

Repeated Mechanism Design

Exploring the use of learning against evolving agent populations

Gerrit Jan van Ahee

Repeated Mechanism Design

RESEARCH ASSIGNMENT

by

Gerrit Jan van Ahee
born in Zwolle, the Netherlands



Software Engineering Research Group
Department of Software Technology
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
www.ewi.tudelft.nl

Repeated Mechanism Design

Author: Gerrit Jan van Ahee
Student id: 1000799
Email: van.ahee@yes2web.nl

Abstract

Mechanism design has become a very popular topic among computer scientist over the past fifteen years. The selection of the right incentives to provide to agents in order to motivate them to take desired actions has proved quite a challenge. The incentives are carefully designed to persuade rational agents, i.e. agents acting in their own best interest, their utility. On the other side of the same coin, multiagent learning has drawn a lot of attention, allowing agents to learn how to use these incentives to improve their utility. Traditionally, mechanism designers target static agent populations: populations where the agents' characteristics are constant over time in expectation. This assumption may not hold, especially in mechanisms that are to be repeated and agents learning to cope with the incentives. In this work, we investigate a generic way of creating an adaptive mechanism, that will be able to adjust the rules and incentives according to the population of agents it encounters. To do this, the notion of a meta-game is introduced: a higher level game, where the mechanism as a player plays against the population of agents. The mechanism player in the meta-game will use learning techniques from the field of multiagent learning that are also available to the agents acting in the adaptive mechanism itself.

Thesis Committee:

Chair: Prof. Dr. C. Witteveen, Faculty EEMCS, TU Delft
University supervisor: Dr. T.B. Klos, Faculty EEMCS, TU Delft

Contents

| | |
|--|------------|
| Contents | iii |
| List of Figures | v |
| 1 Introduction | 1 |
| 1.1 Multiagent Systems | 1 |
| 1.2 Goals and Applications | 2 |
| 1.3 Structure of this Document | 4 |
| 2 Game Theory | 5 |
| 2.1 Games | 5 |
| 2.2 Equilibria | 7 |
| 2.3 Repeated Games | 9 |
| 2.4 Bayesian Games | 11 |
| 3 Mechanism Design | 15 |
| 3.1 Concepts of Mechanisms | 15 |
| 3.2 The Revelation Principle | 16 |
| 4 Learning | 21 |
| 4.1 Reinforcement Learning | 21 |
| 4.2 Evolutionary Game Theory | 25 |
| 5 Adaptive Mechanisms | 31 |
| 5.1 Meta Games | 31 |
| 5.2 Properties of the Players | 34 |
| 6 Future Work | 37 |
| Bibliography | 39 |
| A Referenced Videos | 43 |

List of Figures

| | | |
|-----|--|----|
| 2.1 | The “Ultimatum game” in it’s extensive form | 8 |
| 2.2 | Extensive form of the repeated Prisoners Dilemma. | 10 |
| 2.3 | Information set in the Prisoner’s Dilemma | 11 |
| 4.1 | Unit 3-simplex S_3 with evolutionary trajectories of strategies. | 28 |
| 5.1 | Extensive form of the meta-game. | 32 |

Chapter 1

Introduction

Everywhere around us we see forms of interaction: people driving in traffic, birds fighting over food or teams playing sports. This interaction itself can be studied using the notion of a multiagent system (MAS). A MAS is a collection of autonomous agents and can be extended by having the agents work independently or collectively, supplying agents with more information and the possibility to communicate with other agents.

1.1 Multiagent Systems

Using multiagent systems, all forms of interaction can be modelled, ranging from ants in an ant colony to multinational corporations in economic markets. The systems under our consideration will involve only rational agents, i.e. agents with some form of intelligence, who will behave strategically in order to maximize their individual goals. To quantify this “maximizing of goals” the simplifying assumption of an agent’s *utility* is made. A *utility function* provides a preference order to the states of the world: it now becomes a trivial task to see which state of the system is better to an agent. Rational agents will always attempt to realize a state that maximizes their utility.

Game theory studies those systems in which the agents interact strategically. Multiagent systems with strategic behaviour can be seen as a game, e.g. soccer: twenty-two agents individually and strategically move around and act on a ball in order to achieve the best possible individual outcome. In this case there are two coalitions playing as a single entity (the team), trying to beat the other. With a little imagination, other, more complex systems may also be viewed as games. When determining a pricing strategy, a company will choose the strategy it believes will provoke a potential buyer into buying their product. In this game the company plays both the buyers and its competitors. Voting is another example of an everyday system that may be modelled as a game. Each agent that may be voted for will try to persuade voters using promises and differentiating themselves from their adversaries.

Agents in a game are subject to certain rules and protocols set by the system. Sometimes these rules have emerged over time, such as the ethics in a culture. In most games the rules have been set by a creator. Having rules bounds the possible actions players can choose in any situation. It enables players to envision a sequence of actions to undertake as the game progresses: a strategy. Rules also allow players to

think about the actions their opponents are likely to take, and possibly adapt their own strategy accordingly.

The field of designing rules for games has become a study in its own right: mechanism design. Mechanism designers aim to build complex systems for autonomous agents that, while acting in their individual best interest, will together achieve some global behaviour desired by the designer of the mechanism. This desired behaviour is defined by an *outcome function* and the goal of a game, or mechanism, is the outcome (or outcomes) that maximizes this function. Using the outcome function, outcomes can be measured similar to the way outcomes are measured by agents. By creating laws, law makers in parliament motivate certain behaviour by the people and try to maximize social welfare. However, creating rules is no guarantee that an optimal outcome is actually reached.

In order to get to a goal state, mechanism designers will incorporate incentives in the rules. These incentives will encourage behaviour by the agents, supposedly making their utility higher (lower) if they (do not) end up in a goal state. For example, to discourage speeding, fines have been created to reduce a speeder's utility. This way speeding becomes less preferable and agents would keep to the speed limit. However there may be situations in which an agent may feel the need for speed, and even with the fine his utility will be higher than by taking the "preferred" action. This example shows that providing the appropriate incentives is difficult and they may not always be sufficient to achieve a goal, even in relatively simple situations.

Incentives are carefully crafted to fit the target agent population. They should increase the agent's utility when that agent behaves according to what the designer would like. It is however quite hard to determine an agent's utility function and mechanism designers will have to make assumptions about these functions. In its simplest form, the function merely provides a preference order, which does not say anything about how big an incentive should be for the agent to change its strategy. Often only simple restrictions are placed on this function, e.g. quasi linearity or concavity. In order to get more fitting incentives, more assumptions need to be made about the function.

Some mechanisms are designed to be used and reused, i.e. after reaching a terminal state, the mechanism restarts, possibly with a new agent population. One example is the auctioning of sets of identical goods, e.g. books, in multiple auctions. When an auction is over, the bidders that have obtained a copy will (most likely) not return to a new auction selling the same book. The losing bidders would return, and their numbers will be augmented by new bidders. This new auction can still use the same mechanism as the previous.

1.2 Goals and Applications

We have now seen two potential difficulties: how does the designer accurately make assumptions about the expected agents and do these assumptions remain constant over time? The incentives incorporated in the auction for books are based on the expected behaviour of the bidders; if the assumptions about this behaviour are wrong, the incentives may not work and revenue will decrease. On the other hand, if the assumptions will prove to be right, but the bidder population changes between consecutive runs of the auction, the incentives will again fail to boost the revenue.

A painful example of wrong auction design was the auction of UMTS frequencies in The Netherlands in July 2000 [36]. The Dutch government was selling several spectra after similar auctions had been conducted in the U.K. and later Germany, yielding approximately 38.5 billion and 50 billion euros respectively. The Dutch auction only yielded 2.7 billion. This rather disappointing result is attributed to the wrong choice of auction. Amongst the criticisms were that no attention had been paid to the size of the bidder population¹ or to the fact that newcomers should have a place in the auction.²

At the root of this problem lies a fundamental uncertainty about the behaviour of the encountered agents. Both difficulties mentioned cannot be solved decisively due to the lack of information about the agents. The mechanism can however be designed to provide a best response to the expected population. Currently, most designs use many abstractions, e.g. the shape and properties of the utility function, thereby making the mechanism more generally applicable and less tailored to the population at hand. This is however, what should happen in mechanism design: designers theorize about the agents and create a mechanism to best suit them. When theory has failed during a repeated game, c.f. the book auction, the mechanism must be updated to better suit the bidders. Sometimes, this happens, as in the F.C.C. spectrum auction in the 1990s, where the mechanism was updated between the auctions of different spectra. Unfortunately, this proved a very long and tedious task (c.f. [7, 8]).

The goal of this research is to find a generic way to automate this process. We aim to transform any static mechanism into an adaptive one, providing a best response to the agent populations it has encountered so far. Obviously there is no second chance for a mechanism that is only to be used once, so the focus will be on repeated mechanisms. These systems will be able to *learn* from past experiences to better reply to a new agent population. In earlier work we have provided a proof of this concept, see [14].

There is also an application to non-repeating mechanisms. After selecting some generic mechanism, expectations are formed about the properties of the agents. The selected mechanism will still have properties left to be specified. The optimal values for these could be learned by simulation: the agent population is taken to be constant, i.e. all assumptions are right, and these agents play a repeated game where the mechanism is allowed to update its properties according to its encounters. These simulations will supply a fast means of tweaking the selected mechanism's properties in order to best achieve its desired outcome. This approach has also been investigated by Pardoe et al. [23] and intersects the field of automated mechanism design, c.f. Conitzer [5, chapter 6].

An adaptive mechanism should be able to alter its configuration to better suit the agents in it almost instantly and without the interference of outside experts, as was the case in the F.C.C. auctions. If the mechanism notices a change in agent behaviour, it should itself change to provide a best response to the expected behaviour in the next round of the mechanism. This requires the mechanism to have some form of intelligence and will cause it to act strategically itself. As the problem now moves to a strategic interaction between the mechanism and the agent population as a whole, the problem enters game theory. The notion of a "meta-game" will be used to study this

¹Only six bidders contended five different spectra.

²The U.K. auction provided one part exclusively to a newcomer, thus virtually creating a separate auction just for newcomers for one spectral band.

interaction and the properties of this game will provide useful insights in the design of adaptive mechanisms.

1.3 Structure of this Document

Chapters 2 and 3 will provide a thorough background on the theory of games and mechanism design. Chapter 2 will focus on playing a game with the intention to maximize individual utility, while chapter 3 will discuss the other side: designing rules and incentives to motivate agents into a global desired outcome. To provide the mechanism with the “intelligence” needed to adapt itself to an agent population, three learning techniques from the field of artificial intelligence will be discussed in chapter 4. Chapter 5 will investigate adaptive mechanisms and the properties of the meta-game. These will lead to questions that will need further research and are presented in the final chapter.

In this research, aside from books, journals and proceedings, much information has also been extracted from videos on the internet, such as tech talks provided by Google. These videos will be mentioned occasionally in footnotes and a full reference and description is available in Appendix A.

Chapter 2

Game Theory

Game theory is at the cross section of mathematics and economics. It aims to study situations that involve multiple agents interacting strategically, i.e. agents acting in their individual best interest, and to capture this interaction in a mathematical way. This provides many areas for study, ranging from economics to social sciences and biology, and can be applied to all levels of interaction: small toy games like *tic tac toe*, investment banks bidding for treasury bills and even political candidates competing for votes. Founded by John von Neumann and Oskar Morgenstern to analyse human behaviour in an economic setting in 1944, the conceptually simple and elegant theory of games has attracted many great scientists. A nice and comprehensible introduction to this field is given by Rubinstein¹ and in [2, 22].

This chapter will start with some of the fundamental concepts of game theory, providing toy games to illustrate them. These toy games are all simplifications of many practical economic, social or biological phenomena. Using the concepts several ways of forming a strategy, a sequence of actions, will be discussed. Finally we shall look at what happens to the selection of a strategy when games are repeated and information needed to make this selection is no longer available to all players.

2.1 Games

Agents in a game act strategically in order to realize some outcome of this game that is most beneficial to each of them individually. A game will end in an outcome, based on the individual actions taken by the agents. These actions may be available to all agents, or be individual, just as one company has different resources and budgets as another, and will have different actions at its disposal.

Definition 1 (Game). A game $G = \langle I, O, f, (A_i, \Theta_i, u_i)_{i \in I} \rangle$ consists of a set of agents I with $|I| = N$, a set of outcomes O and an outcome function $f : A \rightarrow O$. For each agent $i \in I$ the game provides a set of actions A_i , with $A = \prod_i A_i$ the joint action space of the player population I , a set of information $\theta_i \in \Theta_i$, and a utility function $u_i : A \rightarrow \mathbb{R}$.

An agent will form a strategy based on the information and actions available to him in the game.

¹Ariel Rubinstein giving a talk at NYU on John Nash. Refer to Appendix A for a description and the location of this video.

Definition 2 (Strategy). A strategy s_i for agent i is a function mapping information to action $s_i : \theta_i \rightarrow A_i$.

Which strategy to play at any time in the game depends on many things: the actions opponents will be playing, any private information an agent has or does not have, all have an influence on the progressing game. Each action taken, by the agent itself or an opponent, may prevent certain outcomes from being reached. This question of which action to select is fundamental in game theory.

To aid in the analysis of a game, it may be visualised as a matrix, also called the *normal form* of a game. Each agent gets its own dimension enumerating the actions available. Each matrix cell corresponds to an outcome and shows the valuations of each of the agents for this outcome. Viewing this matrix from the perspective of a single agent, only his own valuations are shown: this is called the *Payoff matrix* for that agent. As an example we show a simple two player game. The normal form will be a table and the agents will be referred to as Alice (the *row player*) and Bob (the *column player*).² The bottom-left entry in each cell corresponds to the valuation of the row player for this outcome, the top-right entry to the valuation of the column player. The valuation of outcomes can be used as a measure for the utility a player receives.

| | Heads | Tails |
|-------|---------|---------|
| Heads | 1 -1 | -1 1 |
| Tails | -1 1 | 1 -1 |

(a) Matching Pennies

| | Left | Right |
|-------|----------|----------|
| Left | 1 1 | -1 -1 |
| Right | -1 -1 | 1 1 |

(b) Driving Game

Table 2.1: Payoff tables for “Matching Pennies” and “The Driving Game”.

Tables 2.1a and 2.1b show the normal forms for the games “Matching Pennies” and “The Driving Game”. In matching pennies, both players flip a coin. The column player wins when both coins show the same face, otherwise the row player wins. Matching pennies is an example of the original type of game studied by Von Neumann and Morgenstern. It studies pure conflict: the gain of the row player equals the loss of the column player and vice versa. This is called a *zero sum game* as the total gain and loss equal zero.

The driving game is one most of us play every day. Two agents are driving towards each other. Needless to say, both players win when both drive on the right side or both on the left side of the road: it is a game of pure coordination. Because there are outcomes in which both agents gain, this game is not zero sum. The tables illustrate this by the valuations placed on the combinations of actions. Most games are not so easily classified into pure conflict or pure coordination, they will contain elements of both types. Some outcomes will be beneficial, i.e. provide a positive valuation, to both agents, while others may be conflicting. One of the best known games with this property is the Prisoner’s Dilemma, which shall be discussed in the next section.

²Refer to http://en.wikipedia.org/wiki/Alice_and_Bob for more information on Alice and Bob.

2.2 Equilibria

The notion of a *solution concept* is used to predict the way a game will be played. It describes the actions to be taken by each of the players. Reasoning about solutions is very important for an agent to determine which action to take next. The most important class of solution concepts is the *equilibrium concept*, or equilibrium. An equilibrium is a set of strategies that takes into account the actions of other players and within this set, it is irrational for an agent to change his action. This narrows the choice of action for agents, but does not always provide a decisive answer to which action to take.

There are many different equilibrium concepts, most notably the Nash equilibrium (NE). In an NE, no player has anything to gain by *unilaterally* changing its strategy: each action is a best response to the set of opponent actions.

Definition 3 (Nash Equilibrium). *A Nash equilibrium is a set of strategies s^* , one for each player, where no single player can both deviate from this strategy and achieve a higher utility. For each player i and all his strategies $s'_i \neq s_i^*$:*

$$u_i(f(\{s_1^*, \dots, s_i^*, \dots, s_N^*\})) \geq u_i(f(\{s_1^*, \dots, s'_i, \dots, s_N^*\})).$$

A strict Nash equilibrium is a set of strategies s^ with for each player i and all his strategies $s'_i \neq s_i^*$*

$$u_i(f(\{s_1^*, \dots, s_i^*, \dots, s_N^*\})) > u_i(f(\{s_1^*, \dots, s'_i, \dots, s_N^*\})).$$

John Nash proved in 1952 that every game has at least one Nash equilibrium. This may sound good, but unfortunately there are two drawbacks: firstly, there may not be one NE, but multiple. The driving game has two equilibria: left-left and right-right, which to take remains unclear. Secondly, the action to take may not lead to an optimal outcome. This fact is easily shown using a simple two person paradox: the “Prisoner’s Dilemma”, shown in table 2.2.

In the prisoner’s dilemma, Alice and Bob have been apprehended by the police and individually given a choice: cooperate with each other and not talk to the police or defect and confess to the crimes both have committed. Since the police do not have a good case, both players will get off with a very mild conviction if they both cooperate. However, if one tells on the other (defects), (s)he will go free while the other gets a large sentence. If both players talk, they both get half the large sentence. Table 2.2 shows utilities rather than payoffs, i.e. the higher the utility, the lower the sentence and vice versa. The NE will be for both agents to defect, resulting in a utility for each player of 1, while they could have had 5 each. But each agent could then improve its outcome by unilaterally changing its cooperate action to defect, increasing payoff by 5. But when, say, Alice realizes Bob will do so, she knows her utility drops to 0 and also defecting will then increase her utility from 0 to 1: both players end up getting 1. This particular type of strategy is a *dominant strategy*: regardless of what actions other players are playing, this is the most rational strategy to use.

The equilibrium defect-defect consist of a clear directive to the players: defect. This is called a *pure strategy*. It may not be possible to find a pure strategy to play, e.g. in the game of matching pennies shown in table 2.1a. As there is no clue as to what

| | | |
|-----------|-----------|--------|
| | Cooperate | Defect |
| Cooperate | 5, 5 | 0, 10 |
| Defect | 10, 0 | 1, 1 |

Table 2.2: Payoff table for the “Prisoner’s dilemma”.

action the opponent will play, it makes sense to randomize over the possible actions. In this case the (Nash equilibrium) strategy of both agents will be to play 1/2-Heads and 1/2-Tails, i.e. choose arbitrarily between the available options, which yields an expected utility of 1. The probability for selecting either action can be derived from the payoff it leads to. When a strategy involves randomization over possible actions, it is called a *mixed* strategy, and is denoted σ to distinguish it from pure strategies s .

Definition 4 (Mixed Strategy). A mixed strategy σ_i for agent i is a function mapping information to a probability distribution over the set of possible actions $\sigma_i : \theta_i \rightarrow \Delta A_i$.

Games may also be represented as a tree: each leaf represents an outcome and the branches correspond to the possible actions. The paths to these leaves are the actions taken by the agents. This representation is called the *extensive form* of the game. This form is especially convenient for games with sequential behaviour, i.e. when agents take turns in acting. Using backward induction, agents can reason from the leaves, where the payoffs and utilities are known, back towards the root node, being the first action to take. At each internal node, the action leading to the best outcome is remembered. This approach views a subgame at every internal node and selects the best action to take. The solution to this game is called a subgame-perfect equilibrium: in each subgame, the best course of action is literally known. The subgame-perfect equilibrium is a refinement of the Nash equilibrium.

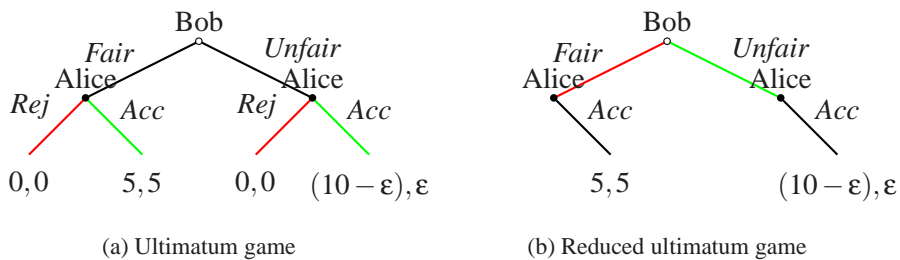


Figure 2.1: The “Ultimatum game” in its extensive form. The right tree shows the situation where Alice has already made up her mind on what action to play.

The “Ultimatum Game” (c.f. figure 2.1a) has two players: Bob and Alice. Bob is given a sum of money to share with Alice. He may propose how to divide it, however, if Alice does not accept his deal, no-one will get any money. In this simplified version Bob has two actions, make a fair or unfair offer. The fair thing to do, would be to

divide the money evenly; alternatively, Bob may offer Alice only an arbitrarily small percentage of the total sum, leaving her with more than she started with, but with a much smaller gain than Bob, i.e. an unfair offer. Alice may accept or reject this offer. She plays two subgames in which she is the only player. In both games she will play “accept”, as she will not get anything if she plays “reject”. One level up in the tree has Bob as a player, but the game has now changed, as playing “accept” is the dominant strategy for Alice, as shown in figure 2.1b. Clearly Bob will now play “unfair”.

2.3 Repeated Games

So far, all games have had a single play, i.e. after fully playing their strategies, the agents get to an outcome, receive their utility and the game ends. This type of game is a *stage game* and several solution concepts have been presented. Things change drastically however when the game under consideration is to be restarted after an outcome is reached: a repeated game.

As with a stage game, agents in a repeated game will try to maximize the total utility they get from playing the game. This time however, utility is the sum of several utilities received from the outcomes of every iteration of the game. It was already mentioned that we play the driving game every day and we want to get maximum utility from it each time. This repetition also changes the way agents arrive at an equilibrium.

To study the new possible solutions to a repeated game, we need to further divide this class of games into *finite* and *infinite* (time) horizon games. This merely states whether the number of repetitions is limited. This may at first seem trivial, but it has severe impact on the solution. With a finite time horizon we may use backward induction over the individual iterations of the game. This induction is based on the assumption that future play will be done rationally. Suppose the prisoners dilemma (see payoff table 2.2) is repeated twice, as shown in the extensive form in figure 2.3. This looks just like any other extensive form game and may be treated as such. The subgame perfect equilibrium solution of this game is easily found, as is indicated by the green edges in figure 2.3. These are the rational choices of the players, the irrational choices have been colored red. The completely green path from root to leaf shows the equilibrium for this repeated game: both players will always defect.

Thus far repeating the game has not proved any more difficult than just playing it once: though the game tree may become too big to use in calculations, theoretically nothing changes. If the horizon is expanded infinitely however, things change dramatically [27, 18, 1], as backward induction is no longer possible as there are no leaves to start reasoning.³

One of the advantages agents have in repeated games is retaliation. Using this, agents may punish opponents that have deviated from some socially optimal strategy. This gives rise to the *trigger strategy*: an agent will initially conform to the social optimum, but switch to a punishing action as soon as the opponent has played an unwanted action too many times.

In the infinitely repeated prisoners dilemma, this provides additional equilibria, besides defect-defect. All of these will have all agents cooperating, as this is the outcome

³The same situation occurs when none of the players know which repetition will be the last.

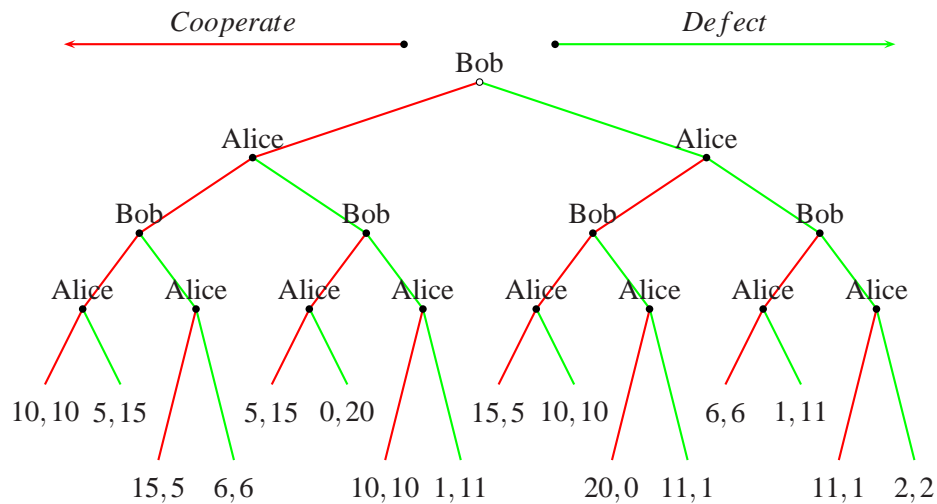


Figure 2.2: Extensive form of the repeated Prisoners Dilemma.

socially, as opposed to individually, favored by both. These new equilibrium strategies work through the fear of retaliation. As soon as Bob defects, thereby increasing his payoff by 5, he will trigger Alice to also defect the next round, and rounds to follow. Bob will have now received more utility than Alice, but less than he would have had if he had cooperated, thus receiving less total utility. Consider the following sequence of actions shown in table 2.3. Bobs total utility would have been 20 if he had not defected.

| Iteration | Action | | Total utility | |
|-----------|-----------|-----------|---------------|-----|
| | Alice | Bob | Alice | Bob |
| 1 | Cooperate | Cooperate | 5 | 5 |
| 2 | Cooperate | Defect | 5 | 15 |
| 3 | Defect | Defect | 6 | 16 |
| 4 | Defect | Defect | 7 | 17 |

Table 2.3: Example sequence of the repeated prisoners dilemma with Alice playing a trigger strategy.

As any rational player will always try to maximize his utility, the equilibrium strategy to play here is cooperate, which differs from the stage game Nash equilibrium. There is one more thing to keep in mind: the fear of retaliation must be valid. Two popular punishing equilibrium strategies are the *grimm trigger* and *tit for tat*. The grimm trigger will punish an opponent by reverting to defect, and never stop defecting, as soon as the opponent stops cooperating. Tit for tat, or more generic: tit for n tats, will have the punishing player defect until the opponent has resumed cooperating for n iterations.

The previous example shows that retaliation is a powerful force, and can make agents move away from playing the stage game Nash equilibrium strategy and any social optimum will dominate that Nash equilibrium. It may even dominate in a finitely repeated game, if it is unknown which repetition will be the last. The social optimum could quite possibly be the goal of the game. This optimum is also called a Pareto optimum:

Definition 5 (Pareto optimal). *An outcome of a game is called Pareto optimal, or Pareto efficient, if no player can be better off without making another worse.*

A trigger strategy is a solution to the repeated game, provided the punishment is fierce enough. The equilibrium concept this strategy is a part of is called *Evolutionary Stable State*. An evolutionary stable state has the property that if all players use it, no single player can use a different strategy and be better off in the long run, i.e. no rational alternative exists. If all agents in the prisoner's dilemma play the trigger strategy and one decides to defect, he will not be better off. Always-defect however, is also an evolutionary stable state in this game. We will further investigate evolutionary stable states in section 4.2.

2.4 Bayesian Games

The games discussed thus far were clear to all players. All agents had the same information and all the actions were known to each player. A game in which all actions are known to all players is called a game with *complete* information, which refers to the structure of the game. If all information is known, the game is called a game with *perfect* information, referring to the game's state. In the real world, most games are either incomplete or imperfect or both.

Provisions for games with incomplete and imperfect information have already been made in definition 1. Each player i has private information θ_i (imperfect information) and a private action set A_i (incomplete information). The prisoner's dilemma (table 2.2) is a game of complete information as all possible actions are known to both players. It is also a game of imperfect information, since neither player knows what the opponent's action is when they play theirs.

In an extensive form game, where each node shows a state of a game, the uncertainty about other player's actions can be visualised using *information sets*. An information set is a collection of nodes in an extensive form game in which an agent can be at any time. In which of these nodes the game really is, depends on the action played by the opponent, and is unknown by the agent. Figure 2.4 shows Bob and Alice playing the prisoner's dilemma. When Alice decides which action to take, she cannot tell whether Bob is cooperating or defecting, indicated by the dotted line. Her information set contains two nodes, one for each possible situation. In this case, defecting will still be the rational choice, as it is the dominant strategy. In a game of perfect information each information set contains exactly one node.

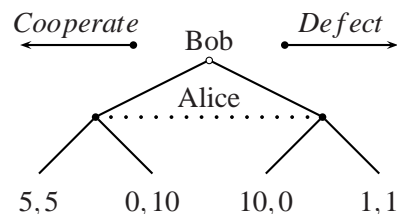


Figure 2.3: The information set for Alice when playing the Prisoner's Dilemma. Alice does not know in which state the game is.

Games with imperfect information form a class called *Bayesian Games*. Each player i in a Bayesian game has a *type* θ_i that fully describes his private information. Though players do not know the true type of their opponents, they do have beliefs about them. These beliefs form a type space Θ_i for player i .

Definition 6 (Bayesian game). *A Bayesian game extends the game defined in definition 1 by providing a type-profile $\theta = (\theta_1, \dots, \theta_n)$: an n -tuple of types, one for each player. This type-profile comes from the joint type space of all types believed to be in the game $\Theta = \prod_i \Theta_i$. When focusing on a player i 's opponents, the type-profile of these opponents is called a deleted type-profile, denoted as $\theta_{-i} = (\theta_1, \dots, \theta_{i-1}, \theta_{i+1}, \dots, \theta_n) \in \Theta_{-i}$. A strategy in a Bayesian game now becomes a function $s_i : \Theta_i \rightarrow A_i$.*

As agents may find themselves in several nodes of the game tree at once, due to the uncertainty of an opponent's actions, subgame perfect equilibria may not be available. To find a solution in a Bayesian game, John Harsanyi proposed introducing a new player, often called "Nature" or "Chance" [10].⁴ Chance will be the first to move and selects a type-profile from Θ and reveals each player his type. Chance selects this type-profile using a probability distribution over Θ , $p \in \Delta\Theta$, when it makes its initial move. This probability distribution is assumed common knowledge, i.e. all agents know it, and each player may derive his beliefs about his opponents' types from it. The belief player i has, may be seen as a conditional probability $\hat{p}_i(\theta_{-i}|\theta_i) \in \Delta\Theta_{-i}$: the probability that deleted type-profile θ_{-i} is used in this game, given the player's own type θ_i (which is obviously known).

The solution concept for this type of game is a generalization of the Nash equilibrium, a Bayesian-Nash equilibrium. Rather than using the utility for evaluating strategies, agents use the *expected* utility they receive, distributed according to the common knowledge distribution used by the chance player.

Definition 7 (Bayesian-Nash Equilibrium). *A set of strategies $s^* = (s_1^*(\theta_1), \dots, s_n^*(\theta_n))$ is a Bayesian-Nash equilibrium if for every agent i , $\theta_i \in \Theta_i$ and $s_i \neq s_i^*$:*

$$\mathbb{E} [u_i(o(s_i^*, s_{-i}^*), \theta_i)] \geq \mathbb{E} [u_i(o(s_i, s_{-i}^*), \theta_i)].$$

Consider Alice has an item she wants to sell to Bob or Carol in an auction. Bob and Carol have a private valuation for the item, taken uniformly from $\Theta_i = [0, 1]$, i.e. $\theta_i \sim U(0, 1)$. Alice's rules are that both players make a private bid and the highest bidder wins the item. It would seem reasonable for both players to bid low, but this way they risk not getting the item even though they may value it more than the other. So they will have to bid higher, until they reach some cutoff value. This value turns out to depend on the payment scheme Alice has in mind.

Since neither player knows the opponents valuation, they have no idea what to bid. Both players will try to maximize their *expected* utility, which simply is the difference between their valuation $v_i = \theta_i$ and the price they pay; not winning yields a utility of 0:

$$u_i(p) = \begin{cases} v_i - p & \text{if player } i \text{ wins} \\ 0 & \text{otherwise} \end{cases}.$$

⁴ Harsanyi was awarded the Nobel Memorial Prize in Economics for this work, together with John Nash and Reinhard Selten.

If Alice has the winning player pay his bid, Bob will bid the value maximizing his expected utility:

$$\mathbb{E}[u_{Bob}(b_{Bob})|b_{Bob}] = p(b_{Bob} > b_{Carol})(v_{Bob} - b_{Bob}).$$

To get to the probability of winning $p(b_{Bob} > b_{Carol})$, Bob guesses Carol will bid some fraction $0 < \alpha \leq 1$ of her valuation, yielding $p(b_{Bob} > \alpha v_{Carol}) = p(v_{Carol} < b_{Bob}/\alpha) = b_{Bob}/\alpha$ as $v_{Carol} \sim U(0, 1)$. Now Bob can find the value to bid by differentiating with respect to b_{Bob} :

$$\begin{aligned} \mathbb{E}[u_{Bob}(b_{Bob})|b_{Bob}] &= \frac{b_{Bob}}{\alpha}(v_{Bob} - b_{Bob}), \\ \frac{d}{db_{Bob}} \left(\frac{b_{Bob}}{\alpha}(v_{Bob} - b_{Bob}) \right) &= 0, \\ b_{Bob} &= \frac{v_{Bob}}{2}. \end{aligned}$$

As the same logic applies to Carol, both players will bid only half their valuation. Even though this is not their true valuation of the item, the players receive a higher expected utility.

If Alice will set the price at the second highest bid, both players will bid their valuation: Bob notices that when $b_{Carol} < v_{Bob}$ any bid $b_{Bob} \geq b_{Carol}$ is optimal (Bob will pay Carol's bid), and when $b_{Carol} \geq v_{Bob}$ any bid $b_{Bob} < b_{Carol}$ is optimal (Bob will not win). Bidding $b_{Bob} = v_{Bob}$ solves both inequalities. This approach will yield better results for Alice.

Chapter 3

Mechanism Design

Mechanism design is often called the inverse of game theory: where game theory studies how to act in a way that is most beneficial to the agent given a set of rules, mechanism design studies how to set the rules in order to have agents act in a way most beneficial to the designer. The previous chapter has shown games with clear rules for every player. There was no greater agenda, each agent was solely interested in its own utility. We will now focus on games that have been specifically designed with a greater goal in mind.

Leonid Hurwicz has been considered to be the founder of mechanism design theory in the early 1960s.¹ His view contained agents exchanging messages to a central algorithm that combines them into an outcome. His mechanism would define rules for the exchange of messages and the goal was to implement desired outcomes as equilibria. Interested readers should refer to Nisan et al. [21, 20] and Dash et al. [9].

In order to get a firm grasp of the theory, some mechanism design basics will be discussed, together with a number of problems it was first applied to. After this, concepts such as truthfulness and the revelation principle are presented, which will lead to some important results in mechanism design.

3.1 Concepts of Mechanisms

For as long as economics has existed, studies into the efficient allocation of goods and services have been conducted. With the introduction of game theory many situations could adequately be described and predicted. In a free market with a “sufficient” number of traders, the goods they trade will have a “fair” price; this price is an equilibrium inherent in this type of market. Things change quickly however, when the number of traders becomes *insufficient*, or the goods are public goods, e.g. health care or the decision to build a road.

In situations where the equilibrium solution is not efficient, rules need to be set to force the equilibria to be efficient, or rather: implement efficient outcomes to be equilibria. As we have seen in the previous chapter, multiagent systems, modelled

¹ Hurwicz was awarded the Nobel memorial award in economics in 2007, together with Myerson and Maskin, for laying the foundations of mechanism design. A thorough overview of his work as well as the entire field of mechanism design can be found in the report accompanying the Nobel award. It is available at http://nobelprize.org/nobel_prizes/economics/laureates/2007/ecoadv07.pdf.

as games, may have multiple equilibria, and making all these outcomes efficient has proved quite hard indeed. More generally speaking, designers of mechanisms will try to implement outcomes as equilibria that maximize a very specific *outcome function*. These functions can be measured in agent utility, such as a social optimum, or in money, e.g. total revenue for a seller in an auction.

To aid in the implementation of efficient equilibria, mechanism designers use incentives. The mechanism will provide agents with something they like if they act according to the designer's wishes. The most intuitive form of incentive is money. Besides providing an outcome function, the social choice function, mechanisms with money will also implement a separate *payment function*.

Definition 8 (Mechanism). *A mechanism $\mathcal{M} = \langle I, O, f, (\Theta_i, p_i, u_i)_{i \in I} \rangle$ contains a set of possible outcomes O , a set of agents I of size N and a social choice function $f : \Theta \rightarrow O$. Each of these agents i has a type space Θ_i , with $\Theta = \prod_{i=1}^N \Theta_i$ the joint type space of the agent population I , and a payment rule $p_i : \Theta \rightarrow \mathbb{R}$. The agent's utility u_i depends on his type $\theta_i \in \Theta_i$, the outcome $o \in O$ and the payment rule p_i .*

The use of payment as an incentive leads to a desirable property for mechanisms: *budget balance*. This property states that a mechanism does not require any payments coming into and flowing out of the mechanism, i.e. $\sum_i p_i(\theta) = 0$. The weaker notion of budget balance will only constrain the payment entering the mechanism: $\sum_i p_i(\theta) \geq 0$. Furthermore, agents should not be forced to enter a mechanism. This is known as *individual rationality* or the participation constraint. Intuitively, this means agents *expect* a higher utility by participating in the mechanism than without participating:

Definition 9 (Individual rationality). *A mechanism is individual-rational if for each type θ_i , its social choice function $f(\theta)$ has*

$$\mathbb{E}[u_i(f(\theta_i, \theta_{-i}))] \geq \mathbb{E}[\bar{u}_i(\theta_i)],$$

where $u_i(f(\theta_i, \theta_{-i}))$ is distributed according to $p_i \in \Delta\Theta_{-i}$, which is derived from the commonly known distribution, and $\bar{u}_i(\theta_i)$ is the utility the agent would receive if he did not participate.

As illustrated by the auction example in section 2.4, it may benefit an agent in a mechanism to report a type θ'_i that is different from its true type θ_i . This will, in most cases, result in a less than efficient outcome. Mechanism designers will try to make agents report their true type by setting rules and incentives. This leads to the notion of truthfulness in the mechanism. An agent is truthful when he reports his true type, i.e. for every $\theta_i \in \Theta_i$: $s(\theta_i) = \theta_i$. Closely related to truthfulness is *incentive compatibility*: A mechanism is incentive compatible when truthfulness is the equilibrium solution, more specifically: truthfulness is the dominant strategy, i.e. $s^*(\theta) = \{\theta_1, \dots, \theta_n\}$.

3.2 The Revelation Principle

In many situations it is useful to have agents report their type in a single step, rather than spread out over several steps, like in an English (ascending price) auction. Mechanisms that have this property are called *direct-revelation mechanisms*. Direct-revelation

mechanisms are a simple mathematical abstraction and not all may have a real world application. They are however relatively easy to study. Once an optimal direct-revelation mechanism has been identified, designers can “easily” translate it back to a specific mechanism. It has been shown that any equilibrium that may be achieved by any mechanism can also be implemented in a direct-revelation incentive compatible mechanism, i.e. agents report their true type in a single step. This is called the *revelation principle*. Note that this does not influence the value of the outcome, the same result will be attained, only in the direct-revelation incentive compatible mechanism true types are reported and all these are received in a single step. The revelation principle has been applied to a wide range of mechanisms, including mechanisms with multiple stages and agents having hidden actions. It turns out that the class of direct-revelation mechanisms is broad enough to have a truthful version of each mechanism, while being a relatively small set [37].

Proposition 10 (Revelation Principle). *Given a mechanism that implements a social choice function f in dominant strategies, there also exists an incentive compatible direct-revelation mechanism implementing f in dominant strategies. Payments of agents are equal to the payments in the original mechanism.*

The revelation principle in this form, implementing f in dominant strategies, is the strongest form. Mechanism designers prefer to implement equilibria in dominant strategies, because it enables agents to select a strategy without knowledge of other agents or any other outside information. Weaker versions also hold, e.g. any mechanism implementing f in a Bayesian-Nash equilibrium also has an incentive compatible version. A proof for the above proposition is given in [20].

Auctions are a perfect example of a mechanism used to solve an allocation problem. Several formats exist: English (ascending price), Dutch (descending price) and more exotic types, but none of these will motivate agents to bid their true valuation. In an English auction format bidders will keep raising the current bid until it matches their private valuation or they win. There is no incentive to provide the private information. The winning price will, at least theoretically, be equal to the second highest bid, as at that point the winning bidder is the only bidder left who is willing to pay that price and will certainly not bid more.

The direct mechanism for this auction is the Vickrey auction: each bidder provides a sealed bid to the auctioneer, who proceeds to award the item to the highest bidder. The winner will pay the second highest bid, which is the same result as the (theoretical) price of the original English auction. In this mechanism there is no need to hide private information. Though there is no obligation to be truthful, providing false information may result in either paying too much, not getting the item at all or has no effect: truthfulness is the dominant strategy. The Vickrey auction is a classic example implementing the outcome of (e.g.) an English auction in an incentive compatible mechanism.

For agents with quasilinear utility functions, i.e. having utility functions of the form $u(\cdot) = v(\cdot) - p$, the Vickrey-Clarke-Groves (VCG) family of mechanisms is the class of mechanisms that are both incentive compatible and are (weakly) budget balanced. We have already seen the Vickrey auction as a member of this family [25]. When an agent i reports type θ'_i , which may or may not be his true type, the VCG mechanism will compute $o^* = \arg \max_{o \in O} \sum_i v_i(o, \theta'_i)$, the (global) optimal outcome

given the reported types. The payment rule is $p_i = h_i(\theta'_{-i}) - \sum_{j \neq i} v_j(o^*, \theta'_j)$, where $h_i : \Theta_{-i} \rightarrow \mathbb{R}$ is an arbitrary function. By choosing some function independent of agent i 's valuation, the designer makes it impossible for an agent to directly influence his payment. The utility function for agent i , as it is quasilinear, will now be:

$$u_i(o^*, p_i, \theta'_i) = v_i(o^*, \theta_i) + \sum_{j \neq i} v_j(o^*, \theta'_j) - h_i(\theta'_{-i}),$$

which maximizes for $\theta_i^* = \arg \max_{\theta_i \in \Theta_i} v_i(o(\theta_i, \theta'_{-i}), \theta_i) = \theta_i$, i.e. at a truthful revelation. Two things are important here: 1) the agent should realize that truth-revelation will lead to the outcome he favors most (given other players' type revelations), and 2) his revealed type has no influence on the payment he will have to make.

Having a direct-revelation incentive compatible mechanism according to the revelation principle is very useful. However, as incentive compatibility is only one of the desired properties of a mechanism, it should also facilitate others, such as budget balance and individual rationality. Unfortunately it has been shown that for many situations, combinations of these properties are not possible [25].

The most important impossibility result is by Gibbard and Satterthwaite, who showed that a mechanism with a social choice function implemented in dominant strategies with $|O| \geq 3$, is a *dictatorship*. When a mechanism is a dictatorship, there is one agent i with the property that the social choice function f strictly prefers o over o' whenever i strictly prefers o over o' . The most intuitive example of a dictatorship, with agent i the dictator, is $o = \arg \max_{o \in O} v_i(o, \theta_i)$. This implies that there exists no incentive compatible mechanism with more than two outcomes, implemented in dominant strategies, where the agents can have any type of utility function, that has a Pareto efficient outcome. Other impossibility results have been added for more specific settings, e.g. by Hurwicz, and by Myerson and Satterthwaite. Hurwicz' theorem states that there is no direct mechanism implemented in dominant strategies and whose outcomes are both efficient and individually rational. Myerson and Satterthwaite showed the non-existence of direct mechanisms achieving efficient outcomes having budget balance and individual rationality implemented in Bayesian-Nash equilibria (which is a much weaker equilibrium than dominant strategies).

If, however, additional constraints are used, e.g. on the type of utility function or the solution concept, the Gibbard Satterthwaite result may no longer hold. The VGA class of mechanisms needs quasilinear utility functions to escape from the dictatorship. Using these, the mechanism is able to achieve an efficient outcome implemented in dominant strategies, while also providing individual rationality and weak budget balance.

When a designer has sufficiently constrained the scope of the mechanism to be able to find a direct mechanism having the right properties, he may find it to be NP-complete: it may have exponential cost in communication or in computation of the social choice function, or both. A combinatorial auction, where several goods are to be sold in an unknown partition, has no tractable mechanism. One solution to this problem is to find a mechanism according to the revelation principle that *approximates* the original social choice function. Lavi and Swamy [15] provide an approximation technique using linear programming that yields mechanisms that are truthful in expectation, i.e. truthfulness is the Bayesian-Nash equilibrium. This approximation mechanism for a combinatorial auction achieves an approximation guarantee of $O(\sqrt{m})$, with

m the number of items. The subfield of finding incentive compatible approximation mechanisms is called algorithmic mechanism design [21, 20].²

Alternatively, designers can let go of truthfulness as an equilibrium strategy. Conitzer and Sandholm show in [6], that for a mechanism that tries to optimize social welfare, without the use of payments, even with only two agents, there exists a family of mechanisms that performs only polynomial computations, and makes finding a beneficial, non-truthful revelation by the agent NP-complete. If the agent does succeed in finding such a beneficial, non-truthful revelation, the social welfare will be equal to the social welfare in any optimal truthful mechanism. If he does not, the welfare produced will be strictly better than that produced by any optimal truthful mechanism. They show a similar result for communicationally intractable truthful mechanisms.

²Interested readers should refer to the tech talk Noam Nisan gave entitled “Algorithmic Mechanism Design”, c.f. Appendix A.

Chapter 4

Learning

Multiagent learning has recently become very popular among (algorithmic) game theorists. As computing (Nash) equilibria is often intractable, agents inferring the best action to play based on some heuristic may provide a good approximation, see Shoham et al. [30].

Repeated mechanisms have the opportunity to adapt their rules and incentives between iterations. This way, a changing agent population may be countered and the mechanism's total outcome will improve. For this to happen, a way to find a best response to play in the next iteration must be found. This property is available in many algorithms in the field of machine learning and artificial intelligence, c.f. [28]. Many depend on past experiences and select a best action based on this history.

This chapter will discuss two methods of using observed data to choose an action that will most likely lead to the best result: Reinforcement Learning and Evolutionary Game Theory. For each of these learning algorithms, we will look at both their workings as well as their application to multiagent systems.

4.1 Reinforcement Learning

Reinforcement Learning is a very simple and intuitive concept: reward good behaviour and punish bad. The subject of learning will eventually realize which behaviour ends in reward and more often take these “good” actions. Animals are trained in this fashion, rewarding them after successfully completing a trick and punishing them when failing. Reinforcement learning is a well researched subject in the fields of artificial intelligence and machine learning and is making its way into a multiagent setting ([35, 13, 16, 17]). A thorough introduction is given by [32].

Learning by interacting with the environment is central to reinforcement learning: focusing on *sensation*, *action* and *goal*, reinforcement learning will teach an agent which action to take in order to maximize his goal using sensation. By learning from experience the agent will register consequences of his actions. This will allow him to map situations to actions to take when he finds himself in that situation. What action to take will need to be discovered by trial and error. Each action taken will lead to a reward or punishment (low or negative reward), depending on the situation the agent is in. Note that the agent is not told whether his action was the “right” action to play, he only receives a reward. Learning agents will try to maximize the total

reward they receive when acting in different situations. However, actions may affect not only immediate rewards, but also future rewards. Trial and error and the possibility of delayed rewards are very important in reinforcement learning.

To have agents learn, reinforcement learning uses four tools: 1) a *policy*, 2) a *reward function*, 3) a *value function* and 4) optionally a *model*. A policy provides a mapping from (perceived) states to actions. Each time an agent acts, new rewards are received and this policy may need updating. The reward function maps the (perceived) state to a reward, usually a number. This can be compared to the utility an agent receives at the end of each iteration of a repeated mechanism. Maximizing this total reward is the goal of the learning agent. The value function shows the total expected future reward when an agent is in a state. Whereas the reward function shows the immediate desirability of a state, this function indicates the long-term desirability of a state. The value function will also need constant updating as new actions lead to new rewards. Finally, the model will provide the agent with an idea of how the environment works.

The value function is the most important tool for a learning agent. Actions are selected based on these values: actions that lead to states with a high value yield more total reward in the long run. Acting based on the reward function would only bring immediate reward and ignores any delayed rewards that the action may lead to. Without the reward function however, the value function could not be calculated.

A learning agent must find out which action would be best in a given situation: the agent must try different actions to determine to what reward they lead. This process is called *exploration*. By exploring actions in different states, the policy that maps states to actions, i.e. the policy tells the agent how to act in the given state, will be refined. Ideally, when a policy is sufficiently refined, the agent will exclusively use this mapping and will no longer investigate potential better actions. The agent has changed switched to *exploitation*. As it is very hard to determine this moment, both exploration and exploitation are used alternately.

Exploitation of experience can be illustrated using a simple example. Each action has a value $Q(a)$, which is the reward for playing action a . As this can only be an expected value, we denote the true value $Q^*(a)$. The law of large numbers allows us to construct an unbiased estimator of $Q^*(a)$ at time t , using $Q_t(a) = (r_1 + \dots + r_{\#a})/\#a$, with $\#a$ the number of times action a has been selected and r_i the reward received. Exploitation (*greedy* selection) would select $a^* = \arg \max_a Q_t(a)$ at time t , exploration would select any other action. Greedy selection will only exploit and introducing a parameter $0 < \epsilon < 1$ will have the selection algorithm explore with probability ϵ . This method is called ϵ -greedy action selection and easily outperforms greedy algorithms.

More sophisticated methods for action selection exist. Rather than just selecting the action having maximal expected value, *softmax* action selection can be used. The selection rule assigns a probability $\pi_t(a)$ (at time t) of selection to the action based on the value it has:

$$\pi_t(a) = \frac{e^{Q_t(a)/\tau}}{\sum_{a' \in A} e^{Q_t(a')/\tau}}.$$

The parameter $\tau > 0$ is the *temperature*: higher temperatures favor exploration. The action selection methods discussed do not take the current state into account, such as was prescribed in the description.

The policy, reward function and value function all use the current state to calculate their value. This current state s is used to form a probability of the next state s' when playing action a and for the reward that is to be expected. Both the probability of next state s' $\mathcal{P}_{ss'}^a$ and expected reward $\mathcal{R}_{ss'}^a$ are conditioned on the current state:

$$\begin{aligned}\mathcal{P}_{ss'}^a &= \Pr [s_{t+1} = s' | s_t = s, a_t = a] \\ \mathcal{R}_{ss'}^a &= \mathbb{E} [r_{t+1} | s_t = s, s_{t+1} = s', a_t = a].\end{aligned}\quad (4.1)$$

Conditioning only on the current state, as opposed to the entire history of states, actions and rewards that have been observed, is only possible if the environment has the *Markov property*. In this case, the Markov property states, roughly speaking, that all information encountered in the entire history, is also available in the current state. Each state s_k is seen as a realization X_k of a random variable X and after k realizations, the probability of the t^{th} next realization being $X_{k+t} = j$, depends entirely on the value of current state s_k :

Definition 11 (Markov property). *The Markov property states that no knowledge of the history is needed to make expectations about the future, i.e. all information needed for prediction is available in the current state: For a Markov process X_t and $\{0, \dots, k\}$, $t \geq 0$ and $i_1, \dots, i_k, j \in X$ as realizations of X , we have $\Pr[X_{k+t} = j | X_k = i_k, \dots, X_1 = i_1] = \Pr[X_{k+t} = j | X_k = i_k]$.*

Systems that have the Markov property (or in short: are Markov), are called a Markov decision process (MDP). In most practical applications, the environment does not have the Markov property: the true state may not be fully observable (a partially observable Markov decision process), or more of the history is needed (the system is not Markov at all). However, reinforcement learning will still provide good results, even if the environment only approximately has the Markov property. The better this approximation, the better the results.

Using the assumption that the environment is (approximately) Markov and equations 4.1, value functions for learning can be constructed. Two types of value functions exist: taking only the state or both the state and an action. These functions indicate how good it is to be in a state and how good it is to perform a specific action in a state respectively. The notion of “goodness” is expressed in terms of expected reward. The probability distribution used by the value functions to compute this expectation is provided by the policy π , where $\pi(s, a)$ denotes the probability of playing action a while in state s . The value functions shown are known as the *Bellman equations*:

$$\begin{aligned}V^\pi(s) &= \mathbb{E}[R(t) | s^t = s] \\ &= \sum_{a \in A(s)} \pi(s, a) \sum_{s' \in S} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma V^\pi(s')], \\ Q^\pi(s, a) &= \mathbb{E}[R(t) | s^t = s, a^t = a] \\ &= \sum_{s' \in S} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma Q^\pi(s', a)].\end{aligned}\quad (4.2)$$

In equation 4.2 $R(t) = \sum_{i=1}^t r_i$ is used to denote the total reward. When the expected future reward is needed, the individual rewards are discounted, in order to make a (potentially) infinite sum yield a value: $R(k) = R(t) + \sum_{i=t+1}^k \gamma^{i-t-1} r_i$. A discount

factor $\gamma = 0$ has the agent act myopically, i.e. only maximize immediate reward, and as $\gamma \rightarrow 1$, the agent will be more interested in the long-term returns.

Policies π can be compared using their respective value functions, i.e. $\pi \geq \pi'$ if $V^\pi \geq V^{\pi'}$. As there may be more than one optimal policy, the value functions they represent are the same: V^* . The optimal value function is given by $V^*(s) = \max_\pi V^\pi(s)$ for all $s \in S$ and the optimal action-value function by $Q^*(s, a) = \max_\pi Q^\pi(s, a)$ for all $s \in S, a \in A$. Using these optimal policies, Q^* can be written in terms of V^* :

$$Q^*(s, a) = \mathbb{E}[r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a].$$

These optimal functions lead to the *Bellman optimality equations*:

$$\begin{aligned} V^*(s) &= \max_a Q^*(s, a) \\ &= \max_a \mathbb{E}[r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a] \\ &= \max_a \sum_{s' \in S} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma V^*(s')], \end{aligned} \quad (4.3)$$

$$\begin{aligned} Q^*(s, a) &= \max_a \mathbb{E} \left[r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') | s_t = s, a_t = a \right] \\ &= \max_a \sum_{s' \in S} \mathcal{P}_{ss'}^a \left[\mathcal{R}_{ss'}^a + \gamma \max_{a'} Q^*(s_{t+1}, a') \right]. \end{aligned} \quad (4.4)$$

Both the value function and the action-value function have now taken a form that is independent of the policy used. When $\mathcal{P}_{ss'}^a$ and $\mathcal{R}_{ss'}^a$ are known, there is a unique solution to these equations for finite MPDs.

4.1.1 Function Approximation

Learning using the (optimal) Bellman equations has one limitation: the number of states in the system must not be too big. As all states have to be explored in order to decide what action is optimal in any situation, it will take an agent longer to learn as the number of states increases. To increase the learning rate, *function approximation* may be used.

Function approximation allows a learning agent to update beliefs about multiple states at once. This means that only a relatively small fraction of the state space needs to be explored to have the agent decide on his actions. The agent will replace the true function, e.g. the value function V^π , with a parametrized function using a vector of parameters $\bar{\theta}_t$. After each action, the environment's response will lead to a change in this vector $\bar{\theta}_t \rightarrow \bar{\theta}_{t+1}$. This way, a small change in the parameters, induced by a single encountered state, will update the value function for many other states. The size of $\bar{\theta}_t$ is much smaller than the number of states in the system, improving the learning rate of the agent.

Many different methods may be used for approximating functions. The field of machine learning offers many examples of supervised function approximation techniques.¹ As reinforcement learning is a non-supervised and online learning scheme,

¹Supervised learning involves a “teacher” telling the learning method whether the right choice has been made and if not, what the right choice would have been. Learning this way uses a static training set which is used several times to have the agent learn.

each interaction with the environment is used as a training example for the supervised method.

To measure how well the parametrized function approximates the true function, the mean squared error is used:

$$MSE(\bar{\theta}_t) = \sum_{s \in S} \Pr(s) [V^\pi(s) - V_{\bar{\theta}_t}(s_t)]^2,$$

where V^π is the true value function and $\Pr(s)$ is the probability that a state s is encountered, which may be bootstrapped. The true value $V^\pi(s_t)$ is in many situations unknown and the agent may keep a backed up value, e.g. the sample mean, as an unbiased estimator of this true value. As encountered states are used as training examples for the supervised learning method, it is natural to select training examples using the same distribution. The goal of the approximation is to find the $\bar{\theta}^*$ such that $MSE(\bar{\theta}^*) \leq MSE(\bar{\theta})$ holds for all $\bar{\theta}$.

Using the mean squared error leads to a local optimum when using complex approximation such as Artificial Neural Networks. In some situations, the MSE may diverge and no suitable vector is found. The most widely used method for function approximation that guarantees both a global optimum and a converging MSE is *Gradient Descent*. It uses a straightforward update rule.

$$\begin{aligned} \bar{\theta}_{t+1} &= \bar{\theta}_t - \frac{1}{2} \alpha \nabla_{\bar{\theta}_t} [V^\pi(s_t) - V_t(s_t)] \\ &= \bar{\theta}_t + \alpha [V^\pi(s_t) - V_t(s_t)] \nabla_{\bar{\theta}_t} V_t(s_t), \end{aligned}$$

where $\nabla_{\bar{\theta}_t}$ is the vector of partial derivatives of V_t . The factor α is a positive step size parameter and should conform to two constraints in order to converge to a local optimum:

$$\begin{aligned} \sum_{i=0}^{\infty} \alpha_i &= \infty \\ \sum_{i=0}^{\infty} \alpha_i^2 &< \infty. \end{aligned} \tag{4.5}$$

When each parameter $\theta^{(i)}$ is linear, each corresponding partial derivative $\nabla_{\bar{\theta}_t}^{(i)}$ contains a single term, and the approximation will lead to a global optimum.

4.2 Evolutionary Game Theory

In 1973 John Maynard Smith and George Price were looking to model biological evolution. In their work [19] the authors used the concepts of rational agents and the theory of games to achieve this. What became known as evolutionary game theory (EGT) enabled biologists to model different inheritable characteristics of a species of animal and determine why some would prevail over time while others would perish.

Using game theory, biologists would model inheritable characteristics, such as skin coloration or the size of antlers, as strategies for agents: individual animals in a population. The goal of these agents is to have the most descendants, so payoff is provided by Darwinian fitness. For a short introduction in EGT from a biological perspective,

refer to [31]. The models invented using evolutionary game theory even explain why some characteristics could periodically nearly be lost, only to return to become almost dominant. Game theorists have then taken EGT back from biologists to study how the distribution of strategies in a population of agents evolves as a game is repeated; Recently, EGT has found its way to mechanism design [4, 26].

EGT approaches can roughly be placed into two groups: a) using *evolutionary stable states* and b) creating a model of the evolutionary dynamics. The former method, the approach taken by Maynard Smith and Price, aims to identify states, i.e. distributions of strategies in the population, that are stable. Stability indicates that small perturbations in the distribution will have no lasting effect: in time the population will return to the stable state.

Using evolutionary stable states, biologists are able to model evolutionary processes like genetic drift. To model the other major process, mutation, the concept of *evolutionary stable strategy* (ESS) was introduced. A mutation is seen as a new, external strategy entering the system. Both concepts are related, as both indicate a strategy distribution in the population. However, evolutionary stable states need not coincide with evolutionary stable strategies. Whereas an evolutionary stable state describes stability against *internal* strategies, an ESS captures stability against *external* strategies.

Nature allows successful genetic variations to prosper using survival of the fittest. Evolutionary game theory attempts to incorporate the same scheme to increase successful strategies and decrease others. Agents are modelled to have pair-wise interaction; agents play a stage game against a random opponent, using a strategy selected randomly according to the strategy distribution. After each interaction, this distribution is updated, favoring the successful, using some dynamics. Several models of these dynamics have been proposed, differing in assumptions on the population, such as availability of information and intelligence.

The theoretical framework for modeling evolutionary dynamics assumes an infinite sized population. This allows the individual fractions of strategies in the population to be differentiable with respect to time. The model consists of a system of differential equations, one for each strategy i available. The i^{th} differential equation indicates the increase or decrease of the fraction of the population using strategy i .

To form an intuition about modelling the evolutionary dynamics, we will show an example model using the prisoners dilemma, c.f. table 2.2 for the payoffs. Each strategy, Cooperate (C) and Defect (D), has an initial fitness (F_C, F_D) and fraction (p_C, p_D , with $p_C + p_D = 1$), the initial distribution of the strategies over the population. Strategy fitness is denoted W_C and W_D (recall that fitness is the payoff for a strategy), the average fitness of the entire population is denoted $\bar{W} = p_C W_C + p_D W_D$. The fitness of each strategy can now be calculated using:

$$\begin{aligned} W_C &= F_C + p_C F(C, C) + p_D F(C, D) \\ W_D &= F_D + p_C F(D, C) + p_D F(D, D), \end{aligned}$$

with $F(a, b)$ the fitness (payoff) received when playing strategy a against b . This system simply defines fitness as the sum of the expected fitness and the initial fitness. It is then assumed that the new proportions are related to the current proportions and the payoffs received, according to:

$$p_i' = \frac{p_i W_i}{\bar{W}},$$

| | Rock | Paper | Scissors |
|----------|------|-------|----------|
| Rock | 0 | 1 | 0 |
| Paper | 0 | 0 | 1 |
| Scissors | 1 | 0 | 0 |

Table 4.1: Payoff matrix for the Rock Paper Scissors game.

from which a differential equation can be derived:

$$\begin{aligned}
 \frac{dp_i}{dt} &= p'_i - p_i \\
 &= \frac{p_i W_i}{\bar{W}} - p_i \\
 &= \frac{p_i(W_i - \bar{W})}{\bar{W}}.
 \end{aligned}$$

These equations show how each strategy increases and decreases over time, i.e. between iterations.

We have now defined enough tools to analyse this game. As $F(D, C) > F(C, C)$ and $F(D, D) > F(C, D)$, it follows that $W_D > \bar{W} > W_C$ and, more importantly, $\frac{dp_D}{dt} > 0$ and $\frac{dp_C}{dt} < 0$. This indicates that no matter what distribution of strategies the population starts with, eventually each agent will end up only playing Defect: (Defect, Defect) is an evolutionary stable state. Only the initial distribution with every agent playing Cooperate would not lead to this result. In this case however, Defect could be interpreted as an unavailable strategy, an external invader. As an invader, Defect would quickly gain popularity and the population would again move towards only defecting agents, making Defect (in this case) both an evolutionary stable state and an ESS.

With the system of differential equations, we can plot a direction field in the unit-simplex. Each point in the simplex corresponds to a distribution of strategies \mathbf{x} . N available strategies require N differential equations that lead to a direction field in the unit N -simplex S_N . Table 4.2 shows the payoff matrix of the ‘‘Rock Paper Scissors’’ game. This game has three strategies and no pure strategy Nash Equilibrium. Figure 4.1 shows the corresponding unit 3-simplex containing the direction field of the dynamics.

Rest points are evolutionary stable points $\mathbf{x} \in S_N$. The corners of the simplex are always rest points and there may be an additional rest point in the interior of S_N . Finding the latter rest point depends on the model of the dynamics used. The interior rest point in figure 4.1 is the limit of an orbit at $(1/3, 1/3, 1/3)$. Rest points in the simplex are useful indicators for the analysis of the game [12]:

1. if \mathbf{x} is a Nash equilibrium, it is a rest point
2. if \mathbf{x} is a *strict* Nash equilibrium, it is asymptotically stable²

²Asymptotic stability means that not only do initial conditions close to the point \mathbf{x} stay close to \mathbf{x}

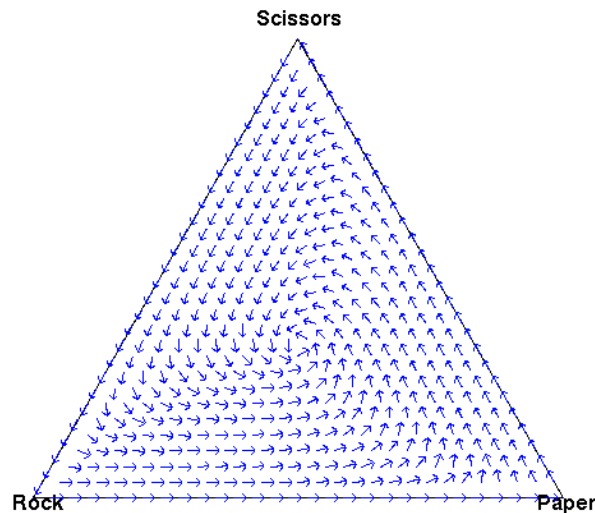


Figure 4.1: Unit 3-simplex S_3 with evolutionary trajectories of strategies.

3. if \mathbf{x} is the limit of an orbit, then for $t \rightarrow \infty$, it is a Nash equilibrium
4. if rest point \mathbf{x} is stable, it is a Nash equilibrium

It is easily verified that these properties hold for Rock Paper Scissors in figure 4.1; indeed, $(1/3, 1/3, 1/3)$ is the single (mixed strategy) Nash equilibrium of the game. The inverse of any of the four properties do *not* hold in general.

Replicator Dynamics The most intuitive and most widely used model of evolutionary dynamics is known as the *Replicator Dynamics*. As before, $\mathbf{x} \in S_N$ denotes the distribution of strategies (the replicators) over the population: each x_i signifies the relative frequency of strategy i . It is assumed that the x_i are differentiable with respect to time (which requires an infinite population). Now the dynamics can be modeled as motion in time: $\mathbf{x}(t)$. Using the payoff matrix A , the success of each strategy against another is measured, the expected payoff for any agent playing strategy i is given by $(A\mathbf{x})_i$, the average population payoff is $\mathbf{x} \cdot A\mathbf{x}$. Finally, it is assumed that the per capita rate of growth is given by the difference between the expected payoff when playing i and the population average payoff [12] and [3, section 2.5]

$$\dot{x}_i = x_i ((A\mathbf{x})_i - \mathbf{x} \cdot A\mathbf{x}),$$

with \dot{x}_i the derivative of x_i with respect to time. The interior rest point for this system is \mathbf{x} for which each $(A\mathbf{x})_i = \mathbf{x} \cdot A\mathbf{x}$, so each $\dot{x}_i = 0$. \mathbf{x} is found as a solution $(A\mathbf{x})_1 = \dots = (A\mathbf{x})_n$, which generically has at most one solution. Under certain conditions, these replicator dynamics reduce to the reinforcement learning discussed in section 4.1 [34].

Using the replicator dynamics the conditions for an ESS can be derived. A strategy profile $\mathbf{p} \in S_N$ is evolutionary stable if it is more successful than any other strategy

(stable), they also approach \mathbf{x} asymptotically (in the limit as $t \rightarrow \infty$). It imposes restrictions on trajectories, making it stronger than stability.

profile $\mathbf{p}' \neq \mathbf{p}$, provided that the frequency for \mathbf{p}' is sufficiently small, below some “invasion barrier”. With x the frequency of the invader and

$$\dot{x} = x(1-x)(x(\mathbf{p}' \cdot \mathbf{A}\mathbf{p}' - \mathbf{p} \cdot \mathbf{A}\mathbf{p}') - (1-x)(\mathbf{p} \cdot \mathbf{A}\mathbf{p} - \mathbf{p}' \cdot \mathbf{A}\mathbf{p})),$$

the rest point with $\dot{x} = 0$, i.e. the invader loses and dies out, is asymptotically stable if two conditions are met:

equilibrium condition $\mathbf{p}' \cdot \mathbf{A}\mathbf{p} \leq \mathbf{p} \cdot \mathbf{A}\mathbf{p}$, or: the ESS must perform *at least* as well as the invader against the ESS.

stability condition If $\mathbf{p}' \cdot \mathbf{A}\mathbf{p} = \mathbf{p} \cdot \mathbf{A}\mathbf{p}$ then $\mathbf{p}' \cdot \mathbf{A}\mathbf{p}' < \mathbf{p} \cdot \mathbf{A}\mathbf{p}'$, or: If an invader performs as well against an ESS, it must perform worse against itself than an ESS.

Other models of evolutionary dynamics also exist. The replicator dynamics require a discrete set of payoffs, so a payoff matrix A can be used. A more general variation also allows for non-linear payoff functions to be used:

$$\dot{x}_i = x_i(a_i(\mathbf{x}) - a(\mathbf{x})).$$

Again, $a_i(\mathbf{x})$ denotes the expected payoff for strategy i when playing a population with distribution \mathbf{x} and $a(\mathbf{x}) = \sum_i x_i a_i(\mathbf{x})$ denotes the average population payoff.

Imitation Dynamics Both models mentioned assume information is available to the entire population. This includes the expected payoff for playing a strategy i and the average population payoff. In many situations this information is not available. A different approach is taken by the *Imitation Dynamics*, c.f. [11, section 8.1] and [38]. Imitation dynamics focuses on the success of strategies played by an agent’s neighbours. The assumptions on the availability of general information about the entire population are now reduced to the assumption of visibility of a neighbour’s strategy and associated success.

At discrete times, an agent compares his information with that of his neighbours. Based on these observations, the agent may update his strategy. These dynamics will favor strategies that have provided a higher utility for the neighbours, not necessarily with a higher utility for the population as a whole. With f_{ij} denoting the likelihood of replacing strategy i with j , the general imitation dynamics are

$$\dot{x}_i = x_i \sum_j (f_{ij}(x) - f_{ji}(x))x_j.$$

Often f_{ij} is taken to be a function depending on the current payoffs, i.e. $f_{ij}(\mathbf{x}) = f(a_i(\mathbf{x}), a_j(\mathbf{x}))$. This function $f(u, v)$ is the imitation rule. The simplest imitation rule would be

$$f(u, v) = \begin{cases} 1 & \text{if } u \geq v \\ 0 & \text{if } u < v. \end{cases}$$

This scheme has the value of x_i increased if its payoff exceeds the *median* rather than the *mean*.

Unfortunately this leads to a discontinuous right hand side. Assuming the imitation rule is an increasing, and continuous, function $f(u, v) = \phi(u - v)$ of the payoff is also plausible and leads to

$$\dot{x}_i = x_i \sum_j \phi(a_i(\mathbf{x}) - a_j(\mathbf{x}))x_j.$$

Multiple Populations Many games involve agents from different, distinct populations interacting. Opposing agents may have different strategy sets and different utility functions, e.g. bidders and sellers in auctions. Both populations influence the effectiveness of each others strategies, see [12, 29]. As such, the dynamics that are used to model these systems are coupled. Equation (4.6) shows an example system with two populations: sellers and bidders. The populations have payoffs A_{seller} and A_{bidder} , n_{seller} and n_{bidder} strategies with distributions $\mathbf{x}_{\text{seller}}$ and $\mathbf{x}_{\text{bidder}}$. Matrix A_i for agent type i is an $n_j \times n_i$ matrix.

$$\begin{aligned}\dot{x}_{i,\text{bidder}} &= x_{i,\text{bidder}} \left((A_{\text{bidder}}\mathbf{x}_{\text{seller}})_i - \mathbf{x}_{\text{bidder}} \cdot A_{\text{bidder}}\mathbf{x}_{\text{seller}} \right) \\ \dot{x}_{i,\text{seller}} &= x_{i,\text{seller}} \left((A_{\text{seller}}\mathbf{x}_{\text{bidder}})_i - \mathbf{x}_{\text{seller}} \cdot A_{\text{seller}}\mathbf{x}_{\text{bidder}} \right)\end{aligned}\quad (4.6)$$

Mixed Strategies All dynamics discussed have implicitly assumed that the strategies used were *pure* strategies, rather than *mixed* strategies, c.f. definition 4. The mixed strategy $\mathbf{p}(i) \in S_n$ may be seen as a probability distribution over the strategies for the agent, not the population.³ The expected payoff for an agent playing type i (which has probability vector $\mathbf{p}(i)$) against an agent playing j is $u_{ij} = \mathbf{p}(i) \cdot A\mathbf{p}(j)$.

As before, $\mathbf{x} \in S_n$ denotes the relative frequency of each strategy in the population. The expected strategy used by the population can now be calculated by

$$\mathbf{p}(\mathbf{x}) = \sum_i x_i \mathbf{p}_i(i).$$

With the payoff matrix A as before, the dynamics for a population with mixed strategies are

$$\dot{x}_i = x_i \left((\mathbf{p}(i) - \mathbf{p}(\mathbf{x})) \cdot A\mathbf{p}(\mathbf{x}) \right).$$

Finite Populations As stated before, in order to get a smooth (differentiable) dynamics, the population size should be infinite. In most practical cases, however, this assumption is not valid. Implications for the dynamics are significant: changes in the distribution of a strategy can no longer occur in arbitrarily small fractions, but as one agent changes his strategy, the corresponding distribution changes by an amount $1/N$, with N the number of agents. Using the dynamics in the way described will show what strategy is favorable given the current distribution, but changing this distribution with a relatively big value may cause it to no longer be favorable, and may even make it perform worse than before in the new distribution.

Different models have been proposed to modify the evolutionary game dynamics, see [12, section 4.6] for a brief overview. Most of these modifications involve the use of stochastics, e.g. the introduction of noise to the replicator equations or modeling the entire system as a Markov process, with each stochastic variable describing the number of agents using the strategy.

³Because the probability distribution $\mathbf{p}(i) \in S_n$ is seen at the agent, $n = N$, with N the number of pure strategies in the population, is not required to hold. Therefore the simplex S_n for $\mathbf{p}(i)$ may also differ from the simplex S_N .

Chapter 5

Adaptive Mechanisms

A mechanism has a well defined goal, e.g. revenue maximization in an auction, maximizing employee participation in a company or maximizing total utility in a society. Each set of actions from the agents will lead to an outcome and possibly a payment to the agents, computed by the mechanism. The mechanism designers will prefer some outcome over others, which can be quantified using a utility function just like individual agents have (for some outcome $o^* \in O$ and some payment $p^* \in P$: $u(o^*, p^*) \geq u(o, p)$ for all $o \in O, p \in P$).

As mechanisms are designed to fit a target population, the expected actions of this population will provide the highest utility to the mechanism. However, when the mechanism is to be repeated over time, evolutionary stable states, providing there are any, may differ from single shot equilibrium. Also the agents may find ways “around the system”: they may think of new, invading, strategies or change their behaviour in another way. This will result in a situation in which the mechanism will provide suboptimal results. To counter an adapting agent population, the mechanism itself must be able to adapt.

In searching for ways to create an adaptive mechanism, we shall first introduce a *meta-game*. As the mechanism itself will adapt the strategies it has to the encountered agents, it will be showing strategic behaviour and may be studied as a player in a game itself: the meta-game. When the general setting of this game is known, the players shall be discussed: their goals and their possible strategies. Finally, the learning techniques described in chapter 4 are used to have the mechanism learn to find the best strategy.

5.1 Meta Games

The field of adaptive mechanism design is relatively young. Recent work by Pardoe et al. [23, 24], as well as our own [14], has shown that even with simple learning heuristics, adaptation to a changing agent population outperforms a static mechanism. The scenario used in each of the works mentioned involves an iterated auction in which identical goods were sold in rounds to a bidder population. The mechanism could use the heuristics to determine the number of rounds per auction that would yield the most revenue.

These ideas may be generalizable to any repeated mechanism. By not just facili-

tating the adaptation of a single parameter of the mechanism, many could be changed between iterations. Using learning techniques such as those discussed in chapter 4, the mechanism can learn the optimal configuration of each of these parameters. This possibility of changing the way the mechanism behaves leads to strategic behaviour by the mechanism. To study this behaviour, the notion of a *meta-game* is introduced.

In a meta-game, the mechanism itself is a rational agent. The opponent of the mechanism player is the *population* of agents that play the mechanism, i.e. a single opponent. The difference between the mechanism as a player and the mechanism itself will often be evident from the context. However, to remove any potential ambiguity, the mechanism as a player will be referred to as the *mechanism player*. Similarly, the population of agents as an single individual player will be referred to as the *population player*.

5.1.1 Analysis of the Meta-Game

The meta-game is a sequential game that prescribes a rather straightforward sequence of moves. Both the mechanism player and the population player receive a type: a configuration $c \in \mathcal{C}$ and $\theta \in \Theta$ respectively. In traditional mechanism design, the mechanism player would receive its type from the mechanism's designer. The population player is handed a type by Harsanyi's Chance player.

The mechanism player moves first by announcing a social choice function and a payment rule, of which only one is needed as there is just one opponent. This corresponds to the situation were individual agents know the game they are about to play. The mechanism player uses a strategy that takes the mechanism's configuration (its type) and outputs the social choice function and a payment rule.

The population player replies by revealing its type, depending on the announced social choice function and payment rule. This type is the result of its mixed strategy and may not be truthful. The strategy set and the corresponding distribution are hidden from the mechanism player. Also there may be additional (hidden) information that the population player uses to form a revelation.

Finally, the meta-game will reward both the mechanism and population players according to the revelations they have made, i.e. evaluate the social choice function for the population player's type. The extensive form of this game is shown in figure 5.1. The role of the mechanism designer is shown, although he is not an active participant of this game, to visualize how the configuration of the mechanism is done. This act is to be taken over by a learning algorithm in order to adapt the mechanism online.

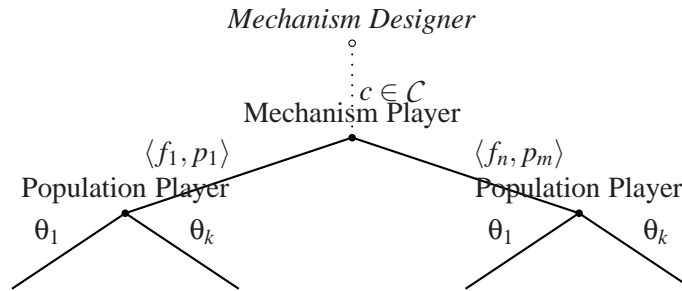


Figure 5.1: Extensive form of the meta-game.

To define the structure of the meta-game, three things are important: 1) the outcome space Φ , 2) the social choice function f and payment rule p and 3) whether the meta-game should be incentive-compatible. To find the answers, we will rephrase our intentions.

We are creating a mechanism that is to be repeated, and during this repetition it may adapt to the behaviour displayed by the agents within it by artificial learning. This learning allows us to study the mechanism as a rational agent through the meta-game. This is an abstraction of a real-world application and the outcome must therefore correspond to the outcome of the underlying mechanism. The meta-game should thus provide the same outcomes as would the underlying mechanism, i.e. $\Phi = O$.

The population player bases its type revelation on the type $\langle f, p \rangle$ reported by the mechanism player, and not on the social choice function of the meta-game. As the outcome of the meta-game should be equal to that of the underlying mechanism for the same population type, it makes sense to use the type reported by the mechanism player. Using the social choice function and payment scheme provided by the mechanism player will make truthfulness the dominant strategy for the mechanism player. This also implies that the meta-game has no influence on the truthfulness of the population player, or any other design constraint of the mechanism, such as budget balance. If the population player is to be truthful, it needs to get its incentives from the mechanism player rather than from the meta-game.

Definition 12 (Meta-Game). *A meta-game for a mechanism M*

$$\mathcal{G}^M = \langle (\mathcal{M}, \mathcal{P}), O, ((C, u_{\mathcal{M}}); (\Theta, u_{\mathcal{P}})) \rangle$$

consists of a mechanism player \mathcal{M} and a population player \mathcal{P} , a set of outcomes O equal to the set of outcomes for M . \mathcal{M} has a set of configurations containing an outcome function and a payment scheme $\langle f, p \rangle \in C$ and a utility function indicating the desirability of the outcome induced by the \mathcal{P} 's actions. \mathcal{P} has a type $\theta \in \Theta$ and a utility function. The meta game uses the outcome function and payment scheme provided by the \mathcal{M} .

5.1.2 Example Meta Game: $(N+1)^{th}$ -price auction

To make the reader a little more familiar to the correspondence between the mechanism being played by agents and the mechanism as a rational player itself, a simple example is introduced: a repeated $(N+1)^{th}$ -price auction. In this (simplified) setting the auction has N identical items to sell to a population of agents. Each iteration the number of items for sale (N) may differ.¹ Individual agents have a single strategy they use when bidding for one of the items. They are interested in obtaining only one item, preferably at the lowest price, but no higher than their private valuation v_i . This leads to a quasilinear utility function for agent i .

$$u_i(o_i, p_i) = \begin{cases} v_i - p_i & \text{if agent } i \text{ wins} \\ 0 & \text{otherwise} \end{cases} .$$

¹Obviously, this is a ridiculous setting when there is no time horizon. Rational agents would simply all bid 0, knowing that they will almost surely win an item. Knowing this, the example is exactly that: an example.

Game play is as follows: the agent places a bid (his revealed type b_i) according to his strategy and private valuation (his true type v_i), the mechanism orders all received bids and awards the N agents with the highest bids an item. All these “winning” agents pay the value of the $(N+1)^{th}$ bid. Agents leave the auction, a new iteration starts and other agents may join.

The meta game is only slightly different, as it is merely a change of perspective. The outcomes in the meta-game consist of a revenue (for the mechanism player), a vector o showing the allocation of the items and a price p to pay (for the mechanism player). The mechanism player’s utility is simply the revenue it gets; the population player’s utility is the sum of its agents utility: $\sum_i \mathbb{I}_i(v_i - p)$, where $\mathbb{I}_i \in \{0, 1\}$ is an indicator variable stating whether bid i was successful.

First, the mechanism player announces the output function and the payment rule, which depends on (and reflects) the number of items N for sale. It is now clear when an agent will win and what he has to pay, in this example the $(N+1)^{th}$ -price. Now the population replies with its type: a vector of bids b_i . The meta-game uses these bids as input for the social choice function and payment rule revealed by the mechanism player, resulting in an outcome: an allocation vector $\mathbf{a} \in \mathbb{I}^n$, and a payment, the $(N+1)^{th}$ -bid, are provided to the population player, and a revenue is passed to the mechanism player.

5.2 Properties of the Players

In order to study the meta-game, an analysis of the properties of the players is in order. These properties consist of the type space, the strategies and the utility functions a player can have.

5.2.1 The Mechanism Player

The repeated mechanism plays an outcome function and a payment scheme from the possible functions F and payment schemes P . Which of these are available depends on the constraints the mechanism must fulfil. This yields a space of configurations $C = F \times P$. In the auction example, the available functions and schemes all have the same form, differing only in a parameter N . The mechanism player first converts the revealed type, a bid-vector \mathbf{b} , to an ordered vector \mathbf{o} . The outcome function played is

$$f_N(\mathbf{b}) = \left\{ \begin{array}{ll} 1 & \text{if } b_i \geq o_N \\ 0 & \text{otherwise} \end{array} \right\}^{|\mathbf{b}|},$$

leading to the vector of indicator values and the payment scheme is $p_N(\mathbf{b}) = o_{N+1}$.

The utility of the mechanism player depends on the goal function of the underlying mechanism. The mechanism player wants to maximize this utility over time. In the auction example the utility for the mechanism player is the revenue it receives and the goal is maximizing the sum of revenues obtained in all iterations.

The size of the families F and p of social choice functions and payment schemes makes finding the right type to play very hard. Additional constraints on the (underlying) mechanism, e.g. incentive compatibility or budget balance, will narrow the search, but may also make it impossible to find any combination at all that satisfies

the constraints and optimizes the goal function, recall the impossibility results for the revelation principle in section 3.2.

To find a suitable type, the adaptive mechanism will need to learn from past experiences. The learning method should be able to find fitting functions for both the social choice function and the payment rule. In the arms race that results from a learning mechanism and an evolving population, the goal is to have the mechanism adapt faster than the population. The best way of fast online adaptation of a function is the use of function approximation. One way of function approximation has already been discussed in section 4.1.

The learning algorithm the mechanism player uses to adapt itself requires a “score” to quantify the success of the latest action, for which the utility of the previous iteration may be used. This function should also be extended however, when specific features are required in the underlying mechanism, e.g. incentive compatibility or budget balance. Without penalizing the loss of (one of) these features, the mechanism player may maximize its goal function but not be, for instance, budget balance. This can be done by setting the utility to 0 when this requirement was violated. Although this violation can easily be detected in the case of budget balance, other requirements may be more difficult, e.g. incentive compatibility.

5.2.2 The Population Player

The population player is made up of a, possibly large, number of agents. These agents are individual, independent and rational, implying that they make decisions individually and strategically. The agents are not required to remain playing in the mechanism indefinitely, but the population of these agent will be, at least in expectation: agents leaving the mechanism will be replaced by new agents entering. The properties of the population as a rational player, e.g. the utility function and the strategy set, will remain constant in expectation over time, even though the agents that make up the population are constantly changing. As the population player is a rational player, he may change his (mixed) strategy.

Between iterations, agents entering the mechanism may select a strategy based on results they have observed in earlier repetitions. This way, successful strategies will be favored over the less successful. As a single player, the population player can be modelled to act in much the same way as was described in section 4.2 on evolutionary game theory. The population has a mixed strategy which represents the distribution of (pure) strategies among the agents in the population. This mixed strategy will change over time as some pure strategies will be favored by the agents within the population.

The type space Θ is the collection of all possible types that the population player may reply with. This depends highly on the mechanism the agents are participating in. In the example this is a vector of positive bids of size equal to the number of bidders: $\Theta = \mathbb{R}^{+|b|}$. The revealed type $\theta' \in \Theta$ does not have to be truthful and is based on the type $c \in \mathcal{C}$ reported by the mechanism.

The utility received by the population player can be taken to be the sum of all individual utilities; in the auction example this would be $\sum_i v_i - p_i$ for all winning agents. This population utility can also be used as a measure for the success of the mixed strategy used by learning player.

Thus far, it is assumed the population player learns through evolutionary dynamics

as this model fits applications involving populations of human agents best. The repeated mechanism may also be used on other types of agents, such as artificial agents. Examples of these are file-sharing agents (e.g. in peer-to-peer systems) and nodes in a computer network (e.g. the internet). Each of these may require a different model for a population player. If most, or all, artificial agents use the same means of learning, e.g. reinforcement learning, the entire population may also be seen as learning through reinforcement learning, rather than through evolutionary dynamics. In the worst case, the population player may even become an adversarial opponent, with all agents working together to beat the underlying mechanism. Modelling the way an agent adapts is useful for simulation and testing the adaptive mechanism.

Chapter 6

Future Work

To further investigate the design of repeated mechanisms, a number of topics will require additional research. The reason for not including this research in the current work is that either the results are yet to be found or relevant literature was beyond the scope of the current research.

Repeated Games Repeated games offer new equilibria and their solution concept is a specialization of the Nash Equilibrium, introducing sequential rationality: any choice must not only be rational in this round, but also in rounds to come. The analysis of repeated games offers the one-shot-deviation principle, which allows the determination of (sequentially rational) subgame-perfect equilibria even in infinitely repeated mechanisms [18]. Investigation into a connection to evolutionary game theory may yield a useful correspondence between evolutionary stable states and these subgame-perfect equilibria. However, the utilities for finding subgame-perfect equilibria assume a static component game, i.e. the game that is repeated does not change. What happens to these concepts when the mechanism will learn and adapt.

Revelation Principle for ESS Although this may circumvent the use for an adaptive mechanism, it is interesting to know: Is there a revelation principle for evolutionary stable strategies (c.f. proposition 10). What are the (im)possibility results for this version of the principle? It would look like this:

Proposition 13 (Evolutionary Stable Revelation Principle). *Given a mechanism that implements a social choice function f in evolutionary stable strategies, then there also exists an incentive compatible mechanism implementing f in evolutionary stable strategies. Payments to agents are equal to the payments in the original mechanism.*

NP-complete outcome functions An interesting approach by Conitzer and Sandholm as discussed in section 3.2 (and [6]) features an NP-complete outcome function. This would make incentive compatibility no longer a requirement as finding a fruitful, non-truthful type to reveal NP-complete for the agents: finding a non-truthful type $\theta_i = \arg \max_{\theta'_i \in \Theta} f(\theta'_i, \theta_{-i})$ is an intractable problem. Is there a generic way to find such an outcome function and how does this relate this to a repeated mechanism.

Rationality of the Population Player A thorough justification of all players in the meta-game is needed, especially the population player. Is this player actually *rational* and if so, is it *strategic*? Both properties will need to be present to allow the meta-game to be analyzed as a proper game using game theoretic concepts. A truly strategic player may learn from past play by the mechanism and form a strategy that will provoke the mechanism player into playing more favorable actions in future repetitions, possibly at the cost of a lower present utility.¹

Learning Function shape Artificial learning focuses on selecting actions or on determining parameter values for a parametrized representation of the goal function. However, this requires the *shape* of this goal function to be known. Unfortunately, the mechanism player plays from a collection of functions, of which the shape is not necessarily known. Is there a way to still learn under these conditions?

Optimization Metric for Mechanism Player In order to measure of the effectiveness of a chosen outcome function and payment scheme pair, some metric needs to be found. One candidate is the Price of Anarchy, c.f. [33]. Can the potential function method be used for repeated mechanisms?

¹Thank you Adriaan for bringing this to my attention.

Bibliography

- [1] Dilip Abreu, David Pearce, and Ennio Stacchetti. Toward a theory of discounted repeated games with imperfect monitoring. *Econometrica*, 58:1041–1063, 1990. 9
- [2] Ken Binmore. *Game Theory - A Very Short Introduction*. Oxford university Press, 2007. 5
- [3] William E. Boyce and Richard C. DiPrima. *Elementary Differential Equations and Boundary Value Problems*. Wiley, 2000. 28
- [4] Andrew Bye. Applying evolutionary game theory to auction mechanism design. In *Proc. 1st IEEE conference on E-Commerce (CEC-03)*, 2003. 26
- [5] Vincent Conitzer. *Computational Aspects of Preference Aggregation*. PhD thesis, Carnegie Mellon University, July 2006. Chapter 6, 8. 3
- [6] Vincent Conitzer and Tuomas Sandholm. Computational criticisms of the revelation principle. In *LOFT*, 2004. presented orally. 19, 37
- [7] Peter Cramton. The efficiency of the FCC spectrum auctions. *The Journal of Law & Economics*, 41:727–736, October 1998. 3
- [8] Peter Cramton and Jesse A. Schwartz. Collusive bidding in the FCC spectrum auctions. *Journal of Regulatory Economics*, 17(17):229–252, May 2000. 3
- [9] Rajdeep K. Dash, David C. Parkes, and Nicholas R. Jennings. Computational mechanism design: A call to arms. In *IEEE Intelligent Systems*, volume 18, pages 40–47, 2003. 15
- [10] John C. Harsanyi. A general theory of rational behavior in game situations. *Econometrica*, 34(3):613 – 634, July 1966. 12
- [11] Josef Hofbauer and Karl Sigmund. *Evolutionary Games and Population Dynamics*. Cambridge University Press, 1998. 29
- [12] Josef Hofbauer and Karl Sigmund. Evolutionary game dynamics. *American Mathematical Society*, 40:479–519, June 2003. 27, 28, 30

-
- [13] Leslie Pack Kaelbling, Micheal L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996. 21
- [14] Tomas B. Klos and Gerrit Jan van Ahee. Evolutionary dynamics for designing multi-period auctions (short paper). In *Proceedings 7th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2008)*. IFAA-MAS, 2008. 3, 31
- [15] Ron Lavi and Chaitanya Swamy. Truthful and near-optimal mechanism design via linear programming. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Scienc*, pages 595–604, 2005. 18
- [16] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pages 157–163, New Brunswick, NJ, 1994. Morgan Kaufmann. 21
- [17] Michael L. Littman, Anthony R. Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: Scaling up. In Armand Prieditis and Stuart Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, pages 362–370, San Francisco, CA, USA, 1995. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA. 21
- [18] George J. Mailath and Larry Samuelson. *Reaped Games and Reputations, long-run relationships*. Oxford University Press, 2006. 9, 37
- [19] John Maynard Smith and George R. Price. The logic of animal conflict. *Nature*, 245:15–18, 1973. 25
- [20] Noam Nisan. Introduction to mechanism design (for computer scientists). In *Algorithmic Game Theory*. Cambridge University Press, 2007. 15, 17, 19
- [21] Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic Behaviour*, 35:166–196, 2001. 15, 19
- [22] Martin J. Osborne. *An Introduction to Game Theory*. Oxford University Press, 2003. 5
- [23] David Pardoe and Peter Stone. Developing adaptive auction mechanisms. *(SIG)ecom Exchanges*, 5(3):1–10, 2005. 3, 31
- [24] David Pardoe, Peter Stone, Maytal Saar-Tsechansky, and Kerem Tomak. Adaptive mechanism design: A metalearning approach. In *The Eighth International Conference on Electronic Commerce*, pages 92–102, 2006. 31
- [25] David C. Parkes. *Iterative Combinatorial Auctions: Economic and Computational Efficiency*. PhD thesis, University of Pennsylvania, 2001. 17, 18
- [26] Stephen Phelps. *Evolutionary Mechanism Design*. PhD thesis, University of Liverpool, 2007. 26

BIBLIOGRAPHY

- [27] Ariel Rubinstein. Equilibrium in supergames with the overtaking criterion. *Journal of Economic Theory*, 21:1–9, 1979. 9
- [28] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003. 21
- [29] Yuzuru Sato and James P. Crutchfield. Coupled replicator equations for the dynamics of learning in multiagent systems. In *Physical Review*, volume 67, page 015206. American Physical Society, Jan 2003. 30
- [30] Yoav Shoham, Rob Powers, and Trond Grenager. If multi-agent learning is the answer, what is the question? Levine’s Working Paper Archive 122247000000001156, UCLA Department of Economics, February 2006. available at <http://ideas.repec.org/p/cla/levarc/122247000000001156.html>. 21
- [31] Karl Sigmund and Martin A. Nowak. Evolutionary game theory. *Current Biology*, 9(14):R503–R505, July 1999. 26
- [32] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998. 21
- [33] Eva Tardos. Introduction to the inefficiency of equilibria. In *Algorithmic Game Theory*. Cambridge University Press, 2007. 38
- [34] Karl Tuyls, Dries Heytens, Ann Nowe, and Bernard Manderick. Extended replicator dynamics as a key to reinforcement learning in multi-agent systems. In *Proceedings of the 14th European Conference on Machine Learning (ECML03), LNAI Volume: 2837*, pages 421–431, 2003. 28
- [35] Karl Tuyls and Ann Nowe. Evolutionary game theory and multi-agent reinforcement learning. *The Knowledge Engineering Review*, 5:63–90, 2005. 21
- [36] Eric van Damme. The Dutch UMTS-auction. CESifo Working Paper Series CESifo Working Paper No., CESifo GmbH, 2002. 3
- [37] Hal R. Varian. Economic mechanism design for computerized agents. In *WOEC’95: Proceedings of the 1st conference on USENIX Workshop on Electronic Commerce*, pages 2–2, Berkeley, CA, USA, 1995. USENIX Association. 17
- [38] Jürgen W. Weibull. Evolution, rationality and equilibrium in games. *European Economic Review*, 42:641 – 649, 1998. 29

Appendix A

Referenced Videos

- Avrim Blum, available at http://videlectures.net/mlss08au_blum_org/

“The first part of this tutorial will discuss adaptive algorithms for making decisions in uncertain environments (e.g., what route should I take to work if I have to decide before I know what traffic will like today?) and connections to central concepts in game theory (e.g. what can we say about how traffic will behave overall if everyone is adapting their behavior in such a way?). He will discuss the notions of external and internal regret, algorithms for ”combining expert advice” and ”sleeping experts” problems, algorithms for implicitly specified problems, and connections to game-theoretic notions of Nash and correlated equilibria. The second part of the tutorial will be about some recent work on learning with similarity functions that are not necessarily legal kernels. The high-level question here is: if you have a measure of similarity between data points, how closely related does it have to be to your classification problem in order to be useful for learning?”
- Ariel Rubinstein, talk at NYU on John Nash, November 2003, available at http://gametheory.tau.ac.il/nashLecture/Ariel_Rubinstein.wmv

“The lecture includes critical comments on the meaning of Game Theory as well as personal reflections on Game Theory, the story of John Nash, and the recent film ’A Beautiful Mind’. In preparation for the lecture, the audience is asked to spend a few minutes responding to some ”problems” presented at a specially designed page. Some statistics on the results of this exercise are presented in the lecture (though individual answers will remain confidential).”
- Nicole Immortica, Google Tech Talks, April 13 2007: “Challenges in the Design of Sponsored Search Auctions”, available at <http://youtube.com/watch?v=JzRHijdE0mk>

“Since its inception in the 1980s, the popularity of the Internet has been growing exponentially, resulting in a mass of shared knowledge and fast, cheap communication. Hand-in-hand with these developments, we have seen the birth of a plethora of new valuable systems and services ranging from web search and email to blogging and social networking sites. Perhaps the most essential system

for monetizing such web services is online advertising. In this talk, I will first present an overview of the most common market mechanism for online advertising, namely pay-per-click auctions. I will then discuss some of the challenges in the design of these auctions...”

- Noam Nisan, Google Tech Talks, August 15 2007: “Algorithmic Mechanism Design”, available at <http://video.google.com/videoplay?docid=6121409064231775355>

“One of the challenges that the Internet raises is the necessity of designing distributed ... alle protocols for settings where the participating computers are owned and operated by different owners with different goals. Over the last decade or so there has been much research that aims to address these issues using ideas taken from the micro-economic field of mechanism design. In this talk I will survey the current state of the field: how mechanism design is applied in computational settings, how far can classical ideas go, and what are the challenges for further research. Among the applications discussed will be combinatorial auctions, cost sharing, scheduling, and routing in networks.”

- S.V.N. Vishwanathan, Google Tech Talks, March 25 2008: “Optimization for Machine Learning”, available at http://youtube.com/watch?v=_U2Sn67Yrf0

“Regularized risk minimization is at the heart of many machine learning algorithms. The underlying objective function to be minimized is convex, and often non-smooth. Classical optimization algorithms cannot handle this efficiently. In this talk we present two algorithms for dealing with convex non-smooth objective functions. First, we extend the well known BFGS quasi-Newton algorithm to handle non-smooth functions. Second, we show how bundle methods can be applied in a machine learning context. We present both theoretical and experimental justification of our algorithms.”

- Seshadhri Comandur, Google Tech Talks, March 14 2008: “Adaptive Algorithms for Online Optimization”, available at <http://www.youtube.com/watch?v=kioqKenKkgE>

“The online learning framework captures a wide variety of learning problems. The setting is as follows - in each round, we have to choose a point from some fixed convex domain. Then, we are presented a convex loss function, according to which we incur a loss. The loss over T rounds is simply the sum of all the losses. The aim of most online learning algorithm is to minimize *regret* : the difference of the algorithm’s loss and the loss of the best fixed decision in hindsight. Unfortunately, in situations where the loss function may vary a lot, the regret is not a good measure of performance. We define *adaptive regret*, a notion that is a much better measure of how well our algorithm is adapting to the changing loss functions. We provide a procedure that converts any standard low-regret algorithm to one that provides low adaptive regret. We use an interesting mix of techniques, and use streaming ideas to make our algorithm efficient. This technique can be applied in many scenarios, such as portfolio management, online shortest paths, and the tree update problem, to name a few.”