valley. By projecting back into the convex set, we move along the edge of the valley.

5.2.1 Analyzing the Performance of the Algorithm

We measure the performance of an algorithm in comparison to the best algorithm in hindsight that knows all of the cost functions and selects one fixed vector.

Definition 5.6 *Given an algorithm A, and a convex programming problem $(F, \{c^1, c^2, \ldots\})$, if $\{x^1, x^2, \ldots\}$ are the vectors selected by A, then the* ***cost*** *of A until time T is*

$$C_A(T) = \sum_{t=1}^{T} c^t(x^t).$$

The cost of a static feasible solution $x \in F$ until time T is

$$C_x(T) = \sum_{t=1}^{T} c^t(x).$$

The regret of algorithm A until time T is

$$R_A(T) = C_A(T) - \min_{x \in F} C_x(T).$$

Our goal is to prove that when $n_t = t^{-1/2}$, the average regret $R_A(T)/T$ of Greedy Projection approaches zero. In order to state our results about bounding the regret of this algorithm, we need to specify some parameters. First, define:

$$\begin{aligned} \|F\| &= \max_{x,y \in F} d(x,y), \\ \|\nabla c\| &= \sup_{x \in F, t \in \{1,2,\ldots\}} \|\nabla c^t(x)\|. \end{aligned}$$

Here is the first result derived in this chapter:

Theorem 5.7 *If $\eta_t = t^{-1/2}$, the regret of the Greedy Projection algorithm (G) satisfies:*

$$R_G(T) \leq \frac{\|F\|^2 \sqrt{T}}{2} + \left(\sqrt{T} - \frac{1}{2}\right) \|\nabla c\|^2.$$

Therefore, $\limsup_{T \to \infty} R_G(T)/T \leq 0$.

The first part of the bound is because we might begin on the wrong side of F. The second part is because we always respond after we see the cost function.

Proof: First we show that without loss of generality, for all t there exists a $g^t \in \mathbf{R}^n$ such that for all x, using $g^t \cdot x$ in place of $c^t(x)$ does not change the behavior of G or decrease the regret of G.

First, begin with arbitrary $\{c^1, c^2, \ldots\}$, run the algorithm and compute $\{x^1, x^2, \ldots\}$. Then define $g^t = \nabla c^t(x^t)$. If we were to change c^t such that for all x, $c^t(x) = g^t \cdot x$, the behavior of the algorithm would be the same. Would the regret be the same?

Because c^t is convex, for all x:

$$c^t(x) \geq (\nabla c^t(x^t)) \cdot (x - x^t) + c^t(x^t).$$

Therefore, for all $x^* \in F$: $c^t(x^*) \geq g^t \cdot (x^* - x^t) + c^t(x^t)$. Thus:

$$\begin{aligned} c^t(x^t) - c^t(x^*) &\leq c^t(x^t) - \left(g^t \cdot (x^* - x^t) + c^t(x^t)\right) \\ &\leq g^t \cdot x^t - g^t \cdot x^*. \end{aligned}$$

Thus the regret would be at least as much with the modified sequence of functions.

We define for all t, $y^{t+1} = x^t - \eta_t g^t$. Observe that $x^{t+1} = P(y^{t+1})$. We will bound the regret of not playing action x^* on time step t.

$$\begin{aligned} y^{t+1} - x^* &= (x^t - x^*) - \eta_t g^t. \\ (y^{t+1} - x^*)^2 &= (x^t - x^*)^2 - 2\eta_t (x^t - x^*) \cdot g^t + \eta_t^2 \| g^t \|^2. \end{aligned} \tag{5.1}$$

Observe that in the expression of the form $a^2 - 2ab + b^2$ in Equation 5.1, a^2 is a potential, $2ab$ is the immediate cost, and b^2 is the error (within a factor of $2\eta_t$). We will now begin to fully flush out these properties. For all $y \in \mathbf{R}^n$, for all $x \in F$, $(y - x)^2 \geq (P(y) - x)^2$ [GW00]. Also, $\| g^t \| \leq \| \nabla c \|$. So:

$$\begin{aligned} (x^{t+1} - x^*)^2 &\leq (x^t - x^*)^2 - 2\eta_t (x^t - x^*) \cdot g^t + \eta_t^2 \| \nabla c \|^2. \\ (x^t - x^*) \cdot g^t &\leq \frac{1}{2\eta_t}\left((x^t - x^*)^2 - (x^{t+1} - x^*)^2\right) + \frac{\eta_t}{2} \| \nabla c \|^2. \end{aligned}$$

Now, by summing we get:

$$\begin{aligned} R_G(T) &= \sum_{t=1}^{T} (x^t - x^*) \cdot g^t \\ &\leq \sum_{t=1}^{T} \left(\frac{1}{2\eta_t}\left((x^t - x^*)^2 - (x^{t+1} - x^*)^2\right) + \frac{\eta_t}{2} \| \nabla c \|^2 \right) \\ &\leq \frac{1}{2\eta_1}(x^1 - x^*)^2 - \frac{1}{2\eta_T}(x^{T+1} - x^*)^2 + \frac{1}{2}\sum_{t=2}^{T}\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right)(x^t - x^*)^2 + \frac{\| \nabla c \|^2}{2}\sum_{t=1}^{T} \eta_t \end{aligned} \tag{5.2}$$

Observe that $-\frac{1}{2\eta_T}(x^{T+1} - x^*)^2$ is negative. Because $x^1, \ldots, x^T$ and x^* are in F, $(x^1 - x^*)^2 \leq \| F \|^2, \ldots, (x^T - x^*) \leq \| F \|^2$.

$$\begin{aligned} R_G(T) &\leq \| F \|^2 \left(\frac{1}{2\eta_1} + \frac{1}{2}\sum_{t=2}^{T}\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right)\right) + \frac{\| \nabla c \|^2}{2}\sum_{t=1}^{T} \eta_t \\ &\leq \| F \|^2 \frac{1}{2\eta_T} + \frac{\| \nabla c \|^2}{2}\sum_{t=1}^{T} \eta_t. \end{aligned} \tag{5.3}$$

Now, if we define $\eta_t = \frac{1}{\sqrt{t}}$, then:

$$\begin{aligned}
\sum_{t=1}^{T} \eta_t &= \sum_{t=1}^{T} \frac{1}{\sqrt{t}} \\
&\leq 1 + \int_{t=1}^{T} \frac{dt}{\sqrt{t}} \\
&= 1 + \left[2\sqrt{t}\right]_1^T \\
&= 2\sqrt{T} - 1.
\end{aligned}$$

Plugging this into Equation 5.3 yields the result. ■

5.2.2 Regret Against a Dynamic Strategy

Another possibility for the offline algorithm is to allow a small amount of change. For instance, imagine that the path that the offline algorithm follows is of limited size.

Definition 5.8 *The* ***path length*** *of a sequence $A' = \{x^1, \ldots, x^T\}$ is:*

$$\sum_{t=1}^{T-1} d(x^t, x^{t+1}).$$

The ***cost*** *of the sequence A' is:*

$$C_{A'}(T) = \sum_{t=1}^{T} c^t(x^t).$$

Define $\mathbb{A}(T, L)$ to be the set of sequences with T vectors and a path length less than or equal to L.

Definition 5.9 *Given an algorithm A and a maximum path length L, the* ***dynamic regret*** *$R_A(T, L)$ is:*

$$R_A(T, L) = C_A(T) - \min_{A' \in \mathbb{A}(T,L)} C_{A'}(T).$$

Theorem 5.10 *If η is fixed, the dynamic regret of the Greedy Projection algorithm satisfies:*

$$R_G(T, L) \leq \frac{7\|F\|^2}{4\eta} + \frac{L\|F\|}{\eta} + \frac{T\eta\|\nabla c\|^2}{2}.$$

Proof: Define z^t to be the optimal dynamic sequence at time t. As with Equation 5.2, we can argue that:

$$\begin{aligned}
R_G(T) &\leq \frac{1}{2}\sum_{t=1}^{T} \frac{1}{\eta}\left((x^t - z^t)^2 - (x^{t+1} - z^t)^2\right) + \frac{\|\nabla c\|^2}{2}\sum_{t=1}^{T} \eta \\
&= \frac{1}{2\eta}\sum_{t=1}^{T}\left((x^t)^2 - (x^{t+1})^2\right) + \frac{1}{2}\sum_{t=1}^{T}\frac{1}{\eta}2(x^{t+1} - x^t)\cdot(z^t) + \frac{T\eta\|\nabla c\|^2}{2} \\
&= \frac{1}{2\eta}\left((x^1)^2 - (x^{T+1})^2\right) + \frac{1}{\eta}x^{T+1}\cdot z^T - \frac{1}{\eta}x^1\cdot z^1 + \sum_{t=2}^{T}\frac{1}{\eta}(z^{t-1} - z^t)\cdot(x^t) + \frac{T\eta\|\nabla c\|^2}{2}.
\end{aligned}$$

Without loss of generality, assume $0 \in F$. Thus for any $a, b \in F$, it is the case that $\|a\| \leq \|F\|$, $\|b\| \leq \|F\|$, and $-\frac{\|F\|^2}{4} \leq a \cdot b \leq \|F\|^2$ ($a \cdot b$ is maximized when $a = b$ and $\|a\| = \|F\|$, and minimized when

$a = -b$ and $\|a\| = \|F\|/2$). So, $x^{T+1} \cdot z^T \le \|F\|^2$ and $-x^1 \cdot z^1 \le \frac{\|F\|^2}{4}$. Also, $(x^1)^2 - (x^{T+1})^2 \le \|F\|^2$, and since for any $a, b \in F$, $a \cdot b \le \|a\|\|b\|$, $(z^{t-1} - z^t) \cdot x^t \le d(z^{t-1}, z^t)\|F\|$. Therefore:

$$\begin{aligned} R_G(T) &\le \frac{7\|F\|^2}{4\eta} + \sum_{t=2}^{T} \frac{1}{\eta} d(z^{t-1}, z^t)\|F\| + \frac{T\eta\|\nabla c\|^2}{2} \\ &\le \frac{7\|F\|^2}{4\eta} + \frac{L\|F\|}{\eta} + \frac{T\eta\|\nabla c\|^2}{2}. \end{aligned}$$

■

Observe that, although the average regret appears constant in T, if one has a fixed T as a target and chooses $\eta = \frac{1}{\sqrt{T}}$, then the average regret approaches zero as the chosen target is a larger T.

5.2.3 Lazy Projection

This section defines a different algorithm that performs surprisingly well.

Algorithm 5.11 ***(Lazy Projection)*** *Select an arbitrary* $x^1 \in F$ *and a sequence of learning rates* $\eta_1, \eta_2, \ldots \in \mathbf{R}^+$. *Define* $y^1 = x^1$. *In time step* t*, after receiving a cost function, define* y^{t+1}*:*

$$y^{t+1} = y^t - \eta_t \nabla c^t(x^t), \tag{5.4}$$

and select the vector:

$$x^{t+1} = P(y^{t+1}). \tag{5.5}$$

Theorem 5.12 *Given a constant learning rate* η*, Lazy Projection's (L) regret is:*

$$R_L(T) \le \frac{\|F\|^2}{2\eta} + \frac{\eta\|\nabla c\|^2 T}{2}.$$

The proof is in Section 5.6. There are two ways of viewing this guarantee. The first is that the algorithm approaches a low (although not zero) average regret. The second is that, for a target number of steps T, one can choose $\eta = 1/\sqrt{T}$. Thus, by partitioning time into a number of periods of increasing length (i.e., each period is quadruple the size of the one before, and the learning rate is half), and resetting the algorithm at the beginning of each period, the average regret will drop to zero in the limit.

It is interesting to note that lazy projection with $\eta_t = t^{-1/2}$ will not always result in the average regret being low. For example, if one imagines $F = [0, 1]$, $x^1 = 1$, and $\|\nabla c\| = 1$. Then, for $T/3$ time steps, $c^t(x) = x$. For $2T/3$ time steps, $c^t(x) = -x$. The algorithm will always play 1, and have an average regret of $1/3$ after time T.

5.3 Generalized Infinitesimal Gradient Ascent

This section establishes that repeated games are online linear programming problems, and an application of our algorithm is a no-external-regret behavior.

5.3.1 Oblivious Deterministic Environments

A behavior[3] is an oblivious deterministic behavior if it plays the same sequence of actions regardless of the actions of the agent. We will use this type of behavior to bridge the gap between the results in online linear programming and repeated games.

Formally, a behavior $\sigma_2 : H \to \Delta(A_2)$ is an **oblivious deterministic behavior** if there exists a function $r : \{1, 2, \ldots\} \to A_2$ where for all $h \in H$:

$$\Pr_{a_2 \in \sigma_2(h)}[a_2 = r(|h| + 1)] = 1.$$

[3]For the basic definitions in repeated bimatrix games, see Section 1.7.1.

5.3.2 Formulating a Repeated Game as an Online Linear Program

For simplicity, suppose that we consider the case where $A_1 = \{1, \ldots, n\}$. Before each time step in a repeated game, we select a distribution over actions. This can be represented as a vector in a n-standard closed simplex, the set of all points $x \in \mathbf{R}^n$ such that for all i, $x_i \geq 0$, and $\sum_{i=1}^{n} x_i = 1$. Define this to be F.

Since we have a utility u_1 instead of cost c, we will perform gradient ascent instead of descent. The utility u_1 is a linear function when the second agent's action becomes known.

Algorithm 5.13 ***(Generalized Infinitesimal Gradient Ascent)*** *Choose a sequence of learning rates $\{\eta_1, \eta_2, \ldots\}$. Begin with an arbitrary vector $x^1 \in F$. Then for each time step t:*

1. *Play according to x^t: play action i with probability x_i^t.*
2. *Observe the action $h_{t,2}$ of the other agent and calculate:*

$$\begin{aligned} y_i^{t+1} &= x_i^t + \eta_t u_1(i, h_{t,2}), \\ x^{t+1} &= P(y^{t+1}). \end{aligned}$$

where $P(y) = \operatorname{argmin}_{x \in F} d(x, y)$, as before.

In this online convex programming problem, $\| F \| \leq \sqrt{2}$, and $\| \nabla c \| \leq \sqrt{|A_1|} \| u_1 \|$. For any $h \in H$, define $R^{ext,a}(h)$ to be the regret in comparison to using the action $a \in A_1$. Formally:

$$R^{ext,a}(h) = \frac{1}{|h|} \sum_{t=1}^{|h|} u_1(a, h_{t,2}) - u_1(h_t).$$

Now, we can apply Theorem 5.7 to get the following result:

Theorem 5.14 *If $\eta_t = t^{-1/2}$, the expected average regret of GIGA for all oblivious deterministic behaviors $\sigma_2 : H \to \Delta(A_2)$, for all $a_1 \in A_1$, for all T is:*

$$\mathbf{E}_{h \in \mu_{\sigma_1, \sigma_2}}[R^{ext,a_1}(h(T))] \leq \frac{1}{\sqrt{T}} + \left(\frac{1}{\sqrt{T}} - \frac{1}{2T} \right) |A_1| |u_1|^2.$$

5.3.3 Self-Oblivious Behavior

It is important to observe that the above is more a method of constructing a behavior than an actual behavior of the form $\sigma_1 : H \to \Delta(A_1)$. By proper simulation, the above can be reformulated into a function from histories to distributions over actions.

Before we begin, we fix x^1. Then, whenever we see a history $h \in H^t$, we simulate how we would have constructed $x^2, \ldots, x^t$ based on the actions of the second agent. Thus, given x^1, the above can be reformulated into $\sigma_1 : H \to \Delta(A_1)$ for any game.

Moreover, σ_1 here is **self-oblivious**, in that it only depends on the history of actions of the second agent. Define $A_2^* = \bigcup_{i=0}^{\infty} A_2^i$, and $\Pi_2 : H \to A_2^*$ such that for all $h \in H$,

$$\Pi_2(h) = \{h_{1,2}, h_{2,2}, h_{3,2}, \ldots, h_{|h|,2}\}$$

Definition 5.15 *A behavior σ_1 is* ***self-oblivious*** *if there exists a function $f : A_2^* \to \Delta(A_1)$ such that for all $h \in H$, $\sigma_1(h) = f(\Pi_2(h))$.*

Self-oblivious algorithms are robust (in the way we formally define in Lemma 5.16) against adaptive adversaries, those that change their technique based on past actions of the behavior.

GIGA is self-oblivious, in that the strategy in the current time step can be calculated given x^1 (a constant) and the past actions of the second agent. It should be noted that not all algorithms are self-oblivious. For instance, Kalai and Vempala [KV02] describe an algorithm that is not self-oblivious, because it uses a "random seed" at the beginning that an adaptive adversary could learn over time and then use in some settings.

The following lemma compartmentalizes the technique used at the end of [FS99].

Lemma 5.16 *Given a self-oblivious behavior σ_1, if for every $\epsilon > 0$ there exists a T such that, for all deterministic, oblivious behaviors $\sigma_2 : H \to \Delta(A_2)$, for all $a_1 \in A_1$, for all $t > T$:*

$$\mathbf{E}_{h \in \mu_{\sigma_1,\sigma_2}} \left[R^{ext,a_1}(h(t)) \right] < \epsilon,$$

then σ_1 is a no-external-regret behavior. Therefore, GIGA is a no-external-regret behavior.

The proof is in Section 5.7.

5.3.4 Lazy Projection and Fictitious Play

There have been some techniques presented to smooth fictitious play in [FL95], and here we present a very simple version that has much of the "spirit" of fictitious play.

Algorithm 5.17 ***(Z Fictitious Play)*** *Choose $\eta > 0$. At time step t define:*

$$y_i^t = \sum_{j=1}^{t-1} \eta u_1(i, h_{j,2}).$$

In other words, y_i^t/η is the total reward one would have received if one had played action i over the entire history. Define:

$$x^t = P(y^t) = \operatorname*{argmin}_{x \in \Delta(A_1)} d(x, y^t).$$

Play according to x^t in time step t.

This algorithm is an instance of Lazy Projection. Whereas cautious fictitious play [FL95] will play a strategy that is *almost* what fictitious play would do *all* of the time, Z Fictitious Play plays *exactly* what fictitious play would do *most* of the time, except on time steps where the past utility of some of the best actions in A_1 are sufficiently close in value to make Z Fictitious Play randomize.

5.4 Converting Old Algorithms

In this section, in order to compare our work with that of others, we show how one can naïvely translate algorithms for mixing experts into algorithms for online linear programs, and online linear programming algorithms into algorithms for online convex programs. This section is a discussion and no formal proofs are given.

5.4.1 Formal Definitions

We begin with defining the experts problem.

Definition 5.18 *An **experts problem** is a set of experts $E = \{e_1, \ldots, e_n\}$ and a sequence of cost vectors $c^1, c^2, \ldots$ where for all i, $c^i \in \mathbf{R}^n$.*

*On each time step t, an **expert algorithm (EA)** first selects a distribution $D^t \in \Delta(E)$, and then observes a cost vector c^t.*

We assume that the EA can handle both positive and negative costs for the experts. If not, the EA can be easily extended by shifting the values into the positive range.

Now, we define an abstract online linear programming problem.

Definition 5.19 *An **online linear programming problem** is a closed convex polytope $F \subseteq \mathbf{R}^n$ and a sequence of cost vectors $c^1, c^2, \ldots$ where for all i,$c^i \in \mathbf{R}^n$.*

*On each time step t, an **online linear programming algorithm (OLPA)** first plays a distribution $D^t \in \Delta(F)$, and then observes a cost vector c^t.*

An OLPA can be constructed from an EA, as described below.

Algorithm 5.20 *Define $v^1, \ldots, v^k$ to be the vertices of the polytope for an online linear program. Choose $E = \{e_1, \ldots, e_k\}$ to be the experts, one for each vertex.*

On each time step t, receive a distribution D^t from the EA, and select vector v^i if expert e_i is selected. Define $c' \in \mathbf{R}^k$ such that $c'_i = c^t \cdot v^i$. Send EA the cost vector $c' \in \mathbf{R}^k$.

The optimal static vector must be a vertex of the polytope, because a linear program always has a solution at a vertex of the polytope. If the original EA can do almost as well as the best expert, this OLPA can do at least as well as the best static vector.

Note that both EAs and Greedy Projection can be used to solve OLPA. The second observation is that most EA have bounds that depend on the number of experts. The number of vertices can be exponential with respect to the number of constraints, and arbitrarily high even for a fixed diameter. Therefore, any normal expert's bound is incomparable to our bound on the regret of Greedy Projection.

There are some EAs that begin with a distribution or uneven weighting over the experts. These EAs may be able to obtain guarantees based on the diameter of the feasible region in this scenario, because a distribution where a cluster of vertices close together (which will all have similar utilities) has the same total starting weight as a lonely vertex with no close neighbors may result in a having a lot of weight on vertices "close" to the optimal vertex in a short period of time.

5.4.2 Converting an OLPA to an Online Convex Programming Algorithm

There are two reasons that Algorithm 5.20 will not work for an online convex program. The first is that an online convex program can have an arbitrary convex shape as a feasible region, such as a circle, which cannot be described as the convex hull of any finite number of points.

The second reason is that a convex function may not have a minimum on the edge of the feasible set. For instance, if $F = \{x : x \cdot x \leq 1\}$ and $c(x) = x \cdot x$, the minimum is in the center of the feasible set.

Now, this first issue is difficult to handle directly[4], so we will simply assume that the OLPA can handle the feasible region of the online convex programming problem. This can be either because the OLPA can handle an arbitrary convex region as in [KV02], or because the convex region of the convex programming problem is a convex polytope.

We handle the second issue by converting the cost function to a linear one. In Theorem 5.7, we find that the worst case is when the cost function is linear. This assumption depends on two properties of the algorithm; the algorithm is deterministic, and the only property of the cost function c^t that is observed is $\nabla c^t(x^t)$.

Now, we form an Online Convex Programming algorithm.

Algorithm 5.21 ***(Exact)*** *On each time step t, receive D^t from the OLPA, and play $x^t = \mathbf{E}_{X \in D^t}[X]$. Send the OLPA the cost vector $\nabla c^t(x^t)$.*

The algorithm is discrete and only observes the gradient at the point x^t, thus we can assume that the cost function is linear. If the cost function is linear, then:

$$\mathbf{E}_{X \in D^t}[c^t(X)] = c^t(\mathbf{E}_{X \in D^t}[X]).$$

x^t may be difficult to compute, so instead of explicitly calculating x^t, we can sample.

[4] One can approximate a convex region by a series of increasingly complex convex polytopes, but this solution is undesirable because the number of experts is proportional to the complexity of the polytope.

Algorithm 5.22 ***(Approx)*** *Select the number of samples $s_1, s_2, \ldots$ to be taken each time step. On each time step t, sample $X_1, \ldots, X_{s_t}$ independently from the distribution D^t. Play:*

$$z^t = \frac{1}{s_t} \sum_{i=1}^{s_t} X_i.$$

Send OLPA the cost vector $\nabla c^t(z^t)$.

We will bound the worst-case difference between Approx and Exact. There are several difficulties in doing so. The first is that Approx is a randomized algorithm, so although it only eventually takes the derivative at a single point, it has a probability of taking the derivative at one of many points, and so both the value and the derivative matter at those points. Secondly, the long-term strategy of the adversary, how it forces the algorithm to move in certain directions, might depend on the random result. Therefore, we try to separate the random element from the deterministic element in a new game. At time step t:

1. Receive D^t from the OLPA.
2. Sample $X_1, \ldots, X_{s_t}$ independently from distribution D^t.
3. Calculate and reveal to the adversary $z^t = \frac{1}{s_t} \sum_{i=1}^{s_t} X_i$.
4. Calculate and reveal to the adversary $x^t = \mathbf{E}_{X \in D^t}[X]$.
5. The adversary selects $g^t, h^t \in \mathbf{R}^n$ where $\| g^t \|, \| h^t \| \leq \| \nabla c \|$.
6. Send the OLPA g^t.

This game updates the OLPA in the same fashion as the Exact algorithm. We define the super regret S as:

$$S(T) = \sum_{t=1}^{T} h^t \cdot (z^t - x^t) + \sum_{t=1}^{T} g^t \cdot x^t - \min_{x^* \in F} \sum_{t=1}^{T} g^t \cdot x^*.$$

We can relate this super regret to the regret of both the Exact and Approx algorithms.

The second half of the super regret is the regret of the Exact algorithm on a sequence of linear functions.

$$\mathbf{E}[S(T)] = \mathbf{E}[\sum_{t=1}^{T} h^t \cdot (z^t - x^t)] + R_{Exact}(T).$$

We can bound $\mathbf{E}[h^t \cdot (z^t - x^t)]$ based on the number of samples taken:

$$\begin{aligned} \mathbf{E}[h^t \cdot (z^t - x^t)] &\leq \mathbf{E}[\| \nabla c \| d(z^t, x^t)] \\ &= \| \nabla c \| \mathbf{E}[d(z^t, x^t)]. \end{aligned}$$

Without loss of generality, we assume that $x^t = 0$, $0 \in F$.

$$\mathbf{E}[d(0, z^t)] = \frac{1}{s_t} \mathbf{E}\left[\sqrt{\sum_{i,j} X_i X_j} \right].$$

For a random variable Y:

$$\mathbf{E}[Y] \leq \sqrt{\mathbf{E}[Y^2]}.$$

We use this fact to prove that:

$$\begin{aligned}
\mathbf{E}[d(0,z^t)] &\leq \frac{1}{s_t}\sqrt{\mathbf{E}\left[\sum_{i,j} X_i \cdot X_j\right]} \\
&= \frac{1}{s_t}\sqrt{\sum_i \mathbf{E}\left[\| X_i \|^2\right]} \\
&\leq \frac{1}{s_t}\sqrt{s_t \| F \|^2} \\
&= \frac{\| F \|}{\sqrt{s_t}}
\end{aligned}$$

$$\mathbf{E}[S(T)] \leq \| \nabla c \| \| F \| \sum_{t=1}^{T} \frac{1}{\sqrt{s_t}} + R_{Exact}(T)$$

By choosing $s_t = t$, $\sum_{t=1}^{T} \frac{1}{\sqrt{s_t}} \leq 2\sqrt{T} - 1$. Thus, by selecting s_t properly:

$$\mathbf{E}[S(T)] = \| \nabla c \| \| F \| (2\sqrt{T} - 1) + R_{Exact}(T)$$

We will now prove that $S(T) \geq R_{Approx}(T)$. Imagine that in the Approx algorithm, each time step the adversary knew in advance the random selection z^t before selecting the cost function c^t. This only increases the regret. In the new game, the adversary selects $g^t = \nabla c^t(z^t)$. Therefore:

$$\begin{aligned}
c^t(z^t) - c^t(x^*) &\leq g^t \cdot (z^t - x^*) \\
&\leq g^t \cdot (z^t - x^t + x^t - x^*) \\
&\leq g^t \cdot (z^t - x^t) + g^t \cdot (x^t - x^*)
\end{aligned}$$

The adversary selects h^t to be a vector of length $\| \nabla c \|$ in the direction of $z^t - x^t$.

$$\begin{aligned}
h^t \cdot (z^t - x^t) &= \| \nabla c \| d(x^t, z^t) \\
&\geq \| g^t \| d(x^t, z^t) \\
&\geq g^t \cdot (z^t - x^t)
\end{aligned}$$

So, finally:

$$\begin{aligned}
c^t(z^t) - c^t(x^*) &\leq h^t \cdot (z^t - x^t) + g^t \cdot x^t - g^t \cdot x^* \\
\sum_{t=1}^{T} \left(c^t(z^t) - c^t(x^*)\right) &\leq \sum_{t=1}^{T} h^t \cdot (z^t - x^t) + \sum_{t=1}^{T} g^t \cdot x^t - \min_{x^* \in F} \sum_{t=1}^{T} g^t \cdot x^* \\
R_{Approx}(T) &\leq S(T)
\end{aligned}$$

Therefore, for the proper number of samples:

$$E[R_{Approx}(T)] \leq R_{Exact}(T) + \| \nabla c \| \| F \| (2\sqrt{T} - 1)$$

5.5 Related Work

Kalai and Vempala [KV02] have developed algorithms to solve online linear programming, which is a specific type of online convex programming. They were attempting to make the algorithm behave in a lazy fashion, changing its vector slowly, whereas here we are attempting to be more dynamic, as is highlighted in Sections 5.2.2 and 5.3.3.

These algorithms were motivated by the algorithm of [SKM00] which applies gradient ascent to repeated games. We extend their algorithm to games with an arbitrary number of actions, and prove that it is a no-external-regret algorithm. There has been extensive work on regret in repeated games and in the experts domain, such as [Bla56, FV99, Fos99, FS99, FL95, FL99, Han57, HMC00, HMC01a, LW89]. What makes this work noteworthy in a very old field is that it proves that a widely-used technique in artificial intelligence, gradient ascent, has a property that is of interest to those in game theory. As stated in Section 5.4, experts algorithms can be used to solve online linear programs and online convex programming problems, but the bounds may become significantly worse. For example, if the feasible region is a regular polygon of diameter 1, then Greedy projection will do well regardless of the size of the polygon, but the experts algorithm bounds will decay as the number of vertices increase.

There are several studies of online gradient descent and related update functions, for example [CBLW94, KW97, HW01, KW01]. These studies focus on prediction problems where the loss functions are convex Bregman divergences. In this chapter, we are considering arbitrary convex functions, in problems that may or may not involve prediction.

In the offline case, [DDL99] have done work on proving that gradient descent and projection for arbitrary Bregman distances converges to the optimal result.

Two of the most notable extensions to the problem in this chapter involve settings where the complete convex function is not observed, only the cost of the algorithm. Particularly, [BM04] show that one can handle arbitrary linear cost functions against an adaptive adversary, and [FKM04] show that one can handle arbitrary convex cost functions against an oblivious adversary.

5.6 Proof of Lazy Projection*

In analyzing Greedy Projection, we had one potential: the distance from the optimal point. In analyzing Lazy Projection, we have two potentials: the distance from y^t to the optimal point (ideal potential), and the distance to the set F (projection potential). The ideal potential is "good", in the sense that we grow distant from the optimal point when we are performing better than the optimal point. However, the projection potential is "bad", in the sense that the farther we go from F, the more difference there is between the vector we wanted to use (y^t) and the vector we did use (x^t). What makes Lazy Projection work is that these two potentials cancel each other.

For all $y \in \mathbf{R}^n$, for all closed, convex sets $F \subseteq \mathbf{R}^n$, define $d(y, F) = \min_{x \in F} d(y, x)$.

The following two lemmas are the subcomponents of our general proof. We prove them in Section 5.6.3.

Lemma 5.23 ***(ideal potential)*** *For any convex set F and any linear cost sequence c^t, defining y^t as in Equation 5.4 with a fixed learning rate η, and any $x^* \in F$:*

$$\sum_{t=1}^{T} \left(c^t(y^t) - c^t(x^*)\right) \leq \frac{\|F\|^2}{2} - \frac{d(y^{T+1}, x^*)^2}{2\eta} + \frac{T\eta \|\nabla c\|^2}{2}$$

Lemma 5.24 ***(projection potential)*** *For any convex set F and any linear cost sequence c^t, defining y^t and x^t as in Equations 5.4 and 5.5:*

$$\sum_{t=1}^{T} \left(c^t(x^t) - c^t(y^t)\right) \leq \frac{d(y^{T+1}, F)^2}{2\eta}$$

In Section 5.6.1, we present an example highlighting the issues of Lemma 5.24. In Section 5.6.2, we prove the critical connection between the n-dimensional case and the one-dimensional case. In Section 5.6.3, we complete the proofs of Lemma 5.23, Lemma 5.24, and Theorem 5.12.

5.6.1 A Motivating Example

In order to motivate the following technical lemmas, consider an example. Choose $F = \{x \in \mathbf{R} : x \leq a\}$. If $y^t \geq a$, $x^t = P(y^t) = a$. Assume that $\eta = 1$ and the cost function at time t is $c^t(x) = g^t \cdot x$. We will attempt to bound the following difference D:

$$D = \sum_{t=1}^{T} c^t(P(y^t)) - c^t(y^t).$$

D is how much less cost we would have accrued had we played the point y^t instead of x^t. Assume that for all t, $y^t > a$. So,

$$D = \sum_{t=1}^{T} \left(c^t(a) - c^t(y^t)\right).$$

Since $g^t = y^{t+1} - y^t$ (a result of the assumptions made at the beginning of the section applied to Equation 5.4) , then:

$$D = \sum_{t=1}^{T} (y^{t+1} - y^t) \cdot (y^t - a).$$

Define $z^t = d(y^t, F) = \min_{x \in F} d(y^t, x) = y^t - a$. The equation becomes:

$$D = \sum_{t=1}^{T} (z^{t+1} - z^t) z^t. \tag{5.6}$$

We can use the following lemma here:

Lemma 5.25 *For all $a, b \in \mathbf{R}$:*

$$(a - b)b \leq \frac{a^2 - b^2}{2}.$$

Proof: This is an algebraic manipulation of $0 \leq (a - b)^2$. ■

Thus, we can form a potential:

$$\begin{aligned} D &\leq \sum_{t=1}^{T} \frac{(z^{t+1})^2 - (z^t)^2}{2} \\ D &\leq \frac{(z^{T+1})^2 - (z^1)^2}{2} \\ D &\leq \frac{(z^{T+1})^2}{2}. \end{aligned}$$

The key step that varies from the general case is that in general $(y^{t+1} - y^t) \cdot (y^t - P(y)) \neq (z^{t+1} - z^t)(z^t)$, so Equation 5.6 does not hold. However, we prove in the next section that the dot product is always less.

5.6.2 Geometric Lemmas

This section contains the technical details of the proof.

Lemma 5.26 *Given a convex set $F \subseteq \mathbf{R}^n$, if $y \in \mathbf{R}^n$ and $x \in F$, then $(y - P(y)) \cdot (x - P(y)) \leq 0$. In other words, the angle between y,$P(y)$, and x is not acute (see Figure 5.1).*

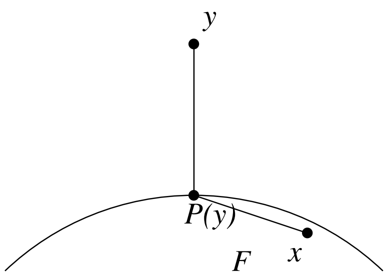

Figure 5.1: Lemma 5.26

Proof: We will prove the contrapositive of the theorem. Consider a point $x' \in F$ such that $(y-x')\cdot(x-x') > 0$. We will prove $x' \neq P(y)$. For all $\lambda \in [0,1]$, define:

$$z(\lambda) = (1-\lambda)x' + (\lambda)x = x' + \lambda(x-x').$$

We will prove that for small positive values of λ, y is closer to $z(\lambda)$ than it is to x'. Since F is convex, $z(\lambda)$ is in F. Now,

$$\begin{aligned}(y-z(\lambda))^2 &= (y-x'-\lambda(x-x'))^2 \\ &= \lambda^2(x-x')^2 - 2\lambda(y-x')\cdot(x-x') + (y-x')^2.\end{aligned}$$

Observe that for $0 < \lambda < \frac{2(y-x')\cdot(x-x')}{(x-x')^2}$, $(y-z(\lambda))^2 < (y-x')^2$. Thus, $P(y) \neq x'$. ■

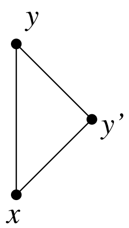

Figure 5.2: Lemma 5.27

Lemma 5.27 *Given $y, x, y' \in \mathbf{R}^n$, then (see Figure 5.2)*

$$(y-x)\cdot(y'-y) \leq d(y,x)(d(y',x) - d(y,x)).$$

Proof: We begin with a simple example. Suppose that $x, y, y' \in \mathbf{R}^2$, $x = (0,0)$, and $y = (1,0)$. We can prove that the property holds for these three vectors.

$$\begin{aligned}(y-x)\cdot(y'-y) &= (y'_1 - 1), \\ d(y,x)(d(y',x) - d(y,x)) &= \sqrt{(y'_1)^2 + (y'_2)^2} - 1 \\ &\geq (y'_1) - 1.\end{aligned}$$

Now, suppose that for a specific y, x, y', we have proven that the property holds. Then, it will hold for $y+a$, $x+a$, $y'+a$ (a translation), because:

$$\begin{aligned}((y+a)-(x+a))\cdot((y'+a)-(y+a)) &= (y-x)\cdot(y'-y) \\ &\leq d(y,x)(d(y',x) - d(y,x)) \\ &= d(y+a,x+a)(d(y'+a,x+a) - d(y+a,x+a)).\end{aligned}$$

Also, if it holds for y,x,y', then it will hold for ky, kx, and ky' (a scaling), because:

$$\begin{aligned}(ky-kx)\cdot(ky'-ky) &= k^2(y-x)\cdot(y'-y)\\ &\leq k^2 d(y,x)(d(y',x)-d(y,x))\\ &= d(ky,kx)(d(ky',kx)-d(ky,kx)).\end{aligned}$$

Also, the property is invariant under a rotation. Define A^T to be the transpose of the matrix A, and a^T to be the transpose of the vector a. Suppose that R is an orthonormal matrix (where $R^T R = I$). Now, for all $a, b \in \mathbf{R}^n$:

$$(Ra)\cdot(Rb) = (Ra)^T Rb = a^T R^T Rb = a^T b = a \cdot b,$$

$$d(Ra, Rb) = \sqrt{(R(a-b))\cdot(R(a-b))} = \sqrt{(a-b)\cdot(a-b)} = d(a,b).$$

We can now prove that if the property holds for y,x,y', then it will hold for Ry, Rx, Ry'.

$$\begin{aligned}(Ry-Rx)\cdot(Ry'-Ry) &= (R(y-x))\cdot(R(y'-y))\\ &= (y-x)\cdot(y'-y)\\ &\leq d(y,x)(d(y',x)-d(y,x))\\ &= d(Ry,Rx)(d(Ry',Rx)-d(Ry,Rx)).\end{aligned}$$

Observe that we can think of $\mathbf{R}^2$ as embedded in $\mathbf{R}^n$ without changing distance or dot product. Any three vectors y, x, y' can be obtained from $(0,1,0,\ldots,0), (0,0,0,\ldots,0), (y'_1, y'_2, 0, \ldots, 0)$ using translation, scaling, and rotation. So, the property holds for all vectors y, x, y'. ■

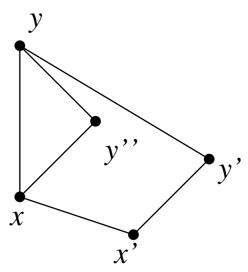

Figure 5.3: Lemma 5.28

Lemma 5.28 *Given $y, x, x', y' \in \mathbf{R}^n$, where $(y-x)\cdot(x'-x) \leq 0$(i.e., the angle yxx' is not acute), then (see Figure 5.3):*

$$(y-x)\cdot(y'-y) \leq d(y,x)(d(y',x')-d(y,x)).$$

Corollary 5.29 *Given $y, y' \in \mathbf{R}^n$, if $z = d(y, P(y))$ and $z' = d(y', P(y'))$:*

$$(y-P(y))\cdot(y'-y) \leq z(z'-z) \leq \frac{(z')^2 - z^2}{2}.$$

Proof: If $y = x$, then the result is trivial. Thus, assume $y \neq x$. We begin with a simple case, as before. We assume $y, x, x', y' \in \mathbf{R}^3$, $y = (1,0,0)$, $x = (0,0,0)$, and $x'_3 = 0$.

Throughout the next part of the proof, we prove that the worst case occurs when $x = x'$. We do this by defining $y'' = y' - x' + x$, and replacing y' with y'' and x' with x. Observe first that $d(y'', x) = d(y', x')$.

Observe that $(y-x)\cdot(y'-y) = y'_1 - 1$, and $(y-x)\cdot(y''-y) = y'_1 - x'_1 - 1$. Since $(y-x)\cdot(x'-x) = x'_1$, $x'_1 \leq 0$. Thus, $(y-x)\cdot(y'-y) \leq (y-x)\cdot(y''-y)$, so the relationship only gets tighter as we force $x = x'$. Thus, the property holds for these vectors by Lemma 5.27.

As in Lemma 5.27, we can prove that the property is invariant under a transformation, rotation, or scaling.

■

5.6.3 Completing the Proof

Now, we complete the proofs. The first part is similar to Theorem 5.7, and the second is a generalization of the argument in Section 5.6.1.

Proof (of Lemma 5.23, ideal potential): First, define A (the quantity we are trying to bound):

$$A = \sum_{i=1}^{T} \left(c^t(y^t) - c^t(x^*)\right).$$

Without loss of generality, for all t there exists a g^t such that $c^t(x) = g^t \cdot x$.

$$A = \sum_{t=1}^{T} g^t \cdot (y^t - x^*).$$

By definition, $y^{t+1} = y^t - \eta g^t$. Similar to Theorem 5.7:

$$\begin{aligned}
y^{t+1} - x^* &= y^t - x^* - \eta g^t \\
(y^{t+1} - x^*)^2 &= (y^t - x^* - \eta g^t)^2 \\
g^t \cdot (y^t - x^*) &= \frac{(y^t - x^*)^2 - (y^{t+1} - x^*)^2}{2\eta} + \frac{\eta}{2}\| g^t \|^2 \\
g^t \cdot (y^t - x^*) &\leq \frac{(y^t - x^*)^2 - (y^{t+1} - x^*)^2}{2\eta} + \frac{\eta \| \nabla c \|^2}{2}.
\end{aligned}$$

Summing, we get:

$$\begin{aligned}
A &\leq \sum_{t=1}^{T} \left(\frac{(y^t - x^*)^2 - (y^{t+1} - x^*)^2}{2\eta} + \frac{\eta \| \nabla c \|^2}{2} \right) \\
A &\leq \frac{(y^1 - x^*)^2 - (y^{T+1} - x^*)^2}{2\eta} + \frac{T\eta \| \nabla c \|^2}{2}.
\end{aligned}$$

Since $y^1, x^* \in F$:

$$A \leq \frac{\| F \|^2}{2\eta} - \frac{d(y^{T+1}, F)^2}{2\eta} + \frac{T\eta \| \nabla c \|^2}{2}.$$

■

Proof (of Lemma 5.24, projection potential): First, define B (the quantity we are trying to bound):

$$\begin{aligned}
B &= \sum_{t=1}^{T} \left(c^t(x^t) - c^t(y^t)\right) \\
&= \sum_{t=1}^{T} \left(c^t(P(y^t)) - c^t(y^t)\right).
\end{aligned}$$

Without loss of generality, for all t there exists a g^t such that $c^t(x) = g^t \cdot x$.

$$B = \sum_{t=1}^{T} g^t \cdot (P(y^t) - y^t).$$

Also, $g^t = \frac{y^t - y^{t+1}}{\eta}$. Thus:

$$\begin{aligned} B &= \frac{1}{\eta}\sum_{t=1}^{T}(y^t - y^{t+1})\cdot(P(y^t) - y^t) \\ &= \frac{1}{\eta}\sum_{t=1}^{T}(y^t - P(y^t))\cdot(y^{t+1} - y^t). \end{aligned}$$

Using Corollary 5.29:

$$\begin{aligned} B &\leq \frac{1}{\eta}\sum_{t=1}^{T}\frac{d(y^{t+1},F)^2 - d(y^t,F)^2}{2} \\ &\leq \frac{d(y^{T+1},F)^2}{2\eta}. \end{aligned}$$

■

Proof (of Theorem 5.12): Since lazy projection is a deterministic algorithm, and it only considers $\nabla c^t(x^t)$, then the worst case is a linear function. Therefore, we only need to consider linear functions.

$$\begin{aligned} R_G(T) &= \sum_{t=1}^{T}\left(c^t(x^t) - c^t(x^*)\right) \\ &= \sum_{t=1}^{T}(c^t(x^t) - c^t(y^t)) + \sum_{t=1}^{T}(c^t(y^t) - c^t(x^*)). \end{aligned}$$

Thus, by Lemma 5.23 and Lemma 5.24:

$$\begin{aligned} R_G(T) &\leq \frac{\|F\|^2}{2\eta} - \frac{d(y^{T+1},F)^2}{2\eta} + \frac{T\eta\|\nabla c\|^2}{2} + \frac{d(y^{T+1},F)^2}{2\eta} \\ &\leq \frac{\|F\|^2}{2\eta} + \frac{T\eta\|\nabla c\|^2}{2}. \end{aligned}$$

■

5.7 Proof of No-Regret*

Our analysis of regret first fixes a behavior σ_1, a time step T, an $\epsilon > 0$ and an action $a_1 \in A_1$. Then, we attempt to develop a behavior σ_2 for the second agent that maximizes the value:

$$\Pr_{h \in P^T_{\sigma_1,\sigma_2}}[R^{ext,a_1}(h) > \epsilon].$$

We will use Doob's Decomposition from the theory of stochastic processes to divide the regret into an "expected" part and a "random" part. We will then bound the expected part of the regret to be lower than the worst case oblivious deterministic second agent, due to the fact that the behavior we are studying is self-oblivious. Then, we will bound the random part with an old result from the theory of martingales.

We first introduce some concepts from stochastic processes that will be useful later in the proof.

Definition 5.30 *A **martingale difference sequence** is a sequence of variables $X_1, X_2, \ldots$ such that for all k, for all sequences $X_1, \ldots, X_k$:*

$$\mathbf{E}[X_{k+1}|X_1, \ldots, X_k] = 0$$

Lemma 5.31 ***(Azuma's Lemma)*** *[Hoe63]: If $X_1, X_2, \ldots$ is a martingale difference sequence, and for all i, $|X_i| \leq b$, then:*

$$\Pr\left[\sum_{i=1}^{k} X_i > a\right] \leq exp\left(-\frac{a^2}{2b^2k}\right).$$

Now, Doob's Decomposition allows us to construct a martingale difference sequence out of an arbitrary random sequence $Z_1, Z_2, \ldots$ by:

$$Y_i = Z_i - \mathbf{E}[Z_i|Z_{i-1}, \ldots, Z_1]$$

Corollary 5.32 *If $Z_1, Z_2, \ldots$ is a sequence of random variables and $|Z_i| \leq b$, then:*

$$\Pr\left[\sum_{i=1}^{k} Z_i - \mathbf{E}[Z_i|Z_{i-1}, \ldots, Z_1] > a\right] \leq exp\left(-\frac{a^2}{8b^2k}\right).$$

Define $R^{ext,a_1} : A_1 \times A_2 \to \mathbf{R}$ such that $R^{ext,a_1}(a_1', a_2) = u_1(a_1, a_2) - u_1(a_1', a_2)$. In this spirit of Doob's Decomposition, we define the functions $V_{\sigma_1} : H \to \mathbf{R}$ and $V_{\sigma_1}^{rem} : H \to \mathbf{R}$:

$$\begin{aligned} V_{\sigma_1}(h) &= \frac{1}{|h|}\sum_{i=1}^{|h|} \mathbf{E}_{a_1' \in \sigma_1(h(i-1))}[R^{ext,a_1}(a_1', h_{i,2})] \\ V_{\sigma_1}^{rem}(h) &= R^{ext,a_1}(h) - V_{\sigma_1}(h). \end{aligned}$$

$V_{\sigma_1}(h)$ represents the regret we expected on each time step given what happened in the past. $V_{\sigma_1}^{rem}(h)$ represents the deviation from that expected regret.

Lemma 5.33 *For any self-oblivious σ_1, for any T, $a_1 \in A_1$, there exists an oblivious, deterministic behavior σ_2^* for the second agent such that, for all $h \in H^T$:*

$$V_{\sigma_1}(h) \leq E_{h \in P^T_{\sigma_1, \sigma_2^*}}[R^{ext,a_1}(h)].$$

Proof: Observe that H^T is a finite set. Define $h^* \in H^T$:

$$h^* = \operatorname*{argmax}_{h \in H^T} V_{\sigma_1}(h)$$

Now, choose some $a_2' \in A_2$, and define σ_2^* such that:

$$\begin{aligned} \Pr_{a_2 \in \sigma_2^*(h)}[a_2 = h^*_{|h|+1,2}] &= 1 \text{ if } |h| < T \\ \Pr_{a_2 \in \sigma_2^*(h)}[a_2 = a_2'] &= 1 \text{ if } |h| \geq T. \end{aligned}$$

By construction, σ_2^* is a deterministic, oblivious behavior. σ_2^* will play the actions in $\Pi_2(h^*)$ for the first T time steps. Observe, that **since σ_1 is self-oblivious, the distribution of the actions of σ_1 at time step i is the same when σ_2^* plays the actions in $\Pi_2(h^*)$, as when the past history is $h^*(i-1)$.** Therefore:

$$\frac{1}{T}\sum_{i=1}^{T} \mathbf{E}_{a_1' \in \sigma_1(h^*(i-1))}[R^{ext,a_1}(a_1', h^*_{i,2})] = \mathbf{E}_{h \in P^T_{\sigma_1, \sigma_2^*}}[R^{ext,a_1}(h)].$$

■

We can use this fact to bound R^{ext,a_1} with high probability. Define:

$$|u_1| = \max_{(a_1,a_2) \in A_1 \times A_2} u_1(a_1, a_2) - \min_{(a_1,a_2) \in A_1 \times A_2} u_1(a_1, a_2).$$

Lemma 5.34 *If for a self-oblivious behavior σ_1 and an $a_1 \in A_1$, for every $\epsilon > 0$ there exists a time t such that for every oblivious, deterministic behavior σ_2 for the second agent, for every time $T > t$:*

$$\mathbf{E}_{h \in P^T_{\sigma_1,\sigma_2}}[R^{ext,a_1}(h)] < \epsilon.$$

Then, for any arbitrary behavior σ'_2:

$$\Pr_{h \in P^T_{\sigma_1,\sigma'_2}}[R^{ext,a_1}(h) > 2\epsilon] < exp\left(\frac{-T\epsilon^2}{8|u_1|^2}\right).$$

Proof: Choose σ'_2 to be any arbitrary behavior and $T > t$. From Lemma 5.33, we can define a oblivious, deterministic behavior σ^*_2 such that:

$$V_{\sigma_1}(h) \leq E_{h \in P^T_{\sigma_1,\sigma^*_2}}[R^{ext,a_1}(h)] < \epsilon.$$

Thus, we have captured the "expected" part of the regret in an arbitrary behavior for the second agent. Now, for all $i \in \{1, \ldots, T\}$, we define Y_i:

$$Y_i(h) = R^{ext,a_1}(h_i) - \mathbf{E}_{a'_1 \in \sigma_1(h(i-1))}[R^{ext,a_1}(a'_1, h_{i,2}))].$$

For all $h \in H$,

$$\begin{aligned}
V^{rem}_{\sigma_1}(h) &= \frac{1}{T}\sum_{i=1}^{T} Y_i(h), \\
R^{ext,a_1}(h) &= V_{\sigma_1}(h) + \frac{1}{T}\sum_{i=1}^{T} Y_i(h) \\
&< \epsilon + \frac{1}{T}\sum_{i=1}^{T} Y_i(h).
\end{aligned}$$

For all i, for all $h \in H^i$:

$$\mathbf{E}_{h' \in P^i_{\sigma_1,\sigma'_2}}[Y_i(h')|Y_1(h) = Y_1(h') \ldots Y_{i-1}(h) = Y_{i-1}(h')] = 0.$$

In other words,[5] given any possible sequence $Y_1, \ldots, Y_{i-1}$, Y_i has an expected value of zero. Also, for all i, for all $h \in H$, $|Y_i(h)| \leq 2|u_1|$. Therefore, by Azuma's Lemma:

$$\begin{aligned}
\Pr_{h \in P^T_{\sigma_1,\sigma'_2}}[\sum_{i=1}^{T} Y_i(h) > T\epsilon] &< exp\left(\frac{-T\epsilon^2}{8|u_1|^2}\right), \\
\Pr_{h \in P^T_{\sigma_1,\sigma_2*}}[R^{ext,a_1}(h) > 2\epsilon] &< exp\left(\frac{-T\epsilon^2}{8|u_1|^2}\right).
\end{aligned}$$

■

Lemma 5.35 *If for a behavior σ_1, for every $\epsilon > 0$ there exists a time t such that for every oblivious, deterministic σ_2, for every $a_1 \in A_1$, for every time $T > t$:*

$$\mathbf{E}_{h \in P^T_{\sigma_1,\sigma_2}}[R^{ext,a_1}(h)] < \epsilon.$$

then, for any arbitrary behavior σ'_2:

$$\Pr_{h \in P^T_{\sigma_1,\sigma'_2}}[R^{ext}_1(h) > 2\epsilon] < |A_1| exp\left(\frac{-T\epsilon^2}{8|u_1|^2}\right).$$

[5]For a distribution $D \in \Delta(S)$, for a function $f : S \to \mathbf{R}$, and a predicate Q, we use the notation $\mathbf{E}_{x \in D}[P(x)|Q(x)]$ to indicate the expected value of $f(x)$ given $Q(x)$ is true when x is drawn from D.

Proof:

$$\begin{aligned}\Pr_{h\in P^T_{\sigma_1,\sigma'_2}}[R_1^{ext}(h) > 2\epsilon] &= \Pr_{h\in P^T_{\sigma_1,\sigma'_2}}[\exists a_1 \in A_1, R^{ext,a_1}(h) > 2\epsilon] \\ &\leq \sum_{a_1\in A_1} \Pr_{h\in P^T_{\sigma_1,\sigma'_2}}[R^{ext,a_1} > 2\epsilon] \\ &\leq |A_1| exp\left(\frac{-T\epsilon^2}{8|u_1|^2}\right).\end{aligned}$$

■

Proof (of Lemma 5.16): For a given $\epsilon > 0$, we wish to find t''' such that:

$$\Pr_{h\in\mu_{\sigma_1,\sigma_2}}[\exists T > t''', R_1^{ext}(h(T)) > \epsilon] < \epsilon.$$

We begin by decomposing the infinite sequence of events. Now, for all t:

$$\begin{aligned}\Pr_{h\in\mu_{\sigma_1,\sigma_2}}[\exists T > t, R_1^{ext}(h(T)) > \epsilon] &\leq \sum_{T=t+1}^{\infty} \Pr_{h\in\mu_{\sigma_1,\sigma_2}}[R_1^{ext}(h(T)) > \epsilon] \\ &\leq \sum_{T=t+1}^{\infty} \Pr_{h\in P^T_{\sigma_1,\sigma_2}}[R_1^{ext}(h) > \epsilon].\end{aligned}$$

Define $\epsilon' = \epsilon/2$. There exists a t' such that, for all $T > t'$:

$$\Pr_{h\in P^T_{\sigma_1,\sigma_2}}[R_1^{ext}(h) > 2\epsilon'] < |A_1| exp\left(\frac{-T(\epsilon')^2}{8|u_1|^2}\right).$$

Summing we get, for all $t \geq t'$:

$$\Pr_{h\in\mu_{\sigma_1,\sigma_2}}[\exists T > t, R_1^{ext}(h(T)) > \epsilon] \leq \sum_{T=t+1}^{\infty} |A_1| exp\left(\frac{-T(\epsilon)^2}{32|u_1|^2}\right).$$

This is a geometric series. For simplicity, define r:

$$r = \exp\left(\frac{-(\epsilon)^2}{32|u_1|^2}\right).$$

The important fact is that $0 < r < 1$. Calculating the sum:

$$\Pr_{h\in\mu_{\sigma_1,\sigma_2}}[\exists T > t, R_1^{ext}(h(T)) > \epsilon] \leq \frac{|A_1| r^{t+1}}{1-r}.$$

Define t'':

$$t'' = \frac{1}{\ln r}\ln\left(\frac{(1-r)\epsilon}{|A_1|}\right) - 1.$$

Thus, for all $t > t''$:

$$\frac{|A_1| r^{t+1}}{1-r} < \epsilon.$$

Thus, if $t''' = \lceil \max(t', t'') \rceil$:

$$\Pr_{h\in\mu_{\sigma_1,\sigma_2}}[\exists T > t''', R_1^{ext}(h(T)) > \epsilon] \leq \epsilon.$$

■

5.8 Conclusions and Future Work

This chapter has defined an online convex programming problem. We have established that gradient descent is a very effective technique on this problem. This work was motivated by trying to better understand the infinitesimal gradient ascent algorithm, and the techniques developed we applied to that problem to establish an extension to infinitesimal gradient ascent that is no-external-regret.

The simplicity of the algorithm allows for the expansion of these results into other areas. For instance, here we deal with a Euclidean geometry: what if one considered gradient descent on a non-Euclidean geometry, like [Ama98, MW01]? Also, the simplicity of GIGA allows for this algorithm to be extended for even stronger results, like WoLF[BV01].

Chapter 6

Response Regret

The concept of regret is designed for the long-term interaction of multiple agents. However, most concepts of regret do not consider even the short-term consequences of an agent's actions: e.g., how other agents may be "nice" to you tomorrow if you are "nice" to them today. For instance, an agent that always defects while playing the Prisoner's Dilemma will never have any internal or external regret. In this chapter, we introduce a new concept of regret, called response regret, that allows one to consider both the immediate and short-term consequences of one's actions. Thus, instead of measuring how an action affected the utility on the time step it was played, we also consider the consequences of the action on the next few time steps, subject to the dynamic nature of the other agent's responses: e.g. if the other agent always is nice to us after we are nice to it, then we should always be nice: however, if the other agent sometimes returns favors and sometimes doesn't, we will not penalize our algorithm for knowing when these times are. We develop algorithms for both external response regret and internal response regret, and show how if two agents minimize internal response regret, then they converge to a correlated equilibrium in repeated bimatrix games, stochastic games, and partially observable stochastic games.

6.1 Introduction

In a repeated bimatrix game, if the first agent uses a traditional no-external-regret behavior, then it considers how well it could have done with a fixed action *assuming that the actions of the second agent are fixed*. For instance, consider the Prisoner's Dilemma (see Section 1.6.1). In this game, if the first agent always defects, and the second agent cooperates on the first time step and then always defects, then the first agent has zero external regret.

Observe that the above history could be generated if the second agent was playing tit-for-tat: always playing what the first agent played on the time step before. And, if the second agent was playing tit-for-tat, the first agent could have done significantly better by always cooperating. However, traditional external regret does not measure this regret, and in fact if both agents always cooperate, the first agent will have a positive external regret.

To handle such issues, we introduce response regret. Response regret is a compromise between doing what is optimal given the other agent's behavior (which is impossible without knowing the other agent's behavior in advance) and doing as well as the best from some class of functions given the second agent is unaffected by the first agent's actions. This is similar to research where one considers how well one could do if the world were a finite state automata. Response regret basically considers the short-term effects of any actions that an agent makes.

Assume that we are playing a discounted repeated Prisoner's Dilemma (see Section 1.7.1) with a discount factor of $\gamma = 2/3$. Suppose that the second agent has the following strategy (called a grim-trigger [OR94, p. 141] behavior):

1. If the other agent has never defected, cooperate.

2. Otherwise, defect.

This is a behavior for the second agent that can be represented as a function $\sigma_2 : H \to \Delta(A_2)$. What should the first agent do? If the first agent always cooperates, then it receives a utility of -3. If the first agent defects on a time step, then after that time step it should always defect. Thus, it will receive -1 on each time step before it defects, 0 on the first time step it defects, and -5 after that. If the first time step it defects on is t, then it will receive a utility of:

$$(-1)\left(1+\ldots+\left(\frac{2}{3}\right)^{t-2}\right)+(0)\left(\frac{2}{3}\right)^{t-1}+(-5)\left(\left(\frac{2}{3}\right)^{t}+\left(\frac{2}{3}\right)^{t+1}+\ldots\right).$$

This reduces to:

$$-3-7\left(\frac{2}{3}\right)^{t-1}.$$

This approaches -3 as t approaches infinity. Thus, it is better never to defect, and any behavior for the first agent that never defects if the second agent always cooperates is a best response to the grim-trigger behavior. The grim-trigger behavior is therefore a best-response to itself (an equilibrium).

When both agents use a behavior that always defects, their discounted utilities are both -15. When they both use the grim-trigger behaviors, they always cooperate and receive a discounted utility of -3. Thus, by considering equilibria of the repeated game, one can perform better.

This brings up an important aspect of traditional no-external-regret algorithms. In the Prisoner's Dilemma, they will almost surely converge to almost always defecting. Can the concept of regret be modified to allow for agents to take advantage of equilibria of this form?[1]

This chapter will discuss convergence of "the average joint behavior" to the set of correlated equilibria, similar to [FV97] observing the empirical frequency distribution converging to the set of correlated equilibria of the bimatrix game, but different in that we discuss convergence to a correlated equilibrium of the repeated game.

In Section 6.2, we will begin by discussing one of the main drawbacks of traditional no-regret algorithms. Then we will show how our concept of external response regret resolves this issue. In Section 6.3, we will show how to modify **Exp4** [ACBFS02] to work in this domain. In Section 6.4, we will show how a stronger concept of regret, called epoch regret, allows us to utilize older algorithms. In Section 6.5, we introduce internal response regret, and show how two no-internal-response-regret agents will converge in the limit to the set of correlated equilibria of the repeated game. In Section 6.6, we extend these concepts of response regret to a more general type of game where the agents do not observe each others' actions but instead make some observation determined by the environment based on the last joint action and the history of all actions and observation up to that point. This generalizes POMDPs (partially observable markov decision processes) and stochastic games, and in this domain we can again converge to the set of all correlated equilibria.

6.2 Preliminaries

In this section, we introduce external response regret. We show how it is a natural extension of external regret when we think about external regret as playing against a robot built by an omniscient builder in a trial. We show an example of how two situations with the same history but different behaviors for the second agent can have drastically different response regret for the first agent. We also show how, even if the first agent and the robot (see below) use the same behavior, zero response regret is not guaranteed.

[1] An earlier algorithm [dFM03] does almost as well as the best expert even taking into account the actions of the other agent, but only if the environment is **flexible**: effectively, at least for the experts used, the environment is "forgiving" and one can recover from any mistake in the past. Our guarantees hold even if the environment is inflexible, and our measure of regret itself takes into account the flexibility of the environment.

6.2.1 Traditional Regret and the Omniscient Builder

Consider the following formulation of traditional external regret. We want to build a good behavior $\sigma_1 : H \to \Delta(A_1)$ for our agent to play against some agent playing $\sigma_2 : H \to A_2$, a deterministic behavior. We compare ourselves to an omniscient builder. The omniscient builder observes σ_1 and σ_2 play for T time steps and generate a history $h \in H^T$. The builder constructs a robot that can play one action $a_1^* \in A_1$. Now, after its construction, the robot is placed uniformly at random at some point in the history h. It plays for one time step, t and compares its performance on that time step to that of the first agent. Thus, if the robot is placed at time t, the robot chooses the action a_1^* and the second agent chooses the action $h_{t,2}$. Thus, the robot receives a utility of $u_1(a_1^*, h_{t,2})$, and the first agent receives a utility of $u_1(h_t)$ on the same time step. Thus, the expected (with respect to t) difference is:

$$\frac{1}{|h|}\sum_{t=1}^{|h|}\left(u_1(a_1^*, h_{t,2}) - u_1(h_t)\right).$$

Now, if the omniscient builder builds a robot to maximize its expected utility, this becomes:

$$\max_{a_1^* \in A_1}\frac{1}{|h|}\sum_{t=1}^{|h|}\left(u_1(a_1^*, h_{t,2}) - u_1(h_t)\right).$$

In other words, the builder knows the history, but not when in the history the robot will play. This is the traditional external regret. Now, in the prisoner's dilemma, such a robot would always want to defect. This is because even though the robot is dropped into a repeated game, it only plays one time step, and *never sees the consequences of its actions on later time steps.*

Now in external response regret, we have the robot play for a longer period of time, which we will call a **trial**. Now the omniscient builder constructs a robot with a behavior $\sigma_1^* : H \to A_1$. The robot is again placed at some time step t chosen uniformly at random. After each time it plays a time step, a coin is flipped. With a probability of γ, it plays another time step, unless time step $|h|$ has been played (the trial cannot last past the end of the observed history).

Define $h(t)$ to be the first t joint actions of a history h, and define $\sigma_{2,h}(h') = \sigma_2(h \circ h')$. Thus, if the robot is placed at time step t, then it will observe a behavior of $\sigma_{2,h(t-1)}$. Define the discounted utility of the ith agent for T time steps playing h to be:

$$u_i^{\gamma,T}(h) = \sum_{t=1}^{T}\gamma^{t-1}u_i(h_t).$$

Define $u_i^{\gamma,T}(\sigma_1,\sigma_2) = \mathbf{E}[u_i^{\gamma,T}(h(\sigma_1,\sigma_2))]$. Thus, the expected utility of the robot in a trial given it begins at time step t is:

$$u_1^{\gamma,(|h|-t)+1}(\sigma_1^*, \sigma_{2,h(t-1)}).$$

Thus, this is the utility of playing from time step t to $|h|$ with a discount factor of γ assuming the second agent begins at t with the same behavior that it was using before. The expected difference between the performance of the robot and the performance of the first agent in a trial is:

$$\frac{1}{|h|}\sum_{t=1}^{|h|}\left(u_1^{\gamma,(|h|-t)+1}(\sigma_1^*, \sigma_{2,h(t-1)}) - u_1^{\gamma,(|h|-t)+1}((h_t, h_{t+1}, \ldots, h_{|h|}))\right).$$

Define Σ_i' to be the set of all deterministic behaviors for the ith agent. When the omniscient builder designs the robot to maximize the robot's expected utility, the expected difference on a trial becomes:

$$R^{rext,\gamma}(h,\sigma_2) = \max_{\sigma_1^* \in \Sigma_1'}\frac{1}{|h|}\sum_{t=1}^{|h|}\left(u_1^{\gamma,(|h|-t)+1}(\sigma_1^*, \sigma_{2,h(t-1)}) - u_1^{\gamma,(|h|-t)+1}((h_t, h_{t+1}, \ldots, h_{|h|}))\right).$$

This is the **external response regret**: the expected difference in utility in a trial between the omniscient builder's robot and the first agent. Examples of this quantity in certain scenarios follow in the next section. Importantly, we show later that external response regret can be estimated in some cases, but it cannot be not known exactly without knowing σ_2.

6.2.2 Measuring What Could Have Been

Consider the following history h: the first agent σ_1 always defects for one hundred time steps. The second agent cooperated on the first time step, and always defected thereafter. Did the first agent choose a reasonable behavior?

Now, from the perspective of traditional external regret, this is a very reasonable behavior: a robot dropped to play for one time step would always defect, and thus given the above history, the traditional external regret would be zero.

However, the external response regret of the first agent in this scenario would depend on how the second agent would have responded if the first agent had cooperated. For instance, the second could have been playing tit-for-tat, grim-trigger, or could have been oblivious to any actions of the first agent. In a tit-for-tat behavior, one always plays the action the other agent played on the last time step (starting with cooperate on the first time step). The difference between tit-for-tat and a grim-trigger behavior is that the tit-for-tat behavior is "forgiving". Observe that the second agent could have played either behavior and generated h. However, $R^{rext,2/3}(h, GRIM)$ and $R^{rext,2/3}(h, TFT)$ are quite different, as we show below. Roughly, the reason is the first agent was doomed from the beginning of the second time step playing grim trigger, but when playing tit-for-tat, the first agent could have improved its situation at any time up until the very end.

If the first agent was playing against grim-trigger, then it could have conceivably done much better from the first time step. However, given its behavior on the first time step, it could have done no better than it did. The omniscient builder would design a robot that always defects in this scenario (i.e., behave the same as the first agent), and therefore the first agent would not have any external response regret. The same would be the case if the second agent was completely oblivious to the first agent.

If the first agent was playing against tit-for-tat, then it could have improved its outcome at any time by playing cooperate. The omniscient builder knows this and designs a robot that cooperates for 99 time steps, and then defects.[2] The first time step that the robot plays, it does 1 worse than σ_1. However, on second and later time steps, the second agent will cooperate with the robot, and thus the robot will have four more utility than σ_1 on later time steps. The average trial lasts about 3 time steps,[3] so the external response regret is about 7.[4]

Now, observe that it is impossible for the first agent to compute its own response regret during the game. This is similar to the situation in [ACBFS02], where they consider a traditional regret setting where one does not know what the outcome of the other actions would have been.

6.2.3 Doing As Well As Slow Grim Trigger

Consider the **slow grim trigger** behavior. This behavior is like grim trigger and it begins by always cooperating. Instead of defecting after the other agent has defected once, slow grim trigger waits until the other agent has defected twice. Imagine that the first agent does not want to do as well as *any* robot that the omniscient builder constructs, but wants to do at least as well in a trial as a robot playing slow grim trigger.

Consider a behavior called **noisy tit-for-tat**. With a 90 percent probability, it does what the other agent did on the previous time step. With a 10 percent probability, it does the opposite. Observe that, if slow grim trigger and noisy tit-for-tat play against each other, they will begin by mostly cooperating. However,

[2] This defection on the one hundredth time step is because it is *certain* that the trial will end. However, the chance of the trial lasting one hundred time steps is $\frac{1}{100}\left(\frac{2}{3}\right)^{100} \approx 3 \times 10^{-20}$.

[3] Precisely, $2.98 + 2\left(\frac{2}{3}\right)^{100} = 2.98 \pm 5 \times 10^{-18}$.

[4] Taking into account the exact average trial length and the defection of the robot on its hundredth time step, the exact external response regret is $6.92 + 8.01\left(\frac{2}{3}\right)^{100} = 6.92 \pm 2 \times 10^{-17}$.

after an average of about twenty time steps, noisy tit-for-tat will have defected twice. Thus, if the first agent uses slow grim trigger against noisy tit-for-tat for a long time, the first agent will not do well. But, if the first agent uses slow grim trigger against noisy tit-for-tat for a short time, the first agent will probably do well.

Thus, if the first agent just played slow grim trigger from the beginning, after one thousand time steps, it would always defect and the noisy tit-for-tat agent would be defecting 90 percent of the time. However, if the robot begins a trial at any later point in the history, it will begin by cooperating. The first time step, noisy tit-for-tat will probably defect. Afterwards, noisy tit-for-tat will probably cooperate for a while. If $\gamma = 2/3$, with a probability of about $1/6$, the trial will not last long enough for tit-for-tat to defect again. Thus, the first agent will on average perform more poorly than slow grim trigger on the trial.

Since we are comparing against a robot that arrives at a random time, it seems reasonable that *restarting* the behavior in some fashion would work. For instance, we could play slow grim trigger over again every ten time steps, or every one thousand time steps. Now, the above example shows that restarting after every one thousand time steps (or more infrequently) will not work. However, restarting a behavior over a shorter period may not work either: perhaps the behavior does something special after ten time steps, but before then achieves no utility.[5]

In the following sections, we will divide time into **epochs**, and then use fairly standard techniques on these epochs. An epoch is a set of sequential time steps, and the set of all epochs partitions all the time steps. However, it is important to understand that we do not know that the robot begins its trial at the beginning of an epoch, so that simply minimizing regret in a haphazard fashion from the beginning of an epoch is insufficient. We will return to this point with an example in Section 6.4.3.

6.3 Applying an Experts Algorithm

Here, we state **Hedge** from [ACBFS02], and **EpochExp4**, a variant of **Exp4** from [ACBFS02]. **EpochExp4** achieves a low external response regret. Before beginning, let us assume that the utility of the first agent on any time step is between 0 and $\| u_1 \|$. Let us divide the time steps into epochs of length k. We can think of ourselves playing a deterministic behavior of the form:

$$\sigma_1 : \bigcup_{t=0}^{k-1} H^t \to A_1$$

on each epoch. Observe that since the epoch is only k time steps, we are only interested in the first k responses. Thus, there are a finite number of behaviors. If at the end of an epoch, we knew how well every behavior would have done had we played it, we could use the following algorithm:

[5]There is a subtle balance that shows more promise than either of these techniques: by choosing to randomly restart a behavior σ_1^* with a probability of $1-\gamma$, one can do in expectation almost as well in a trial as σ_1^*. This will be useful in future research.

Algorithm 6.2: Hedge
Parameters: A real number $\eta > 0$.
Initialization: Set $G_i(0) = 0$ for $i = \{1, \ldots, N\}$.

Repeat for epoch $\tau = 1, 2, \ldots$ until game ends

1. Choose behavior i_τ according to the distribution $\mathbf{p}(\tau)$ where
$$p_i(\tau) = \frac{\exp(\eta G_i(\tau - 1))}{\sum_{j=1}^{N} \exp(\eta G_i(\tau - 1))}$$
2. Receive the reward vector $\mathbf{x}(\tau)$ and score gain $x_{i_t}(\tau)$.
3. Set $G_i(\tau) = G_i(\tau - 1) + x_i(\tau)$ for $i = 1, \ldots, N$.

Thus, $x_i(\tau)$ would be the utility of behavior i on epoch τ. However, although we do not know the value of $x_i(\tau)$, we can form an **unbiased estimator**, a random variable we **can** observe that has the same expected value as what we **want** to observe.

The following example elucidates some of the issues of constructing an unbiased estimator. Imagine that there is an Italian restaurant that sometimes has a special on linguini. Imagine that we went to a Italian restaurant every day, and they had a special on linguini 30 times over the course of one hundred days. Now, over the next one hundred days, instead of going to the restaurant every day, we go there every even day. Surprisingly, we never saw the special at all. However, this may be because that the special is only offered on odd days. Over the next one hundred days, we flip a fair coin each day, and only if it is heads do we go to that restaurant. We see the special 15 times over those days. Now, it may be that they don't offer the special as much any more, but we coincidentally go every time they offer it. However, it is more likely that since we go half as often, we only see the linguini special half the time it is there, so we should estimate that the number of times the special was offered was the number of times we saw it multiplied by two.

Now, on days when we don't go to the restaurant, we do different things. Sometimes on our way to the Italian restaurant, we pass a food court. If the food court is open, we can go there instead, and choose either Chinese food or Vietnamese food. But if it is closed, we can go to the Italian restaurant or the Greek restaurant. Observe that there are several behaviors here, which are conditional on whether or not the food court is open.

- I Italian always
- G Greek always
- CI Chinese if the food court is open, Italian otherwise.
- CG Chinese if the food court is open, Greek otherwise.
- VI Vietnamese if the food court is open, Italian otherwise.
- VG Vietnamese if the food court is open, Greek otherwise.

On some day, imagine we choose a behavior according to the distribution:

$$\Pr[I] = 0.1 \qquad \Pr[G] = 0.2 \qquad \Pr[CI] = 0.1 \qquad \Pr[CG] = 0.1 \qquad \Pr[VI] = 0.2 \qquad \Pr[VG] = 0.3.$$

Now, observe that if some day, the food court is closed, and we go to eat Italian food, then our behavior could have been I,CI, or VI. Thus, when we measure the performance on this day, we know that if we had chosen *any* of the above behaviors, we would have gotten the performance we achieved. Given the fact that the food court was closed, the probability of us performing as we did that day was $\Pr[I \vee CI \vee VI] = 0.4$. Thus, whatever utility we measure, we multiply by $2.5 = \frac{1}{0.4}$ (like we multiplied by 2 when the probability was $\frac{1}{2}$), and attribute it to all of the possible behaviors that could have generated what we did that day. We can apply the same technique to estimate the utility of behaviors we do not observe.

Define S_i to be the possible histories if behavior i is played, and s^i to be the deterministic behavior i itself. Define:

$$\sigma^\tau : \bigcup_{k'=0}^{k-1} H^{k'} \to \Delta(A_1)$$

to be the behavior on epoch τ *before* i_τ is selected: in other words:

$$\sigma^\tau(h)(a_1) = \frac{\sum_{i:h\in S_i, s^i(h)=a_1} p_i(\tau)}{\sum_{i:h\in S_i} p_i(\tau)}.$$

Define $h^\tau = (h_{k(\tau-1)+1}, \ldots, h_{k\tau})$ to be the history of epoch τ. Then:

$$\bar{x}_i(\tau) = \begin{cases} \frac{u_1^\gamma(h^\tau)}{P_{\sigma^\tau}(h^\tau)} & \text{if } h^\tau \in S_i, \\ 0 & \text{otherwise.} \end{cases}$$

This is an unbiased estimator of the utility of using s^i on each epoch. Here, we present a simplified version of **Exp4** for this setting.

Algorithm 6.4: EpochExp4
Parameters: Reals $\eta > 0$ and $\beta \in (0, 1]$
Initialization: Initialize Hedge(η).
Repeat for $\tau = 1, 2, \ldots$ until game ends

1. Get the distribution $p(\tau)$ from Hedge.
2. Select behavior i_τ to be j with probability $\hat{p}_j(\tau) = (1-\beta)p_j(\tau) + \frac{\beta}{K}$. Define σ_1' to be the random behavior.
3. Receive reward $x_{i_t}(\tau) \in [0, 1]$.
4. Feed the simulated reward vector $\hat{x}(\tau)$ back to Hedge, where $\hat{x}_j(\tau)$ is the simple unbiased estimator of the utility of j given that we played σ_1'.

Theorem 6.5 *For any $\epsilon > 0$, $\gamma > 0$, there exists a k,β,η, and $\mathcal{T}$ such that **EpochExp4** has an expected external response regret (with discount factor γ) of less than ϵ after $\mathcal{T}$ epochs of length k.*

Applying this algorithm achieves low external response regret. Increasing k and decreasing β and η over time slowly enough makes the external response regret approach zero. There is a temptation here to think that *any* algorithm that works in a non-stochastic bandits setting will work here when we consider epochs to be time steps of a larger game. However, as we show in Section 6.4.3, this is not the case. In addition, the algorithm should "handle subroutines well". We describe this formally in Section 6.4.5.

6.4 Using Epoch Regret to Analyze External Response Regret

One problem with dealing with external response regret is that one never really knows all the consequences of an action. Thus, in order to make this problem more tractable for traditional online learning algorithms, we place hard limits on how long we measure the effects of an action.

In epoch regret, one first partitions the time steps into epochs. For each epoch $\tau \in \{1, \ldots, \mathcal{T}\}$, define b_τ to be the first time step in the epoch, f_τ to be the final time step of the epoch.

The optimal algorithm still starts at a random point in time. However, the trial ends deterministically if the epoch in which it began ends. If a trial begins in epoch τ at time t, the expected difference between the utility of the robot using σ_1^* and the utility of the agent generating h during the trial is:

$$u_1^{\gamma,(f_\tau - t)+1}(\sigma_1, \sigma_{2,h(t-1)}) - u_1^{\gamma,(f_\tau - t)+1}((h_t, h_{t+1}, \ldots, h_{f_\tau})).$$

Thus, f_τ, the end of the epoch, has replaced $|h|$, the end of the history h, as the hard deadline for the end of the trial. For a history h played against a deterministic agent $\sigma_2 : H \to A_2$, define the epoch regret to be:

$$R^{epochext,\gamma}(h, \sigma_2) = \frac{1}{|h|} \max_{\sigma_1 \in \Sigma_1'} \sum_{\tau=1}^{\mathcal{T}} \sum_{t=b_\tau}^{f_\tau} \left(u_1^{\gamma,(f_\tau - t)+1}(\sigma_1, \sigma_{2,h(t-1)}) - u_1^{\gamma,(f_\tau - t)+1}((h_t, h_{t+1}, \ldots, h_{f_\tau})) \right).$$

6.4.1 The Epoch Penalty

Here, we measure how different the epoch regret and the external response regret can be. We do this by considering a measure halfway in-between. Imagine that, if a trial reaches the last time step of an epoch, then that time step is played, the trial ends, and then the robot receives an *additional* utility of $\frac{\gamma}{1-\gamma} \max_{a_1,a_2} u_1(a_1, a_2)$, and the first agent receives an additional utility of $\frac{\gamma}{1-\gamma} \min_{a_1,a_2} u_1(a_1, a_2)$. We will call the expected difference between the robot's utility and the first agent's utility the **hybrid regret**. Observe that the utility added is the maximum expected utility remaining for the robot at that point in an external response regret trial, and the minimum expected utility remaining for the first agent at that point in a external response regret trial. Thus, hybrid regret exceeds external response regret.

The difference between hybrid regret and epoch regret depends on the probability of reaching the last time step, which in turn depends on the length of the epoch. If all epochs are of length k, with a probability of $\frac{1}{k}$, one begins in the last time step. If one begins on the second-to-last time step, there is a probability of γ of reaching the last time step. If one begins on the first time step, there is a probability of γ^{k-1} of reaching the last time step. Thus, the probability of reaching the last time step on a trial is:

$$\frac{1}{k}\left(1 + \gamma + \ldots + \gamma^{k-1}\right) = \frac{1-\gamma^k}{k(1-\gamma)}.$$

If the trial does reach the end of the epoch, the difference between the epoch trial and the hybrid trial is exactly $\| u_1 \| \frac{\gamma}{1-\gamma}$. Thus, the expected difference between the epoch regret and the hybrid regret is $\| u_1 \| \frac{\gamma(1-\gamma^k)}{k(1-\gamma)^2}$, which we call the **epoch penalty**. This is the maximum difference between the external response regret and the epoch regret, because the hybrid regret exceeds the external response regret. As $k \to \infty$, this epoch penalty goes to zero. Thus, if one wishes to minimize external response regret, it is sufficient to minimize epoch regret if one increases the length of the epochs over time. Therefore, in the remainder of the section, we will focus on how **EpochExp4** minimizes epoch regret.

6.4.2 Giving More Information to the Robot

Imagine that, when the robot is placed at a point in the history, it is informed of the joint actions that occurred from the beginning of the epoch up until the last joint action before it arrived in the history. This additional information can only increase the utility of the robot (it can always simply ignore the information).

However, this is the robot to which we will compare. The reason for this is that it allows us to assume that if the robot is at some time step t, it will have the same information and the same motivations regardless of where it began, resolving one major issue of response regret.

For each $h' \in H$, define $E_{h'}$ to be the epochs that begin with h', formally:

$$E_{h'} = \{\tau : h' = (h_{b_\tau}, h_{b_\tau+1}, \ldots, h_{b_\tau+|h'|-1})\}.$$

Now, we can consider the regret if we restrict where the robot starts to be $|h'|$ steps after the beginning of epochs that start with h'. Define:

$$R_{h'}^{\gamma}(\sigma_2, h) = \max_{\sigma_1 \in \Sigma_1'} \frac{1}{|E_{h'}|} \sum_{\tau \in E_{h'}} \left(u_1^{\gamma,(f_\tau - (b_\tau + |h'|))+1}(\sigma_1, \sigma_{2,h(b_\tau+|h'|-1)}) - u_1(h'_{b_\tau+|h|}, h'_{b_\tau+|h|+1}, \ldots, h'_{f_\tau}) \right).$$

Since the robot now "knows" how the epoch begins and what step of the epoch it is on (but not which epoch), this type of regret is stronger than the epoch regret. We consider the regret when the robot's trial begins t time steps after the beginning of the epoch, but the robot knows how the epoch began,

$$R_t^\gamma(\sigma_2, h) = \frac{1}{\mathcal{T}} \sum_{h' \in H^t} |E_{h'}| R_{h'}^\gamma(\sigma_2, h).$$

6.4.3 The Complexity of Epoch Regret

Before beginning the proof that **EpochExp4** minimizes external response regret, it is important to understand how other similar techniques would not work, thus justifying some of the complexity of the proof. It is clear that **EpochExp4** makes R_0 approach zero, because R_0 is the external regret of a repeated game where each epoch is considered a time step. Thus, the guarantees in [ACBFS02] still hold.

However, what is more interesting are the quantities $R_1, \ldots, R_{k-1}$. An arbitrary non-stochastic regret-minimization algorithm treating epochs as time steps does not always make these (or $R^{epochext}$) approach zero, as we demonstrate below.

Consider a simple game of Matching Pennies. Imagine that each epoch is of length 2. On time steps $\{3, 7, 11, \ldots, 4i+3, \ldots\}$, the second agent plays T, otherwise it plays H, regardless of the actions of the first agent. On time steps $\{1, 5, 9, \ldots, 4i+1, \ldots\}$ the first agent plays h, otherwise it plays t.

Epoch	1		2		3		4	
Time Step	1	2	3	4	5	6	7	8
First Agent	h	t	t	t	h	t	t	t
Second Agent	H	H	T	H	H	H	T	H
Epoch Regret Contribution[6]	2γ	2	$2\gamma - 2$	2	2γ	2	$2\gamma - 2$	2
R_0^γ Contribution[7]	2γ		$2\gamma - 2$		2γ		$2\gamma - 2$	

Thus, in the first time step of each epoch, the first agent "guesses" correctly, and in the second step it "guesses" wrong. The robot starting uniformly from the beginning of some epoch would have to make a guess about what to do in the first time step, but it would always get the second time step correct. Thus, if the discount factor is less than 1/2, $R_0^\gamma < 0$ for this history. This indicates that an algorithm that was minimizing external regret from the beginning of each epoch could have generated this history.

However, observe that R_1^γ is in fact 2: if the robot is started at random at the second time step of some epoch, it will play H, and always be right, whereas in the above history, the first agent is always wrong on the second time step. In general, the epoch regret approaches $0.5 + \gamma$. The omniscient builder's robot plays the action h always.

6.4.4 Handling Multiple Scenarios and Subroutines with Hedge and Exp4

As the last section showed, using an arbitrary regret minimizer treating epochs as time steps will not be sufficient. Therefore, we need to show how **Hedge** and **Exp4** have special properties that are useful to minimize $R^1, \ldots, R^{k-1}$. In particular, minimizing R^1 involves minimizing regret at the end of the epoch *given the first action of each agent in that epoch.* This section shows how **Exp4** (and sometimes **Hedge**) elegantly handles the situation where time steps are distinguished into two sets by some event generated by the world, or when the time steps are distinguished by some action of the algorithm itself. The examples in this section help to illustrate the general technique used in Section 6.4.5 to show how **EpochExp4** handles the partitioning of epochs by the first few joint actions in the epoch, which is a hybrid of the two situations we discuss here.

[6] Expected Utility of always playing h from this point until the end of the epoch minus the observed discounted utility until the end of the epoch.

[7] Expected Utility of always playing h for the entire epoch minus the observed discounted utility until the end of the epoch.

Hedge and Multiple Scenarios

Imagine that there was a situation with a number of time steps, where on each time step t:

1. The world is either in state $S_t = A$ or $S_t = B$. This is known to us before we act on a time step. One of the following two decisions must be made:
2. If the world is in state A, should we use A_1 or A_2? After the choice, we observe $u_t(A_1)$ and $u_t(A_2)$.
3. If the world is in state B, should we use B_1, B_2, or B_3? After the choice, we observe $u_t(B_1)$, $u_t(B_2)$, and $u_t(B_3)$.

There are two ways to formulate this situation:

1. Run two copies of **Hedge**: one on time steps when the world is in state A, another when the world is in state B. For each state, regret will be minimized over the rounds where the world is in that state.
2. Run one copy of **Hedge**: The experts are now six pairs: (A_1, B_1), (A_1, B_2), (A_1, B_3), (A_2, B_1), (A_2, B_2), and (A_2, B_3). Each action means, "use my first part if the state is A, and my second part if the state is B." This randomization technique is closer to the one used in **EpochExp4**.

The interesting part is, regardless of the actions of the algorithm, the sequence of states of the world, or utilities, these two formulations will act identically, having the same distribution over $\{A_1, A_2\}$ if the state is A, and the same distribution over $\{B_1, B_2, B_3\}$ if the state is B.

Formally, define

$$\begin{aligned} G_{A_i}(t) &= \sum_{t':t' \leq t, S_{t'}=A} u_{t'}(A_i), \\ G_{B_j}(t) &= \sum_{t':t' \leq t, S_{t'}=B} u_{t'}(B_j), \\ G_{A_i,B_j}(t) &= G_{A_i}(t) + G_{B_j}(t). \end{aligned}$$

Define y_t to be the expert selected at time t, and a_t to be the action. Thus, if we use two copies of **Hedge**, and the state $S_t = A$, then:

$$\Pr[y_t = a_t = A_i] = \frac{\exp(\eta G_{A_i}(t-1))}{\sum_{i'=1}^{2} \exp(\eta G_{A_{i'}}(t-1))}.$$

If we use only one copy of **Hedge**, then:

$$\Pr[y_t = (A_i, B_j)] = \frac{\exp(\eta G_{A_i,B_j}(t-1))}{\sum_{i'=1}^{2} \sum_{j'=1}^{3} \exp(\eta G_{A_{i'},B_{j'}}(t-1))}.$$

If the state $s_t = A$, then:

$$\begin{aligned} \Pr[a_t = A_i] &= \sum_{j'=1}^{3} \Pr[y_t = (A_i, B_{j'})] \\ &= \frac{\sum_{j'=1}^{3} \exp(\eta G_{A_i,B_{j'}}(t-1))}{\sum_{i'=1}^{2} \sum_{j'=1}^{3} \exp(\eta G_{A_{i'},B_{j'}}(t-1))} \\ &= \frac{\sum_{j'=1}^{3} \exp(\eta G_{A_i}(t-1)) \exp(\eta G_{B_{j'}}(t-1))}{\sum_{i'=1}^{2} \sum_{j'=1}^{3} \exp(\eta G_{A_{i'}}(t-1)) \exp(\eta G_{B_{j'}}(t-1))} \\ &= \frac{\exp(\eta G_{A_i}(t-1))}{\sum_{i'=1}^{2} \exp(\eta G_{A_{i'}}(t-1))}. \end{aligned}$$

Thus, both formulations are identical. This implies that even when our experts are more complicated, for each state **Hedge** is still minimizing regret on the rounds when the world is in that state.

Exp4 and Multiple Scenarios

Just as there are two formulations of **Hedge** in the above scenario, there are also two formulations of **Exp4**. Let us assume we do not observe the utility of actions on time steps that we do not take them: i.e., we only observe the utility actually received.

Then, if we are in state A and choose action A_2, then we see $u(A_2)$, as well as $u(A_2, B_1)$, $u(A_2, B_2)$, and $u(A_2, B_3)$, for that time step. This means that it is still the case for any time step that:

$$\hat{x}_{A_i,B_j}(t) = \begin{cases} \hat{x}_{A_i}(t) & \text{if } s_t = A \\ \hat{x}_{B_j}(t) & \text{if } s_t = B. \end{cases}$$

Thus, as before, the probabilities are identical in the two formulations. This implies that for each state, **Exp4** is also minimizing regret over rounds where the world is in that state, even when it is using more complicated experts.

Exp4 and Multiple Subroutines

Imagine that there was a situation with a number of time steps, where on each time step, the following decisions needed to be made:

1. Should we choose technique A or B?
2. If we choose A, should we use A_1 or A_2?
3. If we choose B, should we use B_1, B_2, or B_3?

Now, imagine that one faced this situation repeatedly. In this case, one has five experts: A_1, A_2, B_1, B_2, and B_3. Imagine we observe only the utility of the action we choose: thus, **Exp4**[8] is a good algorithm to use.

For a moment, let us ignore how A or B is selected. On time steps where B was selected, did we do as well as the best of B_1, B_2, or B_3?

Let us consider the two cases: exploration (where we select an action uniformly at random) and exploitation (where we select an action according to $p_i(t)$).

If we know we are exploring, and we know we selected B_1, B_2, or B_3, then we are equally likely to have chosen any of them.

If we know we are exploiting, and we know we selected B_1, B_2, or B_3, then the probability of choosing B_j is:

$$\begin{aligned}
\Pr[a_t = B_j | a_t \in \{B_1, B_2, B_3\}] &= \frac{\Pr[a_t = B_j]}{\sum_{j'=1}^{3} \Pr[a_t = B_{j'}]} \\
&= \frac{\exp(\eta G_{B_j}(t-1))}{\sum_{X \in \{A_1,A_2,B_1,B_2,B_3\}} \exp(\eta G_X(t-1))} \div \\
&\quad \sum_{j'=1}^{3} \frac{\exp(\eta G_{B_{j'}}(t-1))}{\sum_{X \in \{A_1,A_2,B_1,B_2,B_3\}} \exp(\eta G_X(t-1))} \\
&= \frac{\exp(\eta G_{B_j}(t-1))}{\sum_{j'=1}^{3} \exp(\eta G_{B_{j'}}(t-1))}
\end{aligned}$$

Now, the key behind interpreting this is to observe that if $a_t \neq B_j$, then $\hat{x}_{B_j}(t) = 0$. Thus, if we ran **Exp4** only on time steps where we first chose to use Technique B, we would still be seeing the same output from **Hedge**.

[8] In this case, **Exp3** and **Exp4** from [ACBFS02] are equivalent.

So, the real difference is the ratio between exploration and exploitation: if technique B is worse than technique A, then given that we are using technique B, we are probably exploring. It is interesting to note that the property described in this section does not hold for **Hedge**. The conclusion is that even for each subroutine the regret is being minimized over rounds where that subroutine is used in exploitation (as opposed to exploration).

6.4.5 Handling Regret of Later Rounds: A Formal Model

Now that we have established that an arbitrary regret minimization algorithm applied to the epochs will not necessarily minimize epoch regret, yet that **Exp4** has some interesting properties that might allow it to minimize $R^1 \dots R^{k-1}$. In this section, we show that **EpochExp4** does in fact minimize epoch regret.

We are concerned about whether or not modifying the behavior in each epoch after a certain history h was observed will improve the performance. This is a generalization of the above two situations, because h is observed at the beginning of an epoch because the first agent (as with multiple subroutines) and second agent (as with multiple scenarios) both played actions in it. We will show in this section how on epochs where h was played, it is *as if* we had been running a special regret-minimizing algorithm there, modulo the ratio of exploration to exploitation.

Observe that in some epochs, the behavior that is chosen will never generate a history beginning with h, like in the previous section with multiple subroutines. On other epochs, the behavior of the second agent will never generate h, like in the previous section with multiple scenarios.

Define H^k_h to be the set of all histories of length k that begin with h:

$$H^k_h = \{h \circ h' : h' \in H^{k-|h|}\}.$$

A deterministic behavior can be represented by the set of histories it can generate. One can also represent a "restricted" behavior that does not have a plan of action for every outcome in a similar way. We will define a ***T*-restricted deterministic behavior**, a generalization of a "sleeping expert", as a set of histories S it will try to play such that:

1. If $h'' \in S$, and $j < |h''|$, then $h''(j) \in S$.
2. For all $h'', h' \in S$, if $|h'| \leq |h''|$, if for all $j \in \{1, \dots, |h'|\}$ it is the case that $h''_{j,2} = h'_{j,2}$, then $h' = h(j)$.
3. For all $h' \in T$, there exists an $h'' \in S$ such that $|h''| = k$ and for all $j \in \{1, \dots, k\}$, $h''_{j,2} = h'_{j,2}$.
4. If $h'' \in S$ and $|h''| = k$, then $h'' \in T$.

A T-restricted deterministic behavior is like a sleeping expert. Thus, a H^k_h restricted deterministic behavior σ_1 does not have a plan of action if the second agent deviates from the history h. A H^k restricted deterministic behavior would simply be a normal deterministic behavior. Define

$$B(h) = \{h' \in H^k : \forall j \in \{1, \dots, |h|\}, h'_{j,1} = h_{j,1} \text{ if } h(j-1) = h'(j-1)\}$$

to be the set of all histories such that the first agent does not deviate from h until the second agent deviates from h. Define $R(\sigma_1)$ to be the set of all deterministic behaviors that completes σ_1. Given a unrestricted (i.e. normal) deterministic behavior i, the **weight** of that behavior on epoch τ is:

$$W_\tau(i) = \exp(\eta G_i(\tau - 1)).$$

Define $W_\tau = \sum_i W_\tau(i)$.

The weight of an unrestricted behavior is proportional to its probability assigned by the **Hedge** subroutine of **EpochExp4**:

$$p_i(\tau) = \frac{W_\tau(i)}{W_\tau}.$$

The **full weight** of a restricted behavior is the sum of all the deterministic behaviors that form it:

$$W_\tau(\sigma_1) = \sum_{i \in R(\sigma_1)} W_\tau(i).$$

The probability that is assigned by **Hedge** to the set of behaviors that agree with σ_1 is:

$$p_{\sigma_1}(\tau) = \frac{W_\tau(\sigma_1)}{W_\tau}.$$

However, observe that if we have already observed h at the beginning of the epoch, then the probability is higher. Define $W_t(h)$ to be the sum of the weights of all the behaviors i such that contain $h \in S_i$. Then the probability that we are playing a behavior that agrees with σ_1 given that we have observed h is:

$$p_{\sigma_1|h}(\tau) = \frac{W_\tau(\sigma_1)}{W_\tau(h)}.$$

Also, the **restricted weight** of a T-restricted behavior σ_1 can be considered to be the weight associated with all epochs where the outcome was in T. Define σ_1^t to be the behavior of the algorithm on the τth epoch. Define h^τ to be the history on the τth epoch. Define $G_{\sigma_1,T}(\tau)$ as the weighted reward obtained on epochs that had outcomes in T:

$$G_{\sigma_1,T}(\tau') = \sum_{\{\tau : h^\tau \in T, \tau < \tau'\}} \frac{u_1^\gamma(h)}{P_{\sigma_1^\tau}(h^\tau)}.$$

Now, define the **internal weight** of σ_1 to be:

$$W_\tau^{int}(\sigma_1, T) = \exp(\eta G_{\sigma_1,T}(\tau - 1)).$$

We define $W_\tau^{int}(h)$ to be the sum of the internal weights of all H_h^k behaviors. The following lemma generalizes the observations of Section 6.4.4.

Lemma 6.6 *For any H_h^k-restricted deterministic behavior σ_1:*

$$\frac{W_\tau^{int}(\sigma_1, H_h^k)}{W_\tau^{int}(h)} = \frac{W_\tau(\sigma_1)}{W_\tau(h)} = p_{\sigma_1|h}(\tau).$$

Proof: We observe how two restricted behaviors can be combined to form a single unrestricted behavior. First, consider $B_h^k = \{h' : \forall j \in \{0, \dots, |h| - 1\}, h'(j) = h(j) \Rightarrow h'_{j+1,1} = h_{j+1,1}\}$ to be the set of all histories of length k where if there were any deviations from h in h', the second agent caused the first deviation. First, observe that a B_h^k-restricted behavior is a normal behavior, because for any $h' \in H^k$, there is an $h'' \in B_h^k$ such that for all j, $h'_{j,2} = h''_{j,2}$. This is because B_h^k really only restricts the actions of the first agent. Also, $H_h^k \subseteq B_h^k$. Now, if we consider $(B_h^k \backslash H_h^k)$-restricted behavior σ_1'', and a H_h^k-restricted behavior σ_1, then we observe that the behavior represented by $S_{\sigma_1} \cup S_{\sigma_1''}$ is a normal deterministic behavior, which we call σ_1'''. It is the case that:

$$\begin{aligned} W_\tau(\sigma_1''') &= \exp(\eta G_{\sigma_1'''}(\tau - 1)) \\ &= \exp(\eta G_{\sigma_1'', H_h^k}(\tau - 1) + \eta G_{\sigma_1, B_h^k}(\tau - 1)) \\ &= W_\tau^{int}(\sigma_1'', H_h^k) W_\tau^{int}(\sigma_1, B_h^k). \end{aligned}$$

This is because any history that can be an outcome of σ_1''' is either an outcome of using σ_1'' or an outcome of using σ_1.

Observe that any behavior in $R(\sigma_1)$ can be formed by combining a $(B_h^k \backslash H_h^k)$-restricted deterministic behavior with σ_1 in the above fashion. Define C_h^k to be the set of all $(B_h^k \backslash H_h^k)$-restricted behaviors. Then:

$$W_t(\sigma_1) = W_t^{int}(\sigma_1) \sum_{\sigma_1'' \in C_h^k} W_t^{int}(\sigma_1'').$$

Now, observe that for some σ_1^{iv}:

$$W_t(\sigma_1^{iv}) = W_t^{int}(\sigma_1^{iv}) \sum_{\sigma_1'' \in C_h^k} W_t^{int}(\sigma_1'').$$

Thus, the two behaviors internal and full weights are proportional. The result follows. ■

What we have shown is that the algorithm after it plays any history h more or less behaves as if there were a unique version of **EpochExp4** running at that point.

Define

$$\hat{x}_{\sigma_1}(t) = \begin{cases} \frac{u_1^{\gamma}(h^t)}{p_{\sigma_1^t}(h^t)} & \text{if } h^t \in \sigma_1, \\ 0 & \text{otherwise.} \end{cases}$$

However, there are some differences. First, observe that the behavior in an epoch following a history that **EpochExp4** is learning to avoid is going to be disproportionately exploratory: the reason **EpochExp4** got there in the first place is probably because it was choosing a path that had not performed well in the past.

But, overall, at every length of history, the amount of overall exploration is fixed. Thus, it makes sense to bound the average regret starting from some time step k' in each epoch. In order to do this, define a ***k*′-partial behavior** as a behavior represented by a function of the form $\sigma_1^* : \bigcup_{k''=k'}^{k} H^{k''} \to A_1$. Thus, if on each epoch, we replaced our behavior by σ_1^*, how much better would we have done? Define $\sigma_2^\tau : H \to A_2$ to be the behavior of the second agent on the τth epoch. Define $(\sigma_1^*|\sigma_1')$ to be the randomized behavior that plays like σ_1' for the first k' time steps and like σ_1^* afterwards. Define $\Sigma_1^{k',k}$ to be the set of all k'-partial behaviors. Then for all $\sigma_1 \in \Sigma_1^{k',k}$, define:

$$\begin{aligned} G_{\sigma_1}^{k'}(\mathfrak{T}) &= \sum_{\tau=1}^{\mathfrak{T}} u_1^{\gamma,k}((\sigma_1|\sigma_1^\tau), \sigma_2^\tau), \\ G_{max}^{k'}(\mathfrak{T}) &= \max_{\sigma_1 \in \Sigma_1^{k',k}} G_{\sigma_1}^{k'}(\mathfrak{T}). \end{aligned}$$

This is the maximum utility obtainable by changing the behavior at the end of each epoch to be some fixed behavior. The expected gain of the algorithm is:

$$EG_{\mathbf{EpochExp4}}(\mathfrak{T}) = \sum_{\tau=1}^{\mathfrak{T}} u_1^{\gamma,k}(\sigma_1^\tau, \sigma_2^\tau).$$

The important aspect is that the behavior during the first k' time steps of each epoch are not changed. Define $\Phi_M(x)$ as in [ACBFS02]:

$$\Phi_M(x) = \frac{e^{Mx} - 1 - Mx}{M^2}.$$

Theorem 6.7 *Define $K = |A_1|^k$, and $K' = |A_1|^{k-k'}$. Define N' to be the number of H_h^k behaviors if $|h| = k'$. Define $v = \frac{K\|u_1\|}{\beta(1-\gamma)}$. For any $k' < k$, it is the case that:*

$$EG_{\mathbf{EpochExp4}}(\mathfrak{T}) = EG_{max}^{k'}(\mathfrak{T}) - \left(\beta + \frac{K'\Phi_v(\eta)}{\eta}\frac{\|u_1\|}{1-\gamma}\right) EG_{max}^{k'}(\mathfrak{T}) - \frac{(1-\beta)|H^{k'}|}{\eta} \ln N'.$$

Proof: Remember that we have assumed that all utilities of the first agent on a time step are between 0 and $\|u_1\|$. Observe that $\hat{x}$ is bounded above by v. First, x is bounded above by $\frac{\|u_1\|}{1-\gamma}$. Second, when the agent explores, the first agent has an equal probability of playing any sequence $(A_1)^k$. Thus, $P_{\sigma_1^\tau}(h^\tau)$ is bounded below by $\frac{\beta}{K}$.

Observe that any k'-partial behavior σ_1^* can be decomposed into a set $\{\sigma_1^h\}_{h\in H^{k'}}$, where for each $h \in H^{k'}$, σ_1^h is a H_h^k-restricted behavior. Observe that, for all $h \in H^{k'}$, for the probabilities returned by **Hedge** over the H_h^k-restricted behaviors on epochs where h is played, for any H_h^k restricted deterministic behavior σ_1:

$$\sum_{\tau\in E_h}\sum_{\sigma_1'\in H_h^k} p_{\sigma_1'|h}(\tau)\hat{x}_{\sigma_1'}(\tau) \geq \sum_{\tau\in E_h}\hat{x}_{\sigma_1}(\tau) - \frac{\ln N'}{\eta} - \frac{\Phi_v(\eta)}{\eta}\sum_{\tau\in E_h}\sum_{\sigma_1'\in H_h^k} p_{\sigma_1'|h}(\tau)\hat{x}_{\sigma_1'}(\tau)^2.$$

Observe the deviation from the proof of Theorem 8.1 of [ACBFS02] in the next step. Remember that $P_{\sigma_1^\tau}(h)$ is the probability of σ_1^τ playing h. If $h^t(k') = h$, then:

$$\sum_{\sigma_1'\in H_h^k} p_{\sigma_1'|h}(\tau)\hat{x}_{\sigma_1'}(\tau) \leq \frac{u_1^{\gamma,k}(h^\tau)}{(1-\beta)P_{\sigma_1^\tau}(h)}.$$

Define ρ_1^τ to be the deterministic behavior selected on epoch τ.

$$\sum_{\sigma_1'\in H_h^k} p_{\sigma_1'|h}(\tau)\hat{x}_{\sigma_1'}(\tau)^2 \leq \frac{\hat{x}_{\rho_1^\tau}(h^\tau)}{(1-\beta)P_{\sigma_1^\tau}(h)}\frac{\|u_1\|}{1-\gamma}.$$

Similarly to the arguments in [ACBFS02]:

$$\sum_{\tau\in E_h} x_{\rho_1^\tau}(\tau) \geq P_{\sigma_1^\tau}(h)(1-\beta)\sum_{\tau\in E_h}\hat{x}_{\sigma_1}(\tau) - P_{\sigma_1^\tau}(h)\frac{1-\beta}{\eta}\ln N - \frac{\Phi_v(\eta)}{\eta}\sum_{\tau\in E_h} u_1^{\gamma,k}(h^\tau)\frac{\|u_1\|}{1-\gamma}.$$

The $P_{\sigma_1^\tau}(h)$ can be dropped :

$$\sum_{\tau\in E_h} x_{\rho_1^\tau}(\tau) \geq (1-\beta)\sum_{\tau\in E_h}\hat{x}_{\sigma_1}(\tau) - \frac{1-\beta}{\eta}\ln N - \frac{\Phi_v(\eta)}{\eta}\sum_{\tau\in E_h} u_1^{\gamma,k}(h^\tau)\frac{\|u_1\|}{1-\gamma}.$$

Summing over all $h \in H^{k'}$ yields for any k'-partial behavior σ_1^*:

$$\mathbf{E}[\sum_{\tau=1}^{\mathcal{T}} x_{\rho_1^\tau}(\tau)] \geq (1-\beta)\sum_{\tau=1}^{\mathcal{T}}\mathbf{E}[u_1^{\gamma,k}((\sigma_1^*|\sigma_1^\tau),\sigma_2^\tau)] - \frac{1-\beta}{\eta}|H^{k'}|\ln N' - \frac{\Phi_v(\eta)}{\eta}\sum_{\tau=1}^{\mathcal{T}} u_1^{\gamma,k}(h^\tau)\frac{\|u_1\|}{1-\gamma}.$$

Consider the set of k'-partial behaviors that play a fixed sequence in $(A_1)^{k-k'}$ after any h is observed in an epoch. The sum of the utilities of these K' behaviors must exceed $\sum_{\tau=1}^{T} u_1^{\gamma,k}(h^\tau)$, and they each independently achieve less than $EG_{max}^{k'}$.

$$\mathbf{E}[\sum_{\tau=1}^{\mathcal{T}} x_{\rho_1^\tau}(\tau)] \geq (1-\beta)\sum_{\tau=1}^{\mathcal{T}}\mathbf{E}[u_1^{\gamma,k}((\sigma_1^*|\sigma_1^\tau),\sigma_2^\tau)] - \frac{1-\beta}{\eta}|H^{k'}|\ln N' - \frac{\Phi_v(\eta)}{\eta}\sum_{\tau=1}^{\mathcal{T}} K' EG_{max}^{k'}\frac{\|u_1\|}{1-\gamma}.$$

Thus, when we take the maximum σ_1^* over all k'-partial behaviors, we obtain our desired result. ■

Theorem 6.8 *Define* $K = |A_1|^k$, $v = \frac{K}{\beta}\frac{\|u_1\|}{1-\gamma}$. *After* $\mathcal{T}$ *epochs of length* k, ***EpochExp4*** *has an expected external response regret of less than:*

$$\frac{1-(1/\gamma)^k}{k(1-(1/\gamma))}\left(\beta + \frac{K\Phi_v(\eta)}{\eta}\frac{\|u_1\|}{1-\gamma}\right)\frac{\|u_1\|}{1-\gamma} + \frac{1-(1/\gamma)^k}{k(1-(1/\gamma))}\frac{1-\beta}{\mathcal{T}\eta}|H^k|\ln|A_1| + \|u_1\|\frac{\gamma-\gamma^{k+1}}{k(1-\gamma)^2}.$$

Proof:

$$\mathbf{E}[R_{k'}^\gamma] = \frac{1}{\mathcal{T}}\left(\frac{1}{\gamma}\right)^{k'}\left(EG_{max}^{k'}(\mathcal{T}) - EG_{\mathbf{EpochExp4}}(\mathcal{T})\right).$$

The epoch regret is less than the average of R_t^γ, for all $t \in \{0,\ldots,k-1\}$. Observe that $EG_{max}^{k'}(\mathcal{T}) \leq \frac{\|u_1\|}{1-\gamma}\mathcal{T}$. Taking this into account and adding the epoch penalty yields the result. ■

6.5 Internal Response Regret

Traditional internal regret in a repeated bimatrix game can be framed in the following way. First, an algorithm interacts with an environment, and a history is observed. Then, an omniscient builder, who knows the environment and the history played, builds a robot that is placed at a point uniformly at random in the history, but doesn't know which one. Before the robot acts, it observes the next action that the other agent would have made, and then chooses an action.[9] As with the external version, internal response regret can be formed by having the trial extend to another time step with a probability γ. On this next step, the robot finds out what the algorithm *would have done in the next step*, given the robot's original deviation. Formally, we can imagine that we simulate the behavior of the algorithm after every history and show the results to the omniscient builder before we begin.

Now, in order to formally think of how the robot will play, we have to extend the concept of a "behavior" to include a "suggestion" for the next action. Thus, an optimal agent has a function of the form

$$\sigma_1'' : \bigcup_{t=0}^{\infty} H^t \times (A_1)^{t+1} \to A_1,$$

i.e., its choice given the history of the actions its seen, the suggestions its received in the past, and a suggestion of what to do next. Define Σ_1'' to be the set of all such deterministic functions. As before, we can consider a history to be a function of three behaviors: the suggestion giving function $\sigma_1 : H \to A_1$, the suggestion taking function $\sigma_1'' : \bigcup_{t=0}^{\infty} H^t \times (A_1)^{t+1} \to A_1$, and a deterministic behavior for the other agent $\sigma_2 : H \to A_1$. Defining it recursively, if $h' = h(\sigma_1, \sigma_1'', \sigma_2)$, then:

$$\begin{aligned} h'_{t,1} &= \sigma_1''(h'(t-1), \{\sigma_1(h'(0)), \sigma_1(h'(1)), \ldots, \sigma_1(h'(t-1)\}) \\ h'_{t,2} &= \sigma_2(h'(t-1)) \end{aligned}$$

Thus, in order to decide what to do next, σ_1'' asks σ_1 what it would do. Then, σ_1'' takes the history and the suggestions of σ_1 into account, and makes a decision. σ_2 just decides based on the history. Define $u_1^{\gamma,k}(\sigma_1, \sigma_1'', \sigma_2) = u_1^{\gamma,k}(h(\sigma_1, \sigma_1'', \sigma_2))$.

Given a deterministic behavior $\sigma_1 : H \to A_1$, a deterministic behavior $\sigma_2 : H \to A_2$, and a history h that they both would play, then the internal response regret is:

$$R^{int}(\sigma_1, \sigma_2, h) = \max_{\sigma_1'' \in \Sigma_1''} \frac{1}{|h|} \sum_{t=1}^{|h|} \left(u_1^{\gamma,(t-|h|)+1}(\sigma_{1,h(t-1)}, \sigma_1'', \sigma_{2,h(t-1)}) - u_1^{\gamma,(t-|h|)+1}(h_t, \ldots, h_{|h|}) \right).$$

Now, like before, we will consider an epoch version, and then give more information to the robot. Particularly, we will assume, if the robot is placed in an epoch, then it is aware of the deterministic behavior selected by the first agent during that epoch. Unfortunately, this means that the experts now become more complicated than before. This deterministic behavior must involve responses to things that could not possibly happen if that deterministic behavior was followed. For instance, even if Alice believes that in order to get lunch, she should go to an Indian restaurant and order chicken saag, she should still have a contingency plan in the event that she (or the robot in her situation) instead deviates and ends up in a Mexican restaurant.

Therefore, if the epochs are of length k, our experts are of the form $\sigma_1 : \bigcup_{t=0}^{k-1} H^t \to A_1$. If we assume that we only observe the outcome of choosing the deterministic behavior that was chosen, then we can use the algorithm from [HMC01b].

As before, we will consider a type of regret that is dependent on history. Particularly, we will define E_{h',σ_1} to be the set of epochs which began with h' and the first agent played $\sigma_1 : \bigcup_{t=0}^{k-1} H^t \to A_1$. Observe that now we need not supply a suggestion to the robot, because it could compute for itself the action that the first agent took. Thus, we define:

$$\begin{aligned} R_{h',\sigma_1}(\sigma_1'', \sigma_2, h) \quad &= \max_{\sigma_1' \in \Sigma_1'} \tfrac{1}{|E_{h',\sigma_1}|} \textstyle\sum_{\tau \in E_{h',\sigma_1}} \quad \Big(u_1^{\gamma, f_\tau+1-(b_\tau+|h'|)}(h(\sigma_1', \sigma_{2,h(b_\tau+|h'|-1)})) \\ &\qquad -u_1^{\gamma, f_\tau+1-(b_\tau+|h'|)}((h_{b_\tau+|h'|}, \ldots, b_{f_\tau})\Big). \end{aligned}$$

[9]Actually, this is closer to the concept of swap regret (see Section 1.7.3).

If Σ_1^k is the set of all deterministic behaviors of the form $\sigma_1 : \bigcup_{t=0}^{k-1} H^t \to A_1$, then

$$\frac{1}{|h|} \sum_{h' \in \bigcup_{t=0}^{k-1} H^t, \sigma_1 \in \Sigma_1^k} |E_{h',\sigma_1}| R_{h',\sigma_1}(\sigma_1', \sigma_2, h) \geq R_{int}(\sigma_1', \sigma_2, h).$$

However, observe that R_{h',σ_1} will approach zero if $R_{\emptyset,\sigma_1}$ will approach zero. If not, then every time σ_1 was used, a different behavior could be used after h' to increase the utility.

This establishes that:

Theorem 6.9 *Any internal regret-minimizing algorithm that only requires that it sees the outcome of its own action can be converted into a internal response regret-minimizing algorithm by making $R_{\emptyset,\sigma_1}$ approach zero.*

The remarkable thing about traditional internal regret is that if two agents play against each other while minimizing internal regret, they in some sense converge to the set of all correlated equilibria. A correlated equilibrium (with rational probabilities) can be represented in the following way. Imagine a referee has a bag of balls, and each ball is labeled with a joint action. Before it begins, the referee shows the bag to both agents. It then privately selects a ball uniformly at random, and whispers in each agent's ear its part of the joint action. The interesting thing is that each agent, given what they know about the balls in the bag, and the fact that they know the referee will drawn one at random and read it accurately, they can do no better than use the action the referee whispers in its ear. An ϵ-correlated equilibrium can be represented in the same way, except that the agents can improve by no more than ϵ by choosing some other action other than the one whispered by the referee.

The sense in which the two no-traditional-internal regret behaviors converge is as follows: imagine two agents played a history finite history h. A referee has a bag of $|h|$ balls: the ith ball is labeled h_i. According to the above procedure, the balls in the bag represent an ϵ-correlated equilibrium, and as the history grows longer, ϵ gets smaller.

Now, this can be extended to correlated equilibria of repeated games as well, and there are several different possibilities. Imagine now that instead of having joint actions, each ball had an deterministic behavior for each action. Now, instead of telling each agent its entire deterministic behavior, the referee only tells each agent its next move. If neither agent could do better than listen to the referee, the distribution over deterministic behaviors is a correlated equilibrium. If either couldn't do more than ϵ better, then it is an ϵ-correlated equilibrium.

Imagine that the first agent acts according to $\sigma_1 : H \to A_1$, the second agent acts according to $\sigma_2 : H \to A_2$, and the history observed is h. Now, imagine that the referee has $|h|$ balls, and writes $(\sigma_{1,h(i)}, \sigma_{2,h(i)})$ on the ith ball. As we argue below, this distribution would converge to a correlated equilibrium of the repeated game if both agents are minimizing internal response regret.

Observe that the actions of each agent can be considered to be the recommendations of the referee. If two algorithms A and B are placed at a point chosen uniformly at random in a history generated by two algorithms C and D minimizing internal response regret, then neither A nor B can do much better than simply doing what either C or D did.

Theorem 6.10 *If two agents minimizing internal response regret play against each other and generate h, then for any $\epsilon > 0$, with high probability, given a long enough history h, a joint distribution over behaviors determined by their behaviors and the history of play h in the above manner is an ϵ-correlated equilibrium.*

Observe that this correlated equilibrium is not necessarily subgame perfect. For example, imagine the following variant of tit-for-tat in the Prisoner's Dilemma: if the other agent cooperated the last time step, then cooperate, otherwise defect 9/10 of the time. Now, if the agents both use this behavior, then it is not in either's best interest to ever defect: the retaliation makes it not worth it. However, since this is the case, they will almost never defect, and thus they will almost never need to retaliate against a defection. Internal response regret is minimized, and yet a correlated equilibrium that is not subgame perfect is reached.

Thus, it is an open question whether there exists an achievable variant of internal response regret where if all agents use this variant, they will converge to a subgame perfect correlated equilibrium.

6.6 Multi-Stage Games

A two agent multi-stage game is a generalization of a POMDP or a stochastic game. On each stage, each agent i selects an action from A_i. Then, it receives an observation from O_i, as well as a utility dependent on its observation according to the function $u_i : O_i \to \mathbf{R}$. For now, assume that A_i and O_i are finite. Thus, the events in a single time step are (a_1, a_2, o_1, o_2). Define H_i^*, the set of all histories for the ith agent, to be:

$$H_i^* = \sum_{t=0}^{\infty} (A_i \times O_i)^t.$$

The environment can be represented by a single function:

$$\rho : H_1^* \times H_2^* \times A_1 \times A_2 \to \Delta(O_1 \times O_2).$$

Thus, the observations on a time step are a function of the histories before that time step and the actions of the agents on that time step. A behavior for agent i is a function of the form:

$$\sigma_i : H_i^* \to \Delta(A_i).$$

Now, often it will be convenient to consider the environment from the perspective of agent i:

$$\rho_i : H_i^* \times A_i \to \Delta(O_i).$$

ρ_1 can be constructed from ρ and σ_2, and ρ_2 can be constructed from ρ and σ_1. For some arbitrary $h \in H_1^*$, some arbitrary $a_1 \in A_1$, to determine $\rho_1(h, a_1)$, one must first construct a posterior belief of the history observed by the second agent h', given that h was observed by the first agent. Then, given this posterior belief, one must find a posterior belief of the action taken by the second agent on this time step, a_2. Finally, one finds a posterior belief of the observation about to be made.

6.6.1 Response Regret in the Multi-Stage Game

We focus on a definition of regret based solely on σ_1 and ρ_1, completely ignoring ρ and σ_2. We imagine again the omniscient builder observing the history of both agents as well as any random outcomes of our actions or the actions of the environment. Then, we imagine the builder's robot being placed uniformly at random at any place in the history in the situation of the first agent, and then playing for a number of time steps chosen from a hypergeometric distribution. As it plays, for external response regret, it only observes the observations the first agent would have made. For internal response regret, it observes the observations and the actions the first agent would have made.

Formally, assume ρ_i is deterministic, i.e. $\rho_i : H_i \times A_i \to O_i$. Assume that when we de-randomize the behavior of the first agent, we get $\sigma_i : H_i \to A_i$. Define Σ_i''' to be the set of all such behaviors. Observe that a history can be a function of a behavior and an environment. Thus:

$$R(\rho_i, h) = \max_{\sigma_i \in \Sigma_i'''} \sum_{t=1}^{|h|} \left(u_1^{\gamma,(|h|-t)+1}(\sigma_i, \rho_{i,h(t-1)}) - u_i^{\gamma,(|h|-t)+1}((h_t, \dots, h_{|h|})) \right).$$

$h(t)$ is the first t time steps of the history h. $\rho_{1,h(t-1)}$ is the environment of the first agent after $h(t-1)$ is observed.

As before, this situation can be easily handled by dividing the time steps into epochs, and using a variant of **Exp4**.

Now, in order to formally define internal response regret, we need to define a behavior in this setting that can take advice, a function of the form:

$$\sigma_i' : \bigcup_{t=0}^{\infty} (A_i \times O_i)^t \times (A_i)^{t+1} \to A_i.$$

Define Σ_i^{iv} to be the set of all such behaviors. As with internal response regret, we can define a history $h(\sigma_i, \sigma_i', \rho_i)$ as a function of a deterministic suggestion giving behavior $\sigma_i : H_i \to A_i$, a suggestion taking behavior σ_i', and a deterministic environment $\rho_i : H_i \times A_i \to O_i$. Given a deterministic behavior $\sigma_i : H_i \to A_i$, a deterministic environment $\rho_i : H_i \times A_i \to O_i$, and a history h that they both would play, then the internal response regret is:

$$R^{int}(\sigma_i, \rho_i, h) = \max_{\sigma_i'' \in \Sigma_1^{iv}} \sum_{t=1}^{|h|} \left(u_i^{\gamma,(|h|-t)+1}(h(\sigma_{i,h(t-1)}, \sigma_i'', \rho_{i,h(t-1)})) - u_i^{\gamma,(|h|-t)+1}((h_t, h_{t+1}, \ldots, h_{|h|})) \right).$$

As before, we can divide time into epochs and use the algorithm from [HMC01b]. Multiple agents will converge to the set of correlated equilibria in the same fashion as before.

6.6.2 Multi-Stage Games as a Generalization of Other Frameworks

The multi-stage game is a generalization of many things. For instance, if one imagined that the environment ρ ignored the history and the observation of each agent was the joint action pair they played together, then a multi-stage game would just be a repeated bimatrix game. On the other hand, if the observation of each agent was a joint action and some state $s \in S$, and the next observation was a function of the state in the previous observation and the most recent actions of the agents, then a multi-stage game would be a stochastic game.

The Markovian nature of the state transitions in a stochastic game can be explicitly represented by restricting the environment ρ within which one wants to minimize regret. However, such an assumption will probably not be very useful unless it is also assumed that the second agent is Markovian as well. If the other agent is attempting to learn as well, this is not a good assumption. Thus, it may be best to simply treat the system as a multi-stage game, and ignore the meanings of the observations made.

The hardest to prove is the fact that a multi-stage game can represent a partially observable environment with state. Imagine that there is some initial prior distribution over the states: this is never explicitly represented in the history of either agent: however, the observations of each agent will be drawn from a posterior distribution dependent upon the posterior distribution of the state given the previous actions and observations of both agents. Thus, at any point in time, it is unclear what state the game is in, and in fact may never be clear, even if an infinite history is accessible. Again, one can assume some Markovian properties about the states and observations, as well as a bound on the size of the underlying state space. These may or may not be useful in increasing the convergence rate.

6.6.3 Stochastic Games and Correlated Equilibria

Stochastic games are an extension of repeated games, where the world has a state $s \in S$. The state at time step $t+1$ is a function $T : S \times A_1 \times A_2 \to \Delta(S)$ of the state at time t and the joint action taken. The immediate utility of each agent is a function $u_i : A_1 \times A_2 \times S \to \mathbf{R}$ of the current state and the joint action taken. A history of an entire game is an infinite sequence of triples $h \in (A_1 \times A_2 \times S)^\infty$. The utility of an agent given a history is analogous to before:

$$u_i^\gamma(h) = \sum_{t=1}^{\infty} \gamma^{t-1} u_i(h_t).$$

Although one can consider Nash equilibria of behaviors that depend upon the entire history, in a stochastic game, there always exists a simple Nash equilibrium of the form (f_1, f_2) where f_i is a function $f_i : S \to \Delta(A_i)$, from the current state to a mixed strategy to use next [Sha53]. Similarly, a correlated equilibrium can be of the form $f : S \to \Delta(A_1 \times A_2)$, where the referee makes a suggestion to each agent at each state. People have tried to use techniques that have been successful on Markov decision processes (stochastic games with only one agent) on stochastic games. It was thought that if one could compute the expected discounted utility of each joint action for each state, then one could find the Nash (or correlated) equilibrium by simply finding

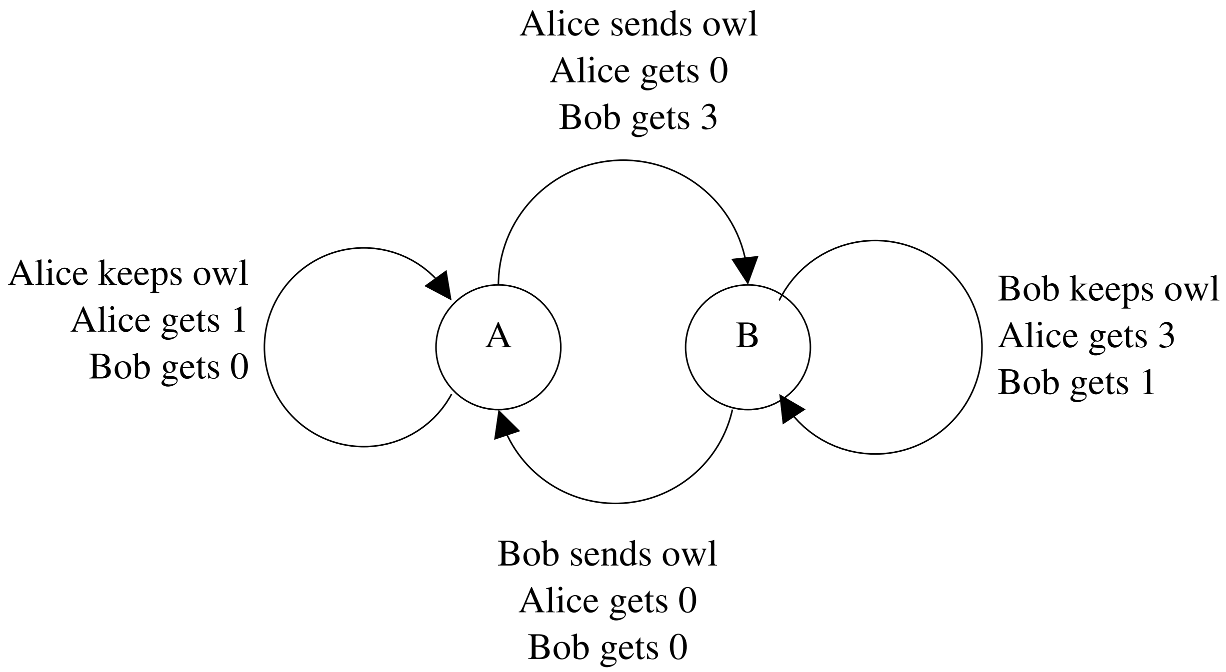

Figure 6.1: The Owl Game (or Marty's Game): when Alice keeps the owl, she gets 1 and Bob gets 0. When Alice sends the owl, she gets 0 and he gets 3. When Bob keeps the owl, she gets 3 and he gets 0. When Bob sends the owl, he gets 0 and she gets 0. The discount factor is 0.99.

a Nash (or correlated) equilibrium for each state independently. The following example shows why this is impossible, because the only Nash equilibrium is randomized, even though one agent is acting in any state.

Alice and Bob, brother and sister, bought an owl together. Since they live apart, they send it back and forth. Each sibling can keep the owl as long as he or she likes. However, after they bought it, Alice and the owl realized they did not like each other very much. But she *really* does not like sending the owl back and forth to Bob (it tends to bite and scratch her). She loves it when the owl is at Bob's place. Bob, on the other hand, likes the owl. Since Alice keeps the owl in a cage, the owl is always happy to see Bob, and both Bob and the owl really enjoy it when the owl comes to him. However, they both get bored when the owl just hangs around the house. When Bob sends the owl back to Alice, the owl gets depressed, and this depresses Bob. Bob hates it when the owl is gone.

Now, imagine that Bob likes receiving the bird so much, he is willing to send it out if he gets to receive it back the next day. This situation is very strange: the game has two states: Alice has the owl, or Bob has the owl. Each agent, when it has the bird, can either send the owl or keep it one more day. Observe that there is no deterministic Nash equilibrium where each agent just chooses one action.

Alice sends the owl, Bob keeps it Because Bob likes receiving the owl so much, he will send it to Alice just to get it back the next day.

Both send the owl Alice is getting scratched all the time. She realizes that it would be better if she just kept it.

Alice keeps the owl, Bob sends it Bob realizes he will not get the owl back if he sends it, so he no longer wants to part with it.

Both keep the owl Alice now wants to get rid of the bird by sending it to Bob.

The only equilibrium where both agent's strategies only depend on the state is randomized. However, realize that the reason that each agent randomizes is not to attempt to outguess the other agent, but to make the other agent indifferent *the time steps before the agent randomizes.* The reason that the techniques for Markov decision processes work so well is because that from every time step, the agent only needs to think about the future. But, in order to construct a Nash equilibrium for a stochastic game that is only dependent on state, an agent needs to choose its behavior in order to effectively justify choices other agents have made in the past. In fact, given the expected utilities of all the actions in all the states, *everything* looks like an equilibrium locally, even though globally there is only *one* Nash equilibrium (or correlated equilibrium).

In this section, we have shown that traditional techniques of local optimization useful in MDPs and zero-sum stochastic games fail to find a Nash equilibrium or correlated equilibrium in a nonzero-sum stochastic game. But, as we have shown previously, it is possible to converge

6.7 Conclusion

In this chapter, we have introduced external and internal response regret in repeated bimatrix games and multi-stage games. In introducing a concept of regret to multi-stage games that considers the short-term consequences of the actions of the agent, we have constructed a reasonable way to act in stochastic games. Two agents minimizing internal response regret will converge to the set of correlated equilibria. This has been one of the primary objectives of learning in stochastic games.

Chapter 7

Regret Minimization

No-regret online learning algorithms are very natural strategies in the context of repeated zero-sum games. E.g., algorithms such as randomized weighted majority, GIGA, and cautious fictitious play guarantee expected payoff comparable to the optimal fixed strategy in hindsight, which is at least the minimax optimum. In a nonzero-sum game, however, achieving good performance depends on both agents being "reasonable". Can an algorithm such as weighted majority guarantee a minimum Nash equilibrium value solely under the assumption that the other agent is using a no-regret algorithm? Can *any* no-regret algorithm? What about weighted majority playing against itself? We explore these questions and demonstrate families of games where these conditions are insufficient even to guarantee the minimum correlated equilibrium value. We also show that it is possible for an arbitrary algorithm to obtain the minimum Nash equilibrium value against a no-internal-regret algorithm whose utilities are unknown.

7.1 Introduction

Hedge [ACBFS02][1] is a very effective algorithm in zero-sum games. For instance, consider Rock-Paper-Scissors:

Game 7.2: Rock-Paper-Scissors

	R	P	S
r	0,0	-1,1	1,-1
p	1,-1	0,0	-1,1
s	-1,1	1,-1	0,0

In this game, regardless of how the second agent plays, the first agent using **Hedge** will almost surely get at least zero averaged over all time steps. Also, if the second agent did something foolish, e.g. played S on seven out of every ten time steps, then **Hedge** will almost surely get at least 0.4. If Agent 1 and Agent 2 both use **Hedge**, then both agents will receive zero value on average as they cycle around the Nash equilibrium.

In nonzero-sum games, however, interesting things can happen. Consider Shapley's Game.

Game 7.4: Shapley's Game

	R	P	S
r	0,0	0,1	1,0
p	1,0	0,0	0,1
s	0,1	1,0	0,0

Empirical tests show that **Hedge** will in fact spiral outward away from the Nash equilibrium in this game. This spiraling behavior is beneficial in Shapley's game, but highly detrimental in the Augmented Shapley Game.

[1]See Section 6.3 for a definition.

Game 7.6: Augmented Shapley Game

	R	P	S	G
r	0,0	0,1	1,0	2,0.34
p	1,0	0,0	0,1	2,0.34
s	0,1	1,0	0,0	2,0.34
g	0.34,2	0.34,2	0.34,2	2.01,2.01

Each agent can play Shapley's Game, or be "giving". The only Nash equilibrium is that both agents are giving. However, suppose we start **Hedge** with all agents having equal weights on all actions. In this case (see Fig. 38) both agents will again cycle, and have the probability of playing g or G drop exponentially. Thus, even though both agents would receive 2.01 if they played the Nash equilibrium, they both get less than 1.

While in a zero-sum game the minimax value is the only interesting value, in a nonzero-sum game, there are at least three interesting values for the first agent.

1. The safe value, or minimax value, that a no-external-regret algorithm is guaranteed regardless of who it plays.
2. The minimum correlated equilibrium value, or the lowest expected value for the first agent in any correlated equilibrium of the single-shot game,[2] which a no-internal-regret algorithm is guaranteed when it plays another no-internal-regret algorithm.
3. The minimum Nash equilibrium value, or the lowest expected value for the first agent in any Nash equilibrium of the single-shot game.

The minimax value is always less than or equal to the minimum correlated equilibrium value, which is always less than or equal to the minimum Nash equilibrium value. In the Augmented Shapley Game, the minimax value and the minimum correlated equilibrium value are both 0.34. The minimum Nash equilibrium value is 2.01. Thus, in a nonzero-sum game, getting the minimax value or minimum correlated equilibrium value is not always a very helpful guarantee. In fact, our first result is:

Theorem 7.7 *There exists a game (the Augmented Shapley Game) where the minimum Nash equilibrium value is strictly greater than the minimum correlated equilibrium value.*

A simple combination of this result with a result from [FV97] shows that there exist two no-internal-regret algorithms that play against each other and achieve a value below the minimum Nash equilibrium value.

But what if we have control over one of the algorithms? It is of course impossible to get the minimum Nash equilibrium value against any arbitrary algorithm. But what if the first agent knew that the second agent was achieving no-external-regret? It is known that if we have control over both algorithms, we can in self-play guarantee the minimum Nash equilibrium value, by having both algorithms always play the lexicographically first Nash equilibrium.[3] This can also be achieved while simultaneously achieving no regret.[4] However, why would one play the lexicographically first Nash equilibrium, as suggested in [CS03] and [BT02]? This is like in the Teaching problem[GK91],[5] where if one is allowed to choose the Teacher and the Learner, one can use a very efficient but unnatural language to communicate the knowledge of the Teacher to the Learner. Instead, in this paper we consider the situation where we have control of only one of the agents, but the other is an arbitrary no-regret agent. Having the second agent use a no-external-regret algorithm is like having the Learner choose a consistent hypothesis. If the first agent is allowed to behave arbitrarily, then we can prove:

[2] We define these in terms of the single-shot game because for a repeated bimatrix game with average reward, there is always a Nash equilibrium where the first agent gets its minimax value.

[3] One could also choose the lexicographically first equilibrium of those that maximize the sum of the expected utilities of the agents.

[4] [CS03] prove a slightly weaker result.

[5] In the Teaching problem, there is a Learner, who knows a set X and a hypothesis space H of functions of the form $h : X \to \{0, 1\}$, and a Teacher, who knows some $h \in H$. The problem is to have the Teacher communicate to the Learner the hypothesis h by presenting a set of labeled examples consistent with h. One way to study the problem is to design a Teacher work with *any* **consistent** Learner: any Learner that "learns" a hypothesis consistent with the labeled examples.

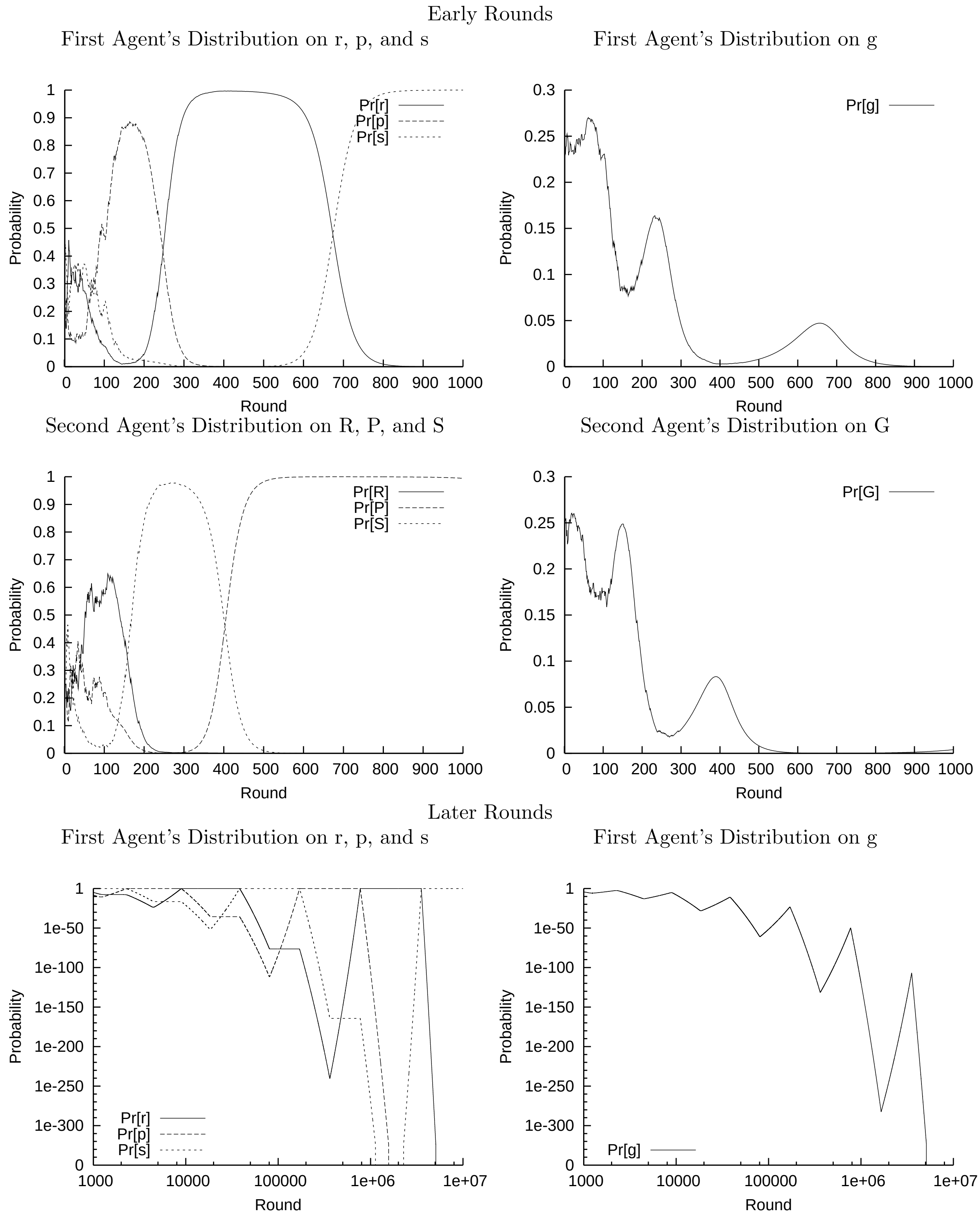

Figure 7.1: Two agents playing the Augmented Shapley Game using **Hedge**. Typical behavior if $p = 0$ for all actions for both agents. The erratic noise at the beginning is the result of randomly chosen actions.

Theorem 7.8 *There exists an algorithm for the first agent that takes as input the utilities of both agents and will achieve the minimum Nash equilibrium value with any no-external-regret algorithm for the second agent.*

The proof is in Section 7.8. The basic idea is that the first agent can simply compute a Nash equilibrium, and each round play a strategy near it. Since the second agent achieves no-external-regret, it will approach a best response to the first agent's algorithm.

The drawback with such a result is the lack of symmetry in the reasoning of the first agent. How can the first agent expect the second agent to achieve no-external-regret if the first agent itself is not doing so? Our main negative result is:

Theorem 7.9 *There exists a repeated bimatrix game such that for any no-external-regret algorithm A for the first agent there exists an no-external-regret algorithm B for the second agent such that the first agent using A will not achieve its minimum correlated equilibrium value when the second agent is using B.*

Corollary 7.10 *There exists a repeated bimatrix game such that for any no-external-regret algorithm A for the first agent there exists an no-external-regret algorithm B for the second agent such that the first agent using A will not achieve its minimum Nash equilibrium value when the second agent is using B.*

The proof sketch of the theorem is in Section 7.4. The corollary follows because the minimum Nash equilibrium value is higher than the minimum correlated equilibrium value. Now, in addition to no-external-regret algorithms like **Hedge**, there are also no-internal-regret algorithms [FV97, HMC00]. What happens if one makes the stronger assumption that the second agent achieves no-internal-regret?

If the first agent is unaware of the second agent's utilities, can the first agent achieve its own minimum Nash equilibrium value? Observe that what makes this difficult is that the first agent can never be sure of its own minimum Nash equilibrium value, because it is dependent on the utilities of the second agent. We have achieved the following result:

Theorem 7.11 *There exists an algorithm for the first agent that takes as input only the first agent's utilities and, for all nondegenerate bimatrix games, achieves the minimum Nash equilibrium value against every no-internal-regret algorithm for the second agent.*

The proof is in Section 7.11. A nondegenerate bimatrix game [Sha74] is roughly a game where there are no special relationships between the utilities of the two agents or the utility of one agent for any two joint actions. Thus, any bimatrix game becomes nondegenerate if all of the utilities are perturbed. However, as stated before, two arbitrary no-internal-regret algorithms are not guaranteed the minimum Nash equilibrium. The big open question that remains is:

Open Problem 7.12 *For any bimatrix game, does there exist a no-internal-regret algorithm for the first agent, such that for any no-internal-regret algorithm for the second agent, the first agent will achieve the minimum Nash equilibrium value?*

In this chapter, we present proof sketches of these last three theorems. Observe that the results here are more general than those presented in [HMC03], where they analyze a particular class of algorithms for which the change in the ith agent's strategy (distribution over actions) from one time step to the next is only a function of the other agent's current strategy and the utility of the ith agent: in other words, agents are not considering the whole history, which may be useful in general. Here, we are considering arbitrary no-external-regret behaviors, and prove that agents not only cannot converge to a Nash equilibrium, but that they cannot achieve the minimum utility they would receive in a Nash equilibrium or even in a correlated equilibrium.

7.2 Formal Definitions

If **NE** is the set of all Nash equilibria for the single-shot game, then the **minimum Nash equilibrium value** for the first agent is

$$\min_{(\beta_1,\beta_2)\in\mathbf{NE}} u_1(\beta_1,\beta_2) \ .$$

If **CE** is the set of all correlated equilibria for the single-shot game, then the **minimum correlated equilibrium value** for the second agent is:

$$\min_{D\in\mathbf{CE}} u_1(D) \ .$$

Internal and external regret for the second agent are defined symmetrically. Finally, we define what it means to achieve a minimum Nash equilibrium value.

Definition 7.13 *The first agent* ***achieves a value v*** *against a set of algorithms B for the second agent if for every $\sigma_2 \in B$, for every $\delta > 0$ there exists a function f where $\lim_{T\to\infty} f(T) = v$ such that with probability at least $1-\delta$, for all T:*

$$\frac{1}{T}\sum_{t=1}^{T} u_1(h_{t,1},h_{t,2}) > f(T) \ .$$

For all $\beta_1 \in \Delta(A_1)$,[6] define $BR(\beta_1) \subseteq \Delta(A_2)$ to be all best responses of the second agent to β_1. For an action $a_2 \in A_2$, define $BR^{-1}(a_2)$ to be the set of all strategies to which a_2 is a best response. The **support of a strategy** β_1 is the set of all actions that have strictly positive probability, $\{a_1 \in A_1 : \beta_1(a_1) > 0\}$. If a game is **nondegenerate**, and (β_1,β_2) is a Nash equilibrium, then:

1. Only actions in the support of β_2 are best responses to β_1, and vice-versa.
2. For all $a_2, a_2' \in A_2$, $u_1(\beta_1,a_2) \neq u_1(\beta_1,a_2')$.[7]
3. If $BR^{-1}(a_2)$ is nonempty, then its interior is nonempty and for all $a_2' \in A_2$, the interior of $BR^{-1}(a_2)$ does not intersect $BR^{-1}(a_2')$.

If a game does not have these properties, then almost any small perturbation of its utilities will cause it to have them.

7.3 Playing against a No-External-Regret Algorithm

It is quite simple to prove that, if one knows the utilities of a no-external-regret algorithm, and one does not care about regret, then one can achieve the minimum Nash equilibrium value. Define $V : \Delta(A_1) \to \mathbf{R}$:

$$V(\beta_1) = \min_{\beta_2\in BR(\beta_1)} u_1(\beta_1,\beta_2) \ .$$

V is the utility the first agent is guaranteed if it plays β_1 and the second agent plays a best response to β_1.

Lemma 7.14 *If the first agent plays the strategy β_1 every time step, and the second agent achieves no-external-regret, then the first agent achieves a value $V(\beta_1)$.*

Proof Sketch: Suppose that the empirical frequency of the second agent up until time T is $\bar{\beta}_{T,2}$. In other words, for each action $a_2 \in A_2$, the second agent has played action a_2 on $T\bar{\beta}_{T,2}(a_2)$ time steps on or before time step T. If the first agent plays a fixed distribution, then $\bar{\beta}_{T,2}$ almost surely must approach $BR(\beta_1)$ for the second agent's average external regret to approach zero. If this is the case, then the first agent will achieve a value of $V(\beta_1)$. ■

[6]For a set S, $\Delta(S)$ refers to the set of all distributions over S.

[7]This is not standard, but does result from a perturbation of the utilities. Observe that this implies that zero-sum games are degenerate.

One cumbersome aspect of V is that since it has discontinuities it may not have a global maximum. But, the next lemma shows that it does achieve values that are greater than or equal to the minimum Nash equilibrium value.

Lemma 7.15 *There exists a* $\beta'_1 \in \Delta(A_1)$ *such that* $V(\beta'_1)$ *is greater than or equal to the minimum Nash equilibrium value.*

Proof Sketch: We will sketch the proof for nondegenerate games. Additional complexities do arise in degenerate games, but the basic ideas are the same.

For a Nash equilibrium β_1, β_2, suppose that $A'_2 \subseteq A_2$ is the support of β_2. Because the game is nondegenerate, then A'_2 contains all actions that are best responses to β_1, so if for all $a_2, a'_2 \in A'_2$, $u_1(\beta_1, a_2) = u_1(\beta_1, a'_2)$, then $V(\beta_1) = u_1(\beta_1, \beta_2)$, proving our result.[8] If this is not the case, then define $a_2^{max} = \operatorname{argmax}_{a_2 \in A'_2} u_1(\beta_1, a_2)$, and $a_2^{min} = \operatorname{argmin}_{a_2 \in A'_2} u_1(\beta_1, a_2)$. Observe that $u_1(\beta_1, a_2^{max}) > u_1(\beta_1, \beta_2)$, because $u_1(\beta_1, a_2^{max}) > u_1(\beta_1, a_2^{min})$, $\beta_2(a_2^{min}) > 0$, and $u_1(\beta_1, \beta_2)$ is a weighted average of $u_1(\beta_1, a_2^{max})$, $u_1(\beta_1, a_2^{min})$, et cetera. Because $u_1(\beta_1, a_2^{max})$ is strictly greater than $u_1(\beta_1, \beta_2)$, there exists an open neighborhood W of β_1 where for all $\beta'_1 \in W$, $u_1(\beta'_1, a_2^{max}) > u_1(\beta_1, \beta_2)$.

In any game, the region $BR^{-1}(a_2^{max})$ is convex. In a nondegenerate game, the interior of $BR^{-1}(a_2^{max})$ is nonempty, and thus every open neighborhood on the boundary of $BR^{-1}(a_2^{max})$ intersects the interior. Since $\beta_1 \in BR^{-1}(a_2^{max})$, its neighborhood W intersects the interior of $BR^{-1}(a_2^{max})$. Thus, choose β''_1 to be a point in this intersection. The only best response to β''_1 is a_2^{max}, because it is in the interior of $BR^{-1}(a_2^{max})$ and the game is nondegenerate, so $V(\beta''_1) = u_1(\beta''_1, a_2^{max})$. Because $\beta''_1 \in W$, it is the case that $u_1(\beta''_1, a_2^{max}) > u_1(\beta_1, \beta_2)$. Thus, in this case $V(\beta''_1)$ is strictly greater than the minimum Nash equilibrium value.

■

Proof (of Theorem 7.8): The above two lemmas can be combined into a proof of Theorem 2. First, calculate the minimum Nash equilibrium value, and choose a β_1 such that $V(\beta_1)$ equals or exceeds this value, and have the first agent play β_1 every time step. Then, the first agent will achieve $V(\beta_1)$ if the second agent achieves no-external-regret. Since $V(\beta_1)$ is greater than or equal to the minimum Nash equilibrium value, the result directly follows. ■

7.4 The Proof Sketch of Theorem 7.9

Now, first we fix σ_1^* to be an arbitrary no-external-regret algorithm for the first agent. We construct a no-external-regret algorithm σ_2^* such that σ_1^* does not achieve the correlated equilibrium value against σ_2^*. We will first construct a deterministic algorithm σ_2 such that when the first agent uses σ_1^* and the second agent uses σ_2, the first agent will have a low average utility, below its correlated equilibrium value. σ_2 will not achieve no-external-regret, but if it is playing σ_1^* almost surely it will eventually have low regret. We can do this because we have chosen σ_2 specifically for σ_1^*, and so σ_2 can be considered to know the strategy of σ_1^* before each time step. Then, we construct σ_2^* from σ_2, where with high probability σ_2^* always "agrees" with σ_2 if it is playing against σ_1^*, but σ_2^* also achieves no-external-regret.

In Section 7.4.1, we define the game. In Section 7.4.2, we define σ_2, and establish that it has low regret with high probability against σ_1^*, and σ_1^* has low value against σ_2 with high probability. In Section 7.4.3, we extend σ_2 to σ_2^*.

7.4.1 The Difficult Game

The game we design will have three pairs of actions for the second agent. HCA and HCB with A and B form a coordination game. LCA and LCB are strictly dominated by HCA and HCB respectively, and also form a coordination game with A and B. LZA and LZB are also strictly dominated by HCA and HCB

[8]However, if the game is truly nondegenerate, this will only hold if A'_2 is a singleton.

respectively, and form a constant-sum game[9] with A and B. Thus, the second agent has three possible games to choose from. Here we make a rough argument for why the game has the properties it has: in the next section, we show how σ_2 uses these properties.

The idea behind σ_2 is simple. We want to play strictly dominated strategies. By playing strictly dominated strategies, we can make σ_1^* receive a utility that is not at all related to its value at a correlated equilibrium.

What we must not allow is for the first agent to simply play its part of a fixed Nash equilibrium, like in the last section. Thus, alternately we will lead the first agent to be more inclined to play one action, and then another, keeping it away from playing its part of any single Nash equilibrium all the time.

Now, because σ_2 is designed with a specific σ_1^* in mind, we can "predict" the distribution of the first agent on the next time step. Thus, we will play an action that is almost a best response to the distribution of the first agent on every time step.

The game is as follows:

Game 7.17: Attractive Dominated Strategies

	HCA	HCB	LCA	LCB	LZA	LZB
A	$1+g_1$,$1+g_2$	$0+g_1$,$0+g_2$	1,1	0,0	1,0	0,1
B	$0+g_1$,$0+g_2$	$1+g_1$,$1+g_2$	0,0	1,1	0,1	1,0

We will set $g_1 = 1000$ and $g_2 = \frac{1}{512}$. The lowest-valued Nash equilibria (or correlated equilibrium) for the first agent is when the first agent plays A and B with equal probability, and the second agent plays HCA and HCB with equal probability. This Nash equilibrium has a utility of $g_1 + 1/2$ for the first agent. σ_2 will be constructed to only use the actions LCA, LCB, LZA, and LZB. The first agent will never see a value higher than 1, and the difference between its value obtained and its value at a correlated equilibrium is at least $g_1 - \frac{1}{2} = 999.5$. We choose $g_2 = \frac{1}{512}$, sufficiently small such that the second agent does not regret too much not playing HCA and HCB.

7.4.2 The Deterministic Algorithm σ_2

The deterministic algorithm σ_2 works in "phases", labeled alternately A and B. Each phase lasts 8 times the length of the last phase. During an A phase, σ_2 uses LCA and LZA, and the first agent is motivated to play A. During a B phase, σ_2 uses LCB and LZB, and the first agent is motivated to play B.

The following chart represents the σ_2 strategy:

	σ_1 more likely[10] to play x next time step	
	$x = A$	$x = B$
During an A Phase	LCA	LZA
During a B Phase	LZB	LCB

So, when σ_1 is more likely to play A, while HCA is a best response, LCA and LZA achieve in expectation only $g_2 = \frac{1}{512}$ less than HCA.

For the first agent, during an A phase it is a best response to play A, and during a B phase it is a best response to play B. Because the phases are so long, any σ_1 will eventually find itself playing A more than B in an A phase, and B more than A in a B phase.

To summarize, σ_2 is using strictly dominated actions, so that the first agent's utilities are completely unrelated to its minimum correlated equilibrium value. However, σ_2 obtains a bit of regret from using strictly dominated actions. σ_2 needs to do almost as well as the best *static* action in order to minimize its regret. It achieves this by forcing σ_1 to be highly *dynamic*, and then exploiting this dynamic to compensate for the regret caused by playing dominated strategies. We will find that eventually, when the regret of σ_1 falls below $\frac{1}{1038}$, we can do strictly better than the best fixed action, achieving a regret less than zero.

[9] The sum of the utilities for any outcome is 1.
[10] If equally likely, go with A.

7.4.3 Making σ_2 No-Regret

We will now construct σ_2^*. We will need σ_2, and some no-external-regret algorithm σ_2'. σ_2^* will begin by playing exactly like σ_2. After some fixed time step t_0, it will begin to watch its regret. If at any point in time after time step t_0, the regret of σ_2^* goes above zero, it switches to using σ_2', feeding it the actions as though the game had just begun. It is not hard to argue that it achieves no-external-regret.

If σ_2^* is playing σ_1^*, then after some time t, with probability p the regret is always less than zero. If t_0 is sufficiently large, the chance that σ_2^* will ever switch to σ_2' is small, so with a high probability σ_2^* never plays HCA or HCB if it is against σ_1^*, causing σ_1^* to not get its minimum Nash equilibrium value.

7.5 Playing against a No-Internal-Regret Algorithm

Can the first agent achieve the minimum Nash equilibrium value if the second agent achieves no-internal-regret, even if the first agent does not know the utility function of the second agent? If the game is nondegenerate, then this can be achieved. The trick is to simultaneously achieve the value of some Nash equilibrium while at the same time confirming that it is a Nash equilibrium.

The fundamental difficulty we face here is that even if the other agent achieves no-internal-regret, we don't know how fast it is minimizing internal regret, because achieving no-internal-regret is an asymptotic property. Thus, we will never know if the actions we have observed in the past are representative of the other agent's utility or just arbitrary actions.

However, we can resolve this issue with the knowledge that almost certainly, eventually the actions of the other agent represent its utility. Thus, while we cannot have even a probabilistic faith in a test that is performed once at any time, if a test is performed an infinite number of times, almost surely it will be accurate.

7.5.1 Testing a Pure Strategy Nash Equilibrium

For example, consider testing if there is a pure Nash equilibrium where you, the first agent, play a_1 and the second agent plays b_1. If this is the case, because the game is nondegenerate, b_1 will be the only best response to a_1. Suppose it is currently time step 100, and you have never played a_1. First, make a guess: "If I play a_1 for $9,900$ out of $10,000$ time steps, then I expect the other agent to play b_1 at least half the time." So, if b_1 is a best response to a_1, and the second agent is minimizing its regret fast, then it will play b_1 at least half the time. Then, so long as the other agent is playing b_1 half the time, you can continue to play a_1. Eventually, the second agent will play b_1 more and more often.

However, if some other action b_2 is really a best response to a_1, then the other agent will eventually start playing b_2 more often, and you will realize that your assumption about the existence of a Nash equilibrium there might be wrong.

Of course, that is not the only reason your test failed. For instance, playing a_1 99 out of 100 times may not have b_1 as a best response, even if playing a_1 does. Also, the second agent's regret just might not be that low yet. Thus, later on, if no other pure or mixed equilibria have been found, you should try this equilibrium again, and play a_1 999 out of 1000 times before you move on to test another equilibrium. If a_1 and b_1 form an equilibrium in a nondegenerate game, then there is some $p < 1$ such that b_1 is a best response to playing a_1 a p fraction of the time, and anything else a fraction $1-p$ of the time. Thus, we can test for pure strategy equilibria and play a hypothetical pure strategy Nash equilibrium while continuing to test it.

7.5.2 Testing a Mixed Strategy Nash Equilibrium

Here, we will briefly discuss testing if there exists a mixed strategy Nash equilibrium in a region. First, we choose some sufficiently small region $R \subseteq \Delta(A_1)$, where all distributions in R have the same support A_1', and guess that there is a Nash equilibrium in R. We will specifically avoid any strategy $\beta_1 \in \Delta(A_1)$ where there exists $a_1, a_1' \in A_1$ such that $u_1(\beta_1, a_1) = u_1(\beta_1, a_1')$, because we have excluded such strategies from being part of a Nash equilibrium in a nondegenerate game. We will test if, in this region, there is a best response with some support $A_2' \subseteq A_2$. Observe that since we know our own utilities, this implies a specific distribution β_2 for the other agent. We find an action $a_2^{max} \in A_2'$ such that

$$\min_{\beta_1 \in R} u_1(\beta_1, a_2^{max}) > \max_{\beta_1 \in R} u_1(\beta_1, \beta_2) \ .$$

For a sufficiently small R, this action will exist. We also make rough guesses about where the region $BR^{-1}(a_2)$ is for each of the actions in A_2'. The majority of the time we play a strategy in the interior of where we think $BR^{-1}(a_2^{max})$ is.

Because when the other agent plays a_2^{max} we get strictly more than the minimum Nash equilibrium value, we do not always have to guarantee getting the minimum Nash equilibrium value every time step. We can sometimes spend short periods playing strategies to see how the other agent responds. Using these short tests, we can verify if our assumption about the existence of a Nash equilibrium in the region $R \times \{\beta_2\}$ is correct.

7.6 Related Work

In the literature, no-external-regret algorithms are sometimes called universally consistent behaviors. Many no-external-regret algorithms have been invented and studied[FV97, FV99, FS99, FL95, FL98, FL99, Han57, HMC00, HMC01a]. The two works most closely related to this chapter are [FL95] and [FV97], in which they develop lower bounds instead of upper bounds on the value. In [FL95], they introduce universal consistency (no-regret) and prove that algorithms that are no-regret are **safe**, achieving the minimax value. In [FL98], the definition of no-regret behaviors is strengthened. In [FV97], they prove that conditionally universally consistent (no-internal-regret) algorithms converge to correlated equilibria, and thus achieve the minimum value of any correlated equilibrium.

There have been many studies about how Nash equilibria can and cannot be reached in certain circumstances. Fictitious play [Rob51, Bro51] is probably the earliest example where on each time step, each agent plays a best response to the current empirical frequencies played by the others. Shapley [Sha64] showed how the empirical distributions did not converge to a Nash equilibria in a specific game. Kalai and Lehrer [KL93] prove that one can approach within ϵ of an ϵ-Nash equilibrium of the repeated game with Bayesian updates from certain priors, even if the priors are not common. However, there has been a great deal of research demonstrating that constructing these priors is not trivial [Nac97, MS99, FY01]. Another recent study is [FY02], where the behaviors use near-optimal strategies in response to some “hypothesis” on the distribution over the next joint probability distribution of the agents given the past m joint actions. These algorithms with a probability greater than $1 - \epsilon$ will play ϵ-equilibria after some time t. Neither of these methods are in general no-regret, but either might be modified in some fashion to make them so.

Another related area in game theory is bounded rationality, where the agents are not assumed to have an accurate model of the world or even to be capable of forming one. In this field, various types of assumptions are made about the memory and computational resources of the agents [Rub98]. In this chapter, we are making restrictions on the goals of the agents as opposed to their computational capabilities. Also, we compare the performance of the algorithm to the utility obtained in a traditional Nash equilibrium, as opposed to equilibria in restricted strategy spaces.

A final type of analysis is analyzing and experimenting with self-play [SKM00, BV01, CS03]. This is a good way to develop behaviors that converge; however any behaviors developed in such an environment should be analyzed when interacting with other established classes of behaviors. Also, restraint is required in developing such algorithms, because one can easily develop behaviors that collaborate in a very unnatural fashion.

7.7 Conclusion

In this chapter, we have studied what utility can be achieved if the other agent achieves no-external-regret. External regret is a natural assumption to make, being a property that has been independently discovered by

	First Agent Arbitrary	First Agent No-External-Regret	First Agent No-Internal-Regret
Second Agent is Arbitrary	CAN: Safe Value. CAN'T: MCEV, MNEV.	CAN: Safe Value. CAN'T: MCEV, MNEV.	CAN: Safe Value. CAN'T: MCEV, MNEV.
Second Agent No-External-Regret	CAN: Safe Value, MCEV, MNEV.	CAN: Safe Value. CAN'T: MCEV, MNEV.	CAN: Safe Value. CAN'T: MCEV, MNEV.
Second Agent No-Internal-Regret	CAN: Safe Value, MCEV, MNEV.	CAN: Safe Value, MCEV. UNKNOWN: MNEV.	CAN: Safe Value, MCEV. UNKNOWN: MNEV.
Second Agent No-Internal-Regret (Utilities unknown)	CAN: Safe Value, MCEV, MNEV.	CAN: Safe Value, MCEV. UNKNOWN: MNEV.	CAN: Safe Value, MCEV. UNKNOWN: MNEV.

Figure 7.2: Summary of results. In each column is a class A of behaviors for the first agent. In each row is a class B of behaviors for the second agent (the fourth row contains results for the case where the first agent does not know the utilities of the second agent). In each cell is listed the utilities that can and cannot be achieved by a first agent from the class A against every behavior for the second agent from the class B: safe value, minimum correlated equilibrium value (MCEV), or minimum Nash equilibrium value (MNEV). CAN MNEV indicates there exists a behavior σ_1 in A, such that for every behavior σ_2 in B, σ_1 achieves the minimum Nash equilibrium value against σ_2. CAN'T MNEV indicates for every behavior σ_1 in A there exists a behavior σ_2 in B such that σ_1 does not achieve the minimum Nash equilibrium value against σ_2. UNKNOWN indicates an open problem.

many people [Han57, Meg80, FL95, LW89]. Here we show that if one assumes the other agent is no-external-regret, then one can easily achieve the minimum Nash equilibrium value. However, this result is unsatisfying, because the algorithm we design that achieves the minimum Nash equilibrium value does not itself achieve no-external-regret. In fact, we show that there does not exist a no-external-regret algorithm that can achieve the minimum correlated equilibrium value against an arbitrary no-external-regret algorithm.

The reason that achieving the minimum Nash equilibrium value is an interesting goal is that it can be significantly higher than the values that natural external regret minimizing algorithms get in some games. Philosophically, there is a difference between a correlated equilibrium that two no-internal-regret algorithms converge to and a correlated equilibrium that two humans agree upon as a self-enforcing contract. It is unlikely that two humans would ever agree upon a correlated equilibrium where both agents receive relatively low utility. But, because the convergence of two no-internal-regret algorithms to the set of correlated equilibria is an emergent behavior instead of a conscious choice, it is not surprising that they might choose to play a correlated equilibrium that is not a good one.

Chapter 8

Future Work

Below are some of the interesting directions this research can be taken.

- Does there exist an incentive-compatible online mechanism for maximizing profit in the temporal bidding problem that achieves at least a log factor of the profit of the optimal offline mechanism?
- In preference elicitation, one can consider the related problem of preference verification: verifying that a given allocation is optimal. If one imagines that the preferences are *strictly increasing*, i.e. giving one more item to someone always increases his or her utility, how are the number of value queries required to compute the optimal allocation related to the number of value queries required to verify the optimal allocation? Is the competitive ratio polynomial in the number of items?
- In online convex programming, there have been several results that have extended gradient descent to situations where the entire cost function is unknown, and only the algorithm's cost is observed. However, does there exist an algorithm that can minimize the cost with high probability against an adaptive adversary?
- Minimizing response regret can be very useful if one believes that one's actions have a short-term effect on the behavior of the other agent or the state of the environment. However, in practice response regret may take a long time to minimize. What are ways of relaxing response regret so that it can be minimized more efficiently? Particularly, if one has ideas about what behaviors may be useful in an environment, can one use this information to modify the concept of response regret?
- In this thesis, we have shown that assuming the other agent achieves no-external-regret is not a very useful way of modeling an intelligent agent. Can one algorithm both achieve no-internal-regret against an arbitrary algorithm and achieve a minimum Nash equilibrium value assuming the other agent achieves no-internal-regret? In general, is there any natural way to represent the belief that the other agent might possibly be intelligent?

We have studied how one can perform well with incomplete information in the presence of other agents. We have studied both how one can design an environment and how one can design an agent in an existing environment. In mechanism design, we assume the agents are rational, that they have accurate beliefs about the world and act optimally based on those beliefs. In an arbitrary game, such an assumption may be less justified, because there may be no a priori optimal plan of action. If we cannot assume the other agents are rational, then what assumptions should we make about them?

In most Bayesian frameworks (like finite domains or discounted Markov decision processes), what should or should not be done is obvious from the beliefs that the agents have (i.e., the problem is more computational than philosophical). In a way, a measure of regret is like a belief about what basic strategies might be useful and how they could be combined to achieve a high utility. To discover the relationships between regret and equilibria, it is first necessary to use regret as a belief, something that can justify each distribution over actions chosen during an infinitely repeated game. As researchers, we should focus both on developing new and more useful measures of regret, as well as being more restrictive with regards to judging whether a behavior is really justified by a measure of regret. The final goal should be a concept of regret such that all behaviors justified by that concept of regret converge to the convex hull of the set of Nash equilibria.

Bibliography

[ABF96] Baruch Awerbuch, Yair Bartal, and Amos Fiat. Distributed paging for general networks. In *Proc. 7th Symp. on Discrete Algorithms*, pages 574–583, 1996.

[ACBFS02] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[AHK93] D. Angluin, L. Hellerstein, and M. Karpinski. Learning read-once formulas with queries. *Journal of the ACM*, 40:185–210, 1993.

[Ama98] S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10:251–276, 1998.

[Ang88] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

[BB00] A. Blum and C. Burch. On-line learning and the metrical task system problem. *Machine Learning*, 39(1):35–58, April 2000.

[BBCM03] N. Bansal, A. Blum, S. Chawla, and A. Meyerson. Online oblivious routing. Submitted, 2003.

[BCG$^+$01] A. Bagchi, A. Chaudhary, R. Garg, M. Goodrich, and V. Kumar. Seller-focused algorithms for online auctioning. In *Proceedings of WADS 2001*, pages 135–147, 2001.

[BdVSV01] Sushil Bikhchandani, Sven de Vries, James Schummer, and Rakesh V. Vohra. Linear programming and Vickrey auctions, 2001. Draft.

[Ber57] C. Berge. Two theorems in graph theory. In *Proceedings of the National Academy of Sciences*, volume 43, pages 842–844, 1957.

[BEY98] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[BGM04] Y. Bartal, R. Gonen, and P. La Mura. Negotiation-range mechanisms: Exploring the limits of truthful efficient markets. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, pages 1–8, 2004.

[BHH95] N. H. Bshouty, T. R. Hancock, and L. Hellerstein. Learning arithmetic read-once formulas. *SIAM Journal on Computing*, 24(4):706–735, 1995.

[BHHK94] N. Bshouty, T. Hancock, L. Hellerstein, and M. Karpinski. An algorithm to learn read-once threshold formulas, and transformations between learning models. *Computational Complexity*, 4:37–61, 1994.

[BKRW03] A. Blum, V. Kumar, A. Rudra, and F. Wu. Online learning in online auctions. In *Proceedings of the Symposium on Discrete Algorithms*, pages 202–204, 2003.

[Bla56] D. Blackwell. An analog of the minimax theorem for vector payoffs. *Pacific J. of Mathematics*, pages 1–8, 1956.

[BM03] D. Bergemann and R. Morris. Robust mechanism design. Technical Report working paper, Yale, 2003.

[BM04] A. Blum and H. McMahan. Online geometric optimization in the bandit setting against an adaptive adversary. In *Proceedings of the Seventeenth Annual Conference on Computational Learning Theory*, 2004.

[Bro51] G. Brown. Iterative solutions of games by fictitious play. In T. C. Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 374–376. Wiley, 1951.

[BSZ02] A. Blum, T. Sandholm, and M. Zinkevich. Online algorithms for market clearing. In *Proceedings of the Thirteenth Annual Symposium on Discrete Algorithms*, pages 971–980, 2002.

[BT02] R. Brafman and M. Tennenholtz. Efficient learning equilibrium. In *Advances in Neural Information Processing Systems 17*, 2002.

[BV01] M. Bowling and M. Veloso. Convergence of gradient dynamics with a variable learning rate. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 27–34, 2001.

[CBLW94] N. Cesa-Bianchi, P. Long, and M. K. Warmuth. Worst-case quadratic bounds for on-line prediction of linear functions by gradient descent. *IEEE Transactions on Neural Networks*, 7:604–619, 1994.

[CS01] Wolfram Conen and Tuomas Sandholm. Preference elicitation in combinatorial auctions: Extended abstract. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 256–259, Tampa, FL, October 2001. A more detailed description of the algorithmic aspects appeared in the IJCAI-2001 Workshop on Economic Agents, Models, and Mechanisms, pp. 71–80.

[CS02a] Wolfram Conen and Tuomas Sandholm. Differential-revelation VCG mechanisms for combinatorial auctions. In *AAMAS-02 workshop on Agent-Mediated Electronic Commerce (AMEC)*, pages 196–197, Bologna, Italy, 2002.

[CS02b] Wolfram Conen and Tuomas Sandholm. Partial-revelation VCG mechanism for combinatorial auctions. In *Proceedings of the National Conference on Artificial Intelligence*, pages 367–372, Edmonton, Canada, 2002.

[CS03] V. Conitzer and T. Sandholm. Awesome: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. In *Proceedings of the International Conference on Machine Learning*, 2003.

[DDL99] S. Della Pietra, V. Della Pietra, and J. Lafferty. Duality and auxilary functions for Bregman distances. Technical Report CMU-CS-01-109, Carnegie Mellon University, 1999.

[dFM03] D. de Farias and N. Meggido. How to combine expert (or novice) advice when actions impact the environment. In *Advances in Neural Information Processing Systems 17*, 2003.

[Dom93] Ian Domowitz. Automating the continuous double auction in practice: Automated trade execution systems in financial markets. In Daniel Friedman and John Rust, editors, *The Double Auction Market*, volume 14 of *Santa Fe Institute Studies in the Sciences of Complexity*, pages 27–60. Addison-Wesley, 1993.

[EYFKT92] Ran El-Yaniv, Amos Fiat, Richard M. Karp, and G. Turpin. Competitive analysis of financial games. In *Proc. 33rd Symp. Foundations of Computer Science*, pages 327–333, 1992.

[FKM04] A. Flaxman, A. Kalai, and H. McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. Manuscript, July 2004.

[FL95] D. Fudenberg and D. Levine. Universal consistency and cautious fictitious play. *Journal of Economic Dynamics and Control*, 19:1065–1089, 1995.

[FL98] D. Fudenberg and D. Levine. *The Theory of Learning in Games*. MIT Press, 1998.

[FL99] D. Fudenberg and D. Levine. Conditional universal consistency. *Games and Economic Behavior*, 29, 1999.

[Fos99] D. Foster. A proof of calibration via Blackwell's approachability theorem. *Games and Economic Behavior*, 29:73–79, 1999.

[FP02] E. Friedman and D. Parkes. Pricing wifi at starbucks. Working paper, November 2002.

[FS96] Yoav Freund and Robert Schapire. Game theory, on-line prediction and boosting. In *Proceedings of the Ninth Conference on Computational Learning Theory*, pages 325–332, 1996.

[FS99] Y. Freund and R. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29:79–103, 1999.

[FV97] D. Foster and R. Vohra. Calibrated learning and correlated equilibrium. *Games and Economic Behavior*, 21(1):40–55, 1997.

[FV99] D. Foster and R. Vohra. Regret in the on-line decision problem. *Games and Economic Behavior*, 29(1):7–35, 1999.

[FY01] D. Foster and H. Young. On the impossibility of predicting the behavior of rational agents. *Proceedings of the National Academy of Sciences*, 98(22):12848–12853, 2001.

[FY02] D. Foster and H. Young. Learning, hypothesis testing, and Nash equilibrium. Available at http://www.cis.upenn.edu/˜foster/research/, 2002.

[GHW00] Andrew Goldberg, J Hartline, and A Wright. Competitive auctions and multiple digital goods. Technical report, InterTrust 00-01, 2000.

[GHW01] Andrew Goldberg, J Hartline, and A Wright. Competitive auctions and digital goods. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Washington, DC, 2001.

[GK91] S. Goldman and M. Kearns. On the complexity of teaching. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 303–314, 1991.

[GPS00] S. Goldman, J. Parwatikar, and S. Suri. On-line scheduling with hard deadlines. *Journal of Algorithms*, 34:370–389, 2000.

[GW00] C. Gentile and M. Warmuth. Proving relative loss bounds for online learning algorithms by the Bregman divergence. In *The 13th Annual Conference on Computational Learning Theory*, June 2000. Tutorial.

[Han57] J. Hannan. Approximation to Bayes risk in repeated play. *Annals of Mathematics Studies*, 39:97–139, 1957.

[Har03] J. Hartline. *Optimization in the Private Value Model: Competitive Analysis Applied to Auction Design*. PhD thesis, University of Washington, 2003.

[HMC00] S. Hart and A. Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68:1127–1150, 2000.

[HMC01a] S. Hart and A. Mas-Colell. A general class of adaptive strategies. *Journal of Economic Theory*, 98:26–54, 2001.

[HMC01b] S. Hart and A. Mas-Colell. A reinforcement procedure leading to correlated equilibrium. In G. Debreu, W. Neuefeind, and W. Trockel, editors, *Economic Essays*, pages 181–200. Springer, 2001.

[HMC03] S. Hart and A. Mas-Colell. Uncoupled dynamics cannot lead to Nash equilibrium. *American Economic Review*, 68(5):1830–1836, 2003.

[Hoe63] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, March 1963.

[HS02] Benoit Hudson and Tuomas Sandholm. Effectiveness of preference elicitation in combinatorial auctions. In *AAMAS-02 workshop on Agent-Mediated Electronic Commerce (AMEC)*, pages 69–86, Bologna, Italy, 2002. Extended version: Carnegie Mellon University, Computer Science Department, CMU-CS-02-124, March. Also: Stanford Institute for Theoretical Economics workshop (SITE-02).

[HS04] B. Hudson and T. Sandholm. Effectiveness of query types and policies for preference elicitation in combinatorial auctions. In *Proceedings of Autonomous Agents and Multiagent Systems*, 2004.

[HW01] M. Herbster and M. K. Warmuth. Tracking the best linear predictor. *Journal of Machine Learning Research*, 1:281–309, 2001.

[KL93] E. Kalai and E. Lehrer. Rational learning leads to Nash equilibrium. *Econometrica*, 61(5):1019–1045, 1993.

[KL04] R. Kleinberg and T. Leighton. The value of knowing a demand curve: Bounds on regret for online posted-price auctions. In *Proceedings of the 44th Symposium on the Foundations of Computer Science*, 2004.

[Koz91] D. Kozen. *The Design and Analysis of Algorithms*. Springer-Verlag, New York, 1991.

[KT99] Ming-Yang Kao and Stephen R. Tate. On-line difference maximization. *SIAM Journal of Discrete Mathematics*, 12(1):78–90, 1999.

[KV02] A. Kalai and S. Vempala. Geometric algorithms for online optimization. Technical report, MIT, 2002.

[KVV90] Richard M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the 22nd Symposium on Theory of Computing*, pages 352–358, Baltimore, Maryland, 1990.

[KW97] J. Kivinen and M. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132:1–64, 1997.

[KW01] J. Kivinen and M. Warmuth. Relative loss bounds for multidimensional regression problems. *Machine Learning Journal*, 45:301–329, 2001.

[LN04] R. Lavi and N. Nisan. Competitive analysis of incentive compatible on-line auctions. *Theoretical Computer Science*, 310(1):159–180, 2004. A preliminary version appeared in *ACM EC '00*, October 2000.

[LT94] R. J. Lipton and A. Tomkins. Online interval scheduling. In *Proceedings of the Fifth Symposium on Discrete Algorithms*, pages 302–311, 1994.

[LW89] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. In *Proceedings of the Second Annual Conference on Computational Learning Theory*, pages 256–261, 1989.

[LW94] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, 1994.

[Meg80] N. Meggido. On repeated games with incomplete information played by non-bayesian players. *International Journal of Game Theory*, 9:157–167, 1980.

[MS83] R. Myerson and M. Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 29:265–281, 1983.

[MS99] R. Miller and C. Sanchirico. The role of absolute continuity in "merging of opinions" and "rational learning". *Games and Economic Behavior*, 29:170–190, 1999.

[MW01] R. Mahony and R. Williamson. Prior knowledge and preferential structures in gradient descent algorithms. *Journal of Machine Learning Research*, 1:311–355, 2001.

[Mye81] R. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6:58–73, 1981.

[Nac97] J. Nachbar. Prediction, optimization, and learning in games. *Econometrica*, 65(2):275–309, 1997.

[Nas50] J. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36:48–49, 1950.

[NS02] Noam Nisan and Ilya Segal. The communication complexity of efficient allocation problems, 2002. Draft. Second version March 5th.

[OR94] M. Osborne and A. Rubinstein. *A Course in Game Theory.* The MIT Press, 1994.

[Por04] R. Porter. Mechanism design for online real-time scheduling. In *Fifth ACM Conference on E-Commerce*, 2004.

[PU00] David C Parkes and Lyle Ungar. Iterative combinatorial auctions: Theory and practice. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 74–81, Austin, TX, August 2000.

[Rob51] J. Robinson. An iterative method of solving a game. *Annals of Mathematics*, 54:296–301, 1951.

[Rub98] A. Rubinstein. *Modeling Bounded Rationality.* The MIT Press, 1998.

[San02] Tuomas Sandholm. eMediator: A next generation electronic commerce server. *Computational Intelligence*, 2002. Special issue on Agent Technology for Electronic Commerce. To appear. Early versions appeared in the Conference on Autonomous Agents (AGENTS-00), pp. 73–96, 2000; AAAI-99 Workshop on AI in Electronic Commerce, Orlando, FL, pp. 46–55, July 1999; and as a Washington University, St. Louis, Dept. of Computer Science technical report WU-CS-99-02, Jan. 1999.

[Sha53] L. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences of the United States of America*, 39:1095–1100, 1953.

[Sha64] L. Shapley. Some topics in two-person games. In M. Dresher, L. Shapley, and A. Tucker, editors, *Advances in Game Theory*, pages 1–28. Princeton University Press, 1964.

[Sha74] L. Shapley. A note on the Lemke-Howson method. In M. Balinski, editor, *Mathematical Programming Study 1: Pivoting and Extensions*, pages 175–189. North-Holland Publishing Company, Amsterdam, November 1974.

[SKM00] S. Singh, M. Kearns, and Y. Mansour. Nash convergence of gradient dynamics in general-sum games. In *Proceedings of the Sixteenth Conference in Uncertainty in Artificial Intelligence*, pages 541–548, 2000.

[SS00] Tuomas Sandholm and Subhash Suri. Improved algorithms for optimal winner determination in combinatorial auctions and generalizations. In *Proceedings of the National Conference on Artificial Intelligence*, pages 90–97, Austin, TX, 2000. Extended version with proofs at www.cs.cmu.edu/˜sandholm/bidbranching.aaai00.extended.ps.

[SS01] Tuomas Sandholm and Subhash Suri. Market clearability. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1145–1151, Seattle, WA, 2001.

[SS02] Tuomas Sandholm and Subhash Suri. Optimal clearing of supply/demand curves. In *AAAI-02 workshop on Agent-Based Technologies for B2B Electronic Commerce*, Edmonton, Canada, 2002.

[SSGL01] Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. CABOB: A fast optimal algorithm for combinatorial auctions. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1102–1108, Seattle, WA, 2001.

[SSGL02] Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. Winner determination in combinatorial auction generalizations. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Bologna, Italy, July 2002. Early version appeared at the AGENTS-01 Workshop on Agent-Based Approaches to B2B, pp. 35–41, Montreal, Canada, May 2001.

[Wil85] R. Wilson. Incentive efficiency of double auctions. *Econometrica*, 53:1101–1116, 1985.

[Wil87] R. Wilson. Game-theoretic approaches to trading-processes. In T. Bewley, editor, *Advances in Economic Theory: Fifth World Congress*, Cambridge, 1987. Cambridge University Press.

[WW00] Peter R Wurman and Michael P Wellman. AkBA: A progressive, anonymous-price combinatorial auction. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 21–29, Minneapolis, MN, October 2000.

[WWW98] Peter R Wurman, Michael P Wellman, and William E Walsh. The Michigan Internet AuctionBot: A configurable auction server for human and software agents. In *Proceedings of the Second International Conference on Autonomous Agents (AGENTS)*, pages 301–308, Minneapolis/St. Paul, MN, May 1998.

[ZBS03] M. Zinkevich, A. Blum, and T. Sandholm. On polynomial-time preference elicitation with value queries. In *Proceedings of the Fourth ACM Conference on Electronic Commerce*, 2003.

[Zin03] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.