# Master thesis 2011
# Implementation, test, and analysis of some solution concepts for repeated games and imperfect information games

By
Christian Juul Kisum, s042665

Supervisor
Valentin Goranko

Department of Informatics and Mathematical Modelling
Technical University of Denmark

IMM-M.Sc.-2011-90

23th of December 2011

**Abstract**

The work presented in this thesis covera an adaption of regret minimization as a solution concept for 2-player non-cooperative strategic form games. And in addition an adaption of the regret minimization (Average regret minimization) proposed by V. Garanko. This solution concept is experimented with and compared to iterated regret minimization, as proposed by Halpern and Pass in 2009.
This is solved by conducting a series of experiments with the different solution concepts. The experiments proved that average regret minimization has the same time complexity, but find the results in a single iteration. Furthermore the results of average regret minimization compared to those of iterated regret minimization. The results were very similar. This implied that average regret minimization was a better solution than iterated regret minimization.

**Resumé**

Denne afhandling omhandler en ny version af "regret minimization" som er et løsningskoncept til "2-player non-cooperative strategic form" spil. Den nye version af "regret minimization" (average regret minimization) er blevet foreslået af V. Garanko. Denne er der blevet udført eksperimenter med og sammenlignet med "iterated regret minimization", som blev foreslået af Halpern og Pass i 2009.
Dette er løst ved at gennemføre en række eksperimenter med de forskellige løsningskoncepter. Forsøgene viste at "average regret minimization" havde samme tidskompleksitet, men skulle kun bruge en enkelt iteration til at finde resultaterne. Yderligere blev resultaterne af "average regret minimization" sammenlignet med "iterated regret minimization". Resultaterne var meget ens. Dette viser at "average regret minimization" er en bedre løsning end "iterated regret minimization".

**Acknowledgements**

I am heartily thankful to my supervisor, Valentin Goranko, whose encouragement, guidance and support from the initial to the final level enabled me to develop an understanding of the subject.

I would like to thank my girlfriend Maja B. Flindt for her support and help with proof reading of the report in the final stage.

Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of the project.

Christian Juul Kisum.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

The field of game theory was started in the years after the end of the Second World War by John von Neumann, who also made a great breakthrough in the area of algorithms. In 1944 he wrote "Theory of Games and Economic Behavior"[7] in collaboration with Oscar Morgenstern, which is considered as the birthplace of game theory, utility theory and microeconomics. This book described equilibria in Zero-sum games. His collected works were before his time, but after the internet was introduced on a more wide scale, the area of game theory flourished and developed with a thunderous pace due to the effect the Internet had on world economics. This happened in the years after the Cold War in which he participated as a soldier. Fortunately, the war itself was mostly fought by using game theory to analyze and optimize the arms race, which entailed that no wide scale conflict occurred.

The wide field of game theory focuses mostly on analysis of human and animal behavior. One of the most commonly facets which game theory is used for is the method of how an entity chooses the optimal strategy in order to maximize a profit or minimize the cost of an action. The situation can be literally anything, any situation with any number of actions. The known methods to analyze the problems are, however, not in any regards infallible. Some situations can be modeled by the known methods, but do not yield results which reflect the actions chosen in the real world.

One of the areas of Game Theory concerns definition and analysis of solution concepts. The term **Solution Concept** covers an idea of how to play a game by setting some premises for choosing and adopting strategies. A solution concept can lead to multiple strategies for a given game, if more than one strategy is equally good according to the premises of the solution concept.

One of the key solution concepts in game theory is the Nash Equlibrium which was proposed by John Forbes Nash, but the concept itself predates his contribution, already in 1838 the Cournot Equilibrium was introduced, which in general is the same as a pure Nash Equilibrium. It was in Nash's Doctorate in 1950, that specified the definition and properties of equlibria, which defined the mixed Nash equilibrium, and is still today one of the key aspects of game theory. It was because of his doctorate the concept of equilibria was named after him[5].

The idea behind **Nash Equilibrium** is to find states in a game where no player can get a better outcome by choosing another strategy if he knows that the opposing player follows this Nash Equlibrium. The strategies that this solution concept dictates can both be a pure strategy where a single action is chosen, or a mixed strategy which is a probability distribution between a set of actions. This

solution concept has some drawbacks, one of them is the complexity of finding the strategies. It has been proven that finding Nash Equilibria is in the complexity class PPAD. This class in mainly for problems for which the solutions are hard to find, but easy to verify and it is guaranteed that a solution will always exist [5]. Another problem with Nash Equalibria is that it is not always a good solution for a problem. This is most obvious in games like the centipede game or Traveler's Dilemma. In these games the recommended strategy to stop as early as possible or to bid as low as possible. This results in one of the worst possible rewards. This is because Nash Equilibria only works if all players adopt this solution concept. For a more specific description of these games, refer to section 3.

Since Nash's contributions, many additions and adaption's have been added, but the mixed Nash Equilibrium is still the backbone of the area of game theory. One of the more recent additions is the usage of a regret minimizing solution concept instead of Nash Equlibria. The idea of using regret minimizing dates back to the years after the breakthrough of Von Neumann, where Savage [6] and Niehans [4] introduced the idea within the area of decision theory. Even though the regret minimization has been used in many years, it was first in the late 80's where regret minimization was applied to game theory by P. B. Linhart and R. Radner in [3]. One of the more recent adoptions of regret minimization is the iterated version described by J. Halpern in [2]. This article and its concepts are what most of the work in this thesis focuses on. The concepts build upon the regret minimizing solution concept which is then adapted to an iterated version. It is then tested on different games where Nash Equilibria do not yield any meaningful results. The idea behind regret minimization is to minimize the possible regret when choosing a strategy. The regret is how much higher a reward a player could have gotten if he had chosen another strategy when the choice of the opposing players has been revealed.

From the regret minimization an adapted solution concept can be created, the average regret minimization. This concept was proposed by V. Goranko. The main idea is, instead of selecting the strategy a strategy which yields the lowest possible maximal regret, the strategy with the lowest average regret is selected. This concept will later be analyzed both theoretically and empirically. The main goal of this thesis, is to examine the results of average regret minimization and how it relates to iterated regret minimization. This is done in order to see if average regret minimization is a better solution in respect to complexity and outcomes. What is needed in order to draw any conclusion is explained in chapter 4.

Throughout this thesis different games are mentioned. The players of these games are all referred to as male in order to simplify the description.

## 1.1 Applications

Game theory has many applications of a wide variety. Its applications stretch from economy to philosophy and from artificial intelligence to political science. Below is a list of some of the more widely known applications.

**Description and modeling** is a use of game theory to describe the behavior of a human population. This is done by making a model which describes situations where a population or larger assembly of people must make some kind of choice. These groups are in some studies perceived as trying to maximize their outcome, and is thereby following the pattern seen in game theory. Several experiments disproves that this is the case in all games, many games have been proposed, where humans do not follow a equilibrium.

**Prescriptive or normative analysis** is used as a tool as to how an entity should behave in a given situation in order to optimize its outcome. This has also been argued that this is not optimal in

all situations. An example could be the Prisoner's Dilemma. By following the equilibrium, one is not getting the best outcome, which could be obtained by cooperating with the other player.

**Economics and business** is the widest use of game theory in practice. Here it is used for predicting how to behave in some economic market. It is also used for creating auctions and other mechanics to optimize the outcome of sales.

**Political science** is the use of game theory in political decisions; this can both be how to fairly divide resources or how politicians react in an upcoming election.

**Biology** is using game theory in many fields. It can be used to describe evolution by looking into mutations being a way to maximize an outcome in some situations. Other things, like the sex ratio of the children born can also be explained by game theory. Lastly the behavior of big animals to small microbes can also be modeled by using game theory.

**Computer science and logic** is using game theory in multiple fields. It is used to optimize the way distributed computations are being performed in a dynamic environment by using online algorithms. Another more direct use of game theory is to control autonomous agents.

**Philosophy and psychology** is also using game theory to describe how the mind reasons in decision making.

**Warfare** is a more dire usage of game theory. It can be used to cut the cost in materials or human life and how to attack the enemy in order to maximize the damage.

Some of the above fields are overlapping. As an example, the psychological usage of game theory of describing decision making is in fact what is also done in political science, just on a bigger scale. The list contains many of the fields in which game theory can be used, but the list is not complete, since game theory in some form can be used in nearly every kind of science.

## 1.2 Reading Guide

This thesis consists of 7 chapters. The first is the introduction which you are reading right now. The following two sections contain the theoretical base for the remaining thesis, where chapter 2 concerns the more basic theory of game theory and the used terminology. Chapter 3 concerns the more specific games and solution concepts used for the topic of the thesis. Chapter 4 then specifies what the aim of the thesis is and how it is handled. Chapter 5 explains how the implementation of the testing tool for the execution and evaluation of the experiments. The next chapter is the core of the thesis which contains the experiments and analysis which examines the subject of the thesis. Chapter 7 contains the final conclusion which summarizes all the results of the thesis.

# Chapter 2

# Preliminaries

This chapter defines the general areas of game theory. These areas consist of some of the basic concepts and terminologies which are used as the foundation for the further and more specific work in within this thesis. This chapter is divided into three sections, representations, game types and the basics of solution concepts. The specific solution concepts and games used within this thesis will be further described in chapter 3.

## 2.1 Game representations

Before the different representations can be discussed, a definition of the term *game* must be covered. Many different definitions of games exist, but most of them share some traits. In [5], the definition of a basic single-move game is defined as: "Game theory aims to model situations in which multiple participants interact or affect each other's outcome". This citation can be used as a foundation for a bottom up analysis of what a game consists of. The key components of a game are: the situation, the participants and the interactions. The situation can be referred to as the set of all possible strategies, the participants as the players of the game, and the interactions as the outcome.

The **players** of a game are defined as a set of $n$ entities $(1, 2, ..., n-1, n)$. A player can be many things, the most original idea is that it is perceived as human; however, it can also be other types of autonomous biological entities like animals, insects, bacteria and other microbes. It can even be more abstract like a collection of entities acting in a coalition. After the computer and the internet became more common, a new type of entity was introduced, the artificial entity, based on a program or algorithm. The most common occurrence is the autonomous agent which is controlled by artificial intelligence.

The **strategies** are the set of possible actions for the players. These are defined as set $S_i$ for each player $i$. Each strategy $s_i$ of a player maps to some kind of action $a$ such as bidding a value *bid*. The collected choice of all the players can be represented as a vector of the strategies $s = (s_1, s_2, ..., s_{n-1}, s_n)$. The collection of all possible strategy vectors is defined as $S = \times_i S_i$. The strategies define all the possible actions a given player can undertake in a given situation, which in theoretical situations is a perfect model; however it can be very hard to model a real life situation, because most often an unlimited number of strategies are available. An example of this could be the following situation: "Two people travel down a narrow corridor in which they cannot pass each other, they both have the following

choice, going forward or backward". In the modeled situation, both players can choose between two strategies, but in a real life situation, both players could perform an unlimited number of different actions, crawl between the legs of the other player, jump over him, stand still, eat a banana etc. So in real life, any number of unique strategies can be chosen, and therefore, in order to model a real life situation, some assumptions must be made in order for relevant strategies to be selected.

The final part of a game is the **outcome**. This is tightly tied to the selected strategies vector, for each vector there is an outcome defined. The outcome can be defined in two ways, either as utility or cost. The **utility** is positive enforcing and **cost** is negative enforcing. This leads to two functions which maps from the strategy vectors to the outcome, $u_i \colon S \to \Re$ and $c_i \colon S \to \Re$ respectively. For each strategy the function $u_i$ returns a utility for the player $i$. Utility and cost are completely interchangeable, since $u_i = -c_i$. The utility could be how much money a player wins from a game, and cost could be the number of years a player should be imprisoned.

These three components add up to a complete game. A game can be either a single or multiple iterations of a given situation and even an unknown or infinite number in some cases. A sample game could be the **Prisoners Dilemma**. This is a classic game used as an example showing different features of game theory. The situation of the game is as follows: Two collaborating criminals have been arrested and now need to decide whether or not to cooperate with the authorities. The punishment they each receive is tied to what they tell the police and what the other prisoner says. Each of the prisoners now have the choice to either keep silent $S$ or confess $C$ to the crime. These choices are their strategies from which they have to choose. The outcome is codependent on what they choose, if they both keep silent, they both receive a 2 year sentence. If they both confess they each receive a sentence of 4 years. If one confesses and the other keeps silent, then the one confessing gets a "discount" on his sentence and only receives 1 year, but the other receives a full sentence of 5 years.
With all these parts of the game defined and it can be represented in a more formal matter. Firstly the **strategic form**. This is great for smaller games because it is visually intuitive. It is also easy be use for representing bigger games with several players, but it becomes less easy to represent visually. This is because of the structure. It consists of a matrix, where the dimension is the same as the number of players. Each row of the matrix represents a player and matching strategies. Each element of the matrix represents an outcome made up of all the strategies which intersects in the element. The Prisoner's Dilemma can be written in strategic form as seen in fig. 2.1. This visual representation is also called the **payoff/cost matrix**. The first number in each element is the cost for player one, which is the row player, and the second number is the cost for player two, the column player.

| | $C$ | $S$ |
|---|---|---|
| $C$ | $4, 4$ | $1, 5$ |
| $S$ | $5, 1$ | $2, 2$ |

Figure 2.1: The Prisoners Dilemma in strategic form

Secondly there exists another widely used representation. This is the **extensive form**. In this representation, the game is presented as a graph instead of a matrix. The graph is tree-shaped and basically works like a game-tree. It has a root in which the state has not changed yet and is before any of the actions of the players have been executed. From this, a number of edges protrude which equals the number of strategies player one can choose between. The vertices to which the edges lead are imaginary states where player one has performed his action and is waiting for player two, but since player two is unaware of players one's action, then he does not know in which state he is. In most cases this state does not exist, but is only a part of the representation. From each of the vertices then protrude a number of edges equal to the number of strategies of player two. In this form a game with multiple

players can easily be represented. Another level in the tree is simply added for each player, and the edges from each of these levels must correspond to the number of strategies of the player. The leaf level of the tree is the end state where all the players have performed their action. All the previous states, excluding the root, usually do not represent real states of the game, because the chosen strategies of the players are all executed simultaneously. Therefore, players do not know the strategy of the other players until after they themselves have chosen one. If the state represents real game states, then the game type is changed, it is no longer a simultaneous move but a sequential game. An example of the extensive form can be seen in fig. 2.2.

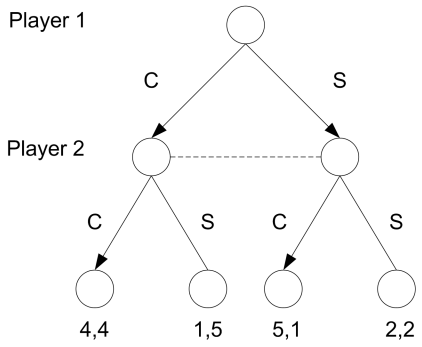


Figure 2.2: The Prisoners Dilemma represented in extensive form

The two game-representations given above are not the only ways to represent a game, but are considered to be the two most widely used game-representations. One other example is the Compact Represented Game. This is usually used when dealing with very big games. An example could be a game with an infinite number of players where the utility depends on only a subset of the other players, then the game can be represented much more compact.

The strategic and extensive form each has advantages and drawbacks. It depends a lot on the type of game they represents. The strategic representations advantage is in the ability to represent small games simplistically. The extensive representation on the other hand is better suited for bigger and more complex games with more than two players.

## 2.2 Game types

Each of the representations described above can be used to represent several types of games. The most widely used game is the "simultaneous move single-shot two player game". This can easily be represented by each as seen with the Prisoner's Dilemma. This simple game can be repeated; in each iteration of the same game is then played. This can change how the strategies are chosen. As it is the same game which is played each round, the game can in each iteration easily be represented by the strategic form, but in order to represent the entire game, the representation needs to be expanded. The strategic form represents a single shot game, so in order to represent the entire game, the available strategies need to encompass the choices for all the iterations. Looking at the Prisoner's Dilemma repeated twice, each player now have 4 strategies to chose from $(CC, SS, CS, SC)$, so the game is now $4 \times 4$. This results in the game matrix expand exponentially. The size of extensive form games grow in the same manner due to the tree-shape.
Most games based on real life applications have a finite size, but some theoretical games can contain an infinite number of players, strategies and/or iterations. These games must be represented by another

structure, and generally be handled differently from normal games.

A different type of game is the games with incomplete and/or imperfect information. The two terms cover two completely different features of a game. When a game has **incomplete information**, then it relates to the information that the players have, or more specific to the information that the players do not have. In these games some can be information hidden from the players, so the playesr do not know the complete structure of the game. This could for instance be the values of outcomes. This type of game is also called **Bayesian games** [5].
In a **imperfect information** game the players are missing precise information about the state of the game. This is seen in simultaneous move games. The players do not know how the other player acted before they act themselves and do therefore not know in what state the game is. This is the case in strategic form games.

All games can have certain features which make them part of a subgroup for which more specialized algorithms can be used. Two common features are zero sum games and symmetric games. In **Zero-Sum games**, the sum of each outcome is 0, hence the name, so that the utility of one player is the cost of the other: $u_1 = c_2 = -u_2 = -c_1$. The other feature is the **symmetric games**. In these games, the strategies of the players are identical. The outcomes for each of the players Is considered symmetric, hence the name. Given a matrix $A$ for player I, the matrix is the payoff matrix were only his utilities are present. The matrix $B$ for player two must then be $B = A^T$. An example of a symmetric game is the Prisoner's Dilemma.

Most of the games investigated until now have contained two players, but games can contain any number of players greater than one. A game with only one player is no longer a theoretical game problem. It is far simpler because the player only needs to choose the strategy which yields the best utility, since by removing the other players, all uncertainty is also removed. When adding more players, the uncertainty raises and the size of the game, in most cases, increases exponentially. Both game representations mentioned previously can be used to represent games with multiple players, the strategic form is just an $n$-dimensional matrix where $n$ is the number of players. In the extensive form, the tree's number of levels are $n + 1$ because of the root.
The order of the players is in the general case indifferent, because they are executing their strategies simultaneously. However, in some games this is not the case, in these games the actions are instead executed sequentially, which causes the game to change again. The game is still the same for the first player, but since he acts before the other players, they get to react on his action and thereby the game changes due to the uncertainty of the first player is removed. A game where one player chooses an action before all the other players is called a **Stackelberg Game** [1].

For the purpose of this thesis an assumption is taken for the games described here, which is that all players do not communicate and are all acting selfishly. The games are then called **non-cooperative**. If the players cooperate and coordinate their moves, then entirely different solutions must be used.

## 2.3 Basics of solution concepts

To describe the basics of solution concepts, the Prisoner's Dilemma is used once again. The game is seen in strategic form in fig. 2.1. A **solution concept** determines the manner in which the strategy is chosen. The result of a solution concept can either be a single strategy or a probability distribution

| | $H$ | $T$ |
|---|---|---|
| $H$ | $1, -1$ | $-1, 1$ |
| $T$ | $-1, 1$ | $1, -1$ |

Figure 2.3: The Matching Pennies game in strategic form

over multiple strategies $s_i = p_1, p_2, ...p_n$ where $n$ is the number of possible strategies.

### 2.3.1 Dominated strategies

The most basic solution concept is **elimination of dominated strategies**. One strategy can be dominated by another, if for all possible outcomes the strategy yields a better utility or cost. The notation of $u_i(s_i, s_{-i})$ is used to describe this property. $s_i$ is the strategy chosen by player $i$, $s_{-i}$ is the choice of all other players. $s'_i$ is all other possible strategies of player $i$ and $s'_{-i}$ likewise for the other players. In order for a game to dominate all other strategies it must fulfill the following equation $u_i(s_i, s'_{-i}) > u_i(s'_i, s'_{-i})$. When all utilities of a strategy is higher than another it **strongly dominates** it.
A less strict definition is when strategies are only **weakly dominated**. A strategy is weakly dominating other strategies if all the utilities are at least as good. The formal definition is $u_i(s_i, s'_{-i}) \geq u_i(s'_i, s'_{-i})$. A strategy will often weakly dominate more strategies than strongly dominate.
An example of strong domination can be seen in the Prisoner's Dilemma. For each of the players, the strategy of Confess is better than Silent, no matter what the other player chooses. If the opposing player chooses to confess it is better if the player confesses, since it yields a cost of 4 instead of 5. If the other player stays silent, then it is still better to confess, since it would yield a cost of 1 instead of 2. So in all situations, it is better to confess. If both players adopt this strategy it will result in a cost of 4 for both players. This is highly unusual, since the outcome for $(S, S)$ yields a better outcome for both players. Most games do not have a dominant strategy, and hence a another solution concept is needed.
A less strict version of using dominated strategies as a mechanism for choosing optimal strategies is the **iterated elimination of dominated strategies**. In each iteration strategies get removed if they are dominated by at least one other strategy. This is repeated until no more strategies can be removed. It is easily shown, that not all games results in a single strategy for each player. In some games no strategies are dominated by others. In these situations the elimination of dominated strategies cannot be used to find optimal strategies. A game which displays this could be the Matching Pennies game which is shown in fig. 2.3.

When iteratively removing strategies the result is only deterministic when removing strongly dominated strategies. When removing weakly dominated strategies the result can vary depending on which order the strategies are removed.

### 2.3.2 Nash Equilibrium

The next solution concept is that of **Nash Equilibrium**. This is much less strict than dominant strategies and is able to find an optimal strategy in every game. A Nash Equilibrium is defined as a state where none of the players can get a better outcome by diverting from the strategies. The equilibrium can either result in a set of pure or mixed strategies. A pure strategy is when a player is certain to choose a specific strategy, while a mixed strategy is when the player has a chance to choose between

multiple strategies. The Prisoner's Dilemma can be used to display a pure Nash Equilibrium. Each of the outcomes must be evaluated in order to find all Nash Equilibria. Firstly looking at the outcome $(S, S)$. In this situation, both player can improve their outcome by diverting to the $C$ strategy, and therefore this outcome cannot be a Nash Equilibrium. Secondly looking at $(C, S)$, it is seen that this is not a Nash Equilibrium because player 2 can improve his outcome by diverting to $C$. The same goes for $(S, C)$ since the Prisoner's Dilemma is a symmetric game, the roles of player 1 and 2 have simply changed. The last outcome $(C, C)$ is the only Nash Equilibrium. This is because either of the players would not gain a better outcome by diverting to another strategy.
It is not possible to find a Nash Equilibrium in all games by the previously mentioned method. An example could be the Matching Pennies game. In each outcome, one of the players could improve his outcome by diverting. To find a Nash Equilibrium, the concept of mixed strategies needs to be introduced. Instead of choosing a specific strategy, a player selects multiple strategies and a probability distribution between them. The idea behind a mixed Nash Equilibrium is the same as the pure equivalent. No players can obtain a better outcome by diverting from the probability distribution. A mixed Nash Equilibrium can be seen in the Matching Pennies game. If both players randomly choose each strategy with a probability of 0.5, then a stable state is obtained. The expected outcome is 0 for both players, and cannot be improved by diverting, the probability distribution between the strategies consists of a Mixed Nash Equilibrium.

### 2.3.3 Other solutions

Many new solution concepts have been introduced through the years. Backward induction and Subgame-perfect equilibrium should be mentioned as they are some of the more widespread solution concepts. One of the more interesting, but more uncommon, is the Regret minimization and its adaption's. This will be analyzed to its full extent in a later chapter. The basic idea behind regret minimization, is that the best strategy to choose, is the one where the player has the least regret about his choice.

# Chapter 3

# The games and the solution concepts

This chapter defines the different games treated in this thesis and how the different solution concepts work, which algorithms they use and their complexities. The different games and solution concepts which are defined in this chapter are all going to be a part of the experiments of this thesis. The first section concerns the games where each of the games used for the experiments are described and formalized as a strategic form game. The different solution concepts are described and an algorithm is given for how to implement them. Lastly two sections are given with a solution concept each for mixed strategies and repeated games.

## 3.1 The Games

For the experiments in this thesis a range of different games are used. All games are taken from [2]. Each of the games are first described as a real world situation, and then formally described as a game in strategic form.

### 3.1.1 The Traveler's Dilemma

The setting is as follows. An airline has lost the luggage of two passengers. The airline now offers the passengers compensation for their lost properties. Each of the passengers must give a sealed bid between 2 and some maximal bid value $bid_{max}$. The passenger $i$ who has the lowest bid $bid_i$ then gets paid $bid_i + p$ where $p$ is a reward for the passenger who has the lowest bid. The other passenger gets paid $bid_i - p$ where $p$ then is a punishment. If both the passengers have the same bids, they both get that value without any reward/punishment.
The situation can then be transformed into a game in strategic form. It is trivial to see that the players are the passengers and their strategies are their bids. Each of the players have $bid_{max} - 2$ strategies named $[2, 3, ..., bid_{max}]$. The utilities of the players can easily be defined as in equation 3.1. A strategic form game can then be created from this information.

$$u(n,m) = \begin{cases} (n+p, n-p) & \text{for } n < m \\ (n,n) & \text{for } n = m \\ (n-p, n+p) & \text{for } n > m \end{cases} \tag{3.1}$$

### 3.1.2 The Centipede Game

This game has two players. The game runs for a number of *rounds*. In every odd round player one chooses whether or not to continue or end the game. In every even round player two must consider whether or not to continue. In round number one a stack of money is presented. When a player chooses to end the game he gets the pile of money and the other player receives the same amount minus some penalty $p$. Every round the amount of money increases by some function $f(r)$. The penalty equals the amount $p$ previous rounds. An important mechanic in this game is; if a player $i$ continues in some round $r$ then the following round the prize must be less if the other player chooses to stop, ie. $f(r) > f(r+1-p)$. This is the main problem of the game, every time a player chooses to continue, he has a slight risk to lose some of the current prize. If this mechanic did not exist, then the game would be rendered trivial because it would always be better to continue. The only thing which remains to be defined is the function $f(r)$. It has two features to which it must uphold, The first is the already specified $f(r) > f(r+1-p)$, the second is that the function must always increase.
A formal game can now be created. From [2] two different functions $f(r)$ are defined. The first is linear increasing and the other is exponential increasing. The game structure is very easy to define as a extensive form game, hence the name. The structure looks like a centipede because it has a long chain of nodes with a "leg" protruding from each node. An example can be seen in fig. 3.1.

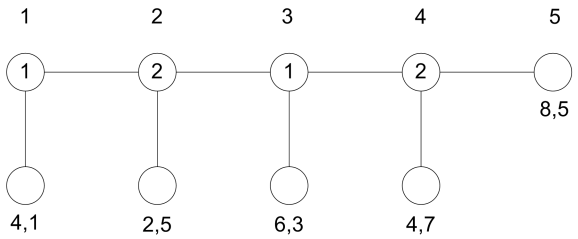


Figure 3.1: A Centipede Game represented in extensive form

This game then has to be converted into a strategic form game. The most basic method to do this is by creating strategies where all possible choices are defined. Meaning that each strategy should encompass information about what the players would do in each round. Both players would in this case have four different strategies. The game would contain a lot of needless information, because if a player chooses to stop in some round, it has the same outcome no matter what he does in later rounds. This leads to each player only needing a single strategy, where he for each round has the option of stopping the game. Besides this an additional strategy is needed for each player. This is in order to cover the situation where both players never stop. The centipede as seen in fig. 3.1 is shown as a strategic form game in table table:centipedestrategic.

Table 3.1: A Centipede Game in strategic form.

| | 2 | 4 | 6 |
|---|---|---|---|
| 1 | 4,1 | 4,1 | 4,1 |
| 3 | 2,5 | 6,3 | 6,3 |
| 5 | 2,5 | 4,7 | 8,5 |

The utilities of the game are dependent on the function $f(r)$. The function is defined in two different ways in the article [2]. The first is where the utilities grow exponentially $f(r) = 2^r$. The other is where they grow linearly $f(r) = r$. From this it can be seen, that in order to maintain the two features described earlier additional details are required, if not the following must hold $p > 1$. For the linear game this condition can simply be set, allowing no games with $p < 2$. In the exponential game this is handled by simply giving the player which stops the earliest an extra bonus of 1.
This information can then be transformed into a strategic game. The utilities can be seen in equation 3.2.

$$\begin{aligned} u_{exponential}(n,m) &= \begin{cases} (n, n-p) & \text{for } n < m \\ (m-p, m) & \text{for } n > m \end{cases} \\ u_{linear}(n,m) &= \begin{cases} (2^n + 1, 2^{n-p}) & \text{for } n < m \\ (2^{m-p}, 2^m + 1) & \text{for } n > m \end{cases} \end{aligned} \tag{3.2}$$

### 3.1.3 The Bertrand competition

In this game, the two players each take the role of a company. Each of these companies have *items* of wares for sale. They can each set a price *bid* for the wares which can be in the range $[0, 1, ..., bid_{max}]$. They now each set a price for the wares without knowing what the opponent is bidding. If one of the players bid lower than the other, then he sells all his wares at the price and the other player sells none. If they bid the same, they sell half their wares to the price.
This can rather easily be converted into a strategic form game. The strategies are the possible bids. So each player has $bid_{max} + 1$ strategies ranging $[0, 1, ..., bid_{max}]$. The utilities are found by looking at whom has the lowest bid or if they have the same. This can be seen in table 3.3.

$$u(n,m) = \begin{cases} (n \cdot items, 0) & \text{for } n < m \\ (\frac{n \cdot items}{2}, \frac{n \cdot items}{2}) & \text{for } n = m \\ (0, m \cdot items) & \text{for } n > m \end{cases} \tag{3.3}$$

### 3.1.4 The Nash Bargaining Game

The Nash Bargaining Game as the following setting where two players try to divide some goods. They can each ask for any value between 0 and the full amount, but if the sum of their request is bigger than the number of goods $max_{sum}$ none of them will receive any. The goods are held by some third party and the bids are given without knowing what the other player asks. If the sum is less than the number of goods each of the players receive what he asked for.
This can then be structured into a strategic form game. Each player has $max_{sum} + 1$ strategies, one for each possible bid. The utilities are then simply either 0 or the value of their bid, depending on the sum of their bids. This is formalized in equation 3.4.

$$u(n,m) = \begin{cases} (n,m) & \text{for } n + m <= sum_{max} \\ (0,0) & \text{for } n + m > sum_{max} \end{cases} \tag{3.4}$$

## 3.2 Solution concepts

This section contains the different solution concepts which will be used for further analysis and experimentation. First the different concepts are defined and then a specific algorithm is given for each concept. The complexity of the algorithm is then analyzed.

### 3.2.1 Mixed strategies

All the solution concepts in this section share a commonality, they are all mixed strategies. They are all a special type of solution concepts, since they do not look at the outcome. The results are purely based on the number of strategies, and some parameter. These solution concepts are created in order to mimic the behavior of some kind of natural opponents, specific for some of the different games treated in this thesis. For the four games in this thesis three different concepts are needed. Each of them is defined in its own section. None of these solution concepts are based on literature and are all created and named from how I expect players are playing the game in a real life situation.

**Uniform:** The solution concept here is based on a player who selects any of the pure strategies with equal possibility. The name uniform is given to this solution concept because of the probability distribution being uniform. To create the mixed strategy each of the strategies are simply assigned the same probability. For a player with $n$ strategies each of the strategies is given a probability $\frac{1}{n}$.

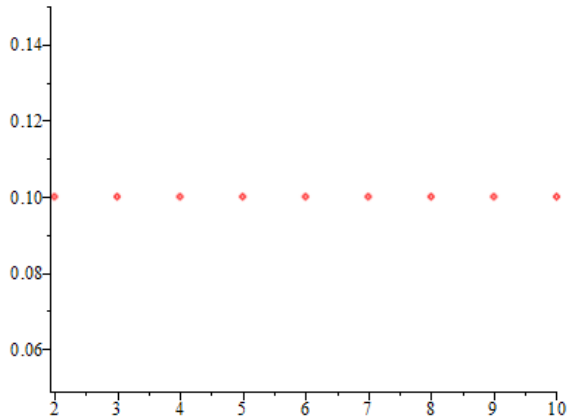


Figure 3.2: An example of the probability distribution of the uniform solution concept

**Cymbal:** The idea behind this solution concept is when preferred strategies are close to some specific strategy, for example, if the player wants his bid to be close to some value in the Traveler's Dilemma. The name is given by the shape of the probability distribution. An example of the probability distribution can be seen in figure 3.3. The probability is linearly increasing until a strategy $s_{max}$. This strategy is the parameter for this solution concept. It indicates what strategy the player is most likely to select. The way the probabilities are distributed is by assigning 1 to the first strategy and then increase the value by 1 for the next strategy. This is then continued for all the strategies until and including $s_{max}$. From this strategy onward the value is then decreased by 1. When all strategies have been assigned a value, the values are normalized in order to represent probabilities. This is done by dividing each value by the sum of all values.

This solution concept can be used in different situations, Such as when a player tends to play strategies which are closer to some specific strategy. This is applicable in Bertrand Competition and Nash Bargaining, where playing strategies near the middle seems to be the natural approach. Another application is when players should have an increasing chance of playing the strategies. Here $s_{max}$ is set to the last strategy, which produces a linear increasing probability distribution.

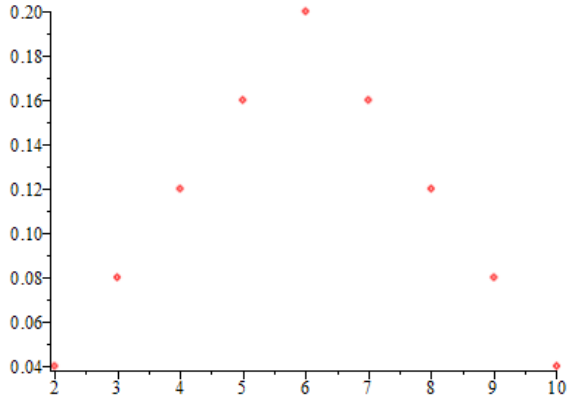


Figure 3.3: An example of the probability distribution for the cymbal solution concept.

**Stop with probability $p$:** This solution concept is created specifically for the Centipede Game. It takes into account the willingness of a player to cooperate. The willingness is represented by the parameter $p$ which is the probability in each round to stop. Since all the games are in strategic form, the probability to stop in each round must be calculated. The probabilities are assigned by the following method. The first strategy is always assigned $1-p$ with the probability to stop. The probability for the next strategy $s_n$ is assigned by multiplying the probability of continuing from all the previous strategies with the probability for stopping at this strategy $p^{n-1}(1-p)$. This is then continued until strategy $s_n$. To this strategy the remaining probability is assigned $p_{s_n} = 1 - \sum_{i=0}^{n-1} p_{s_i}$.

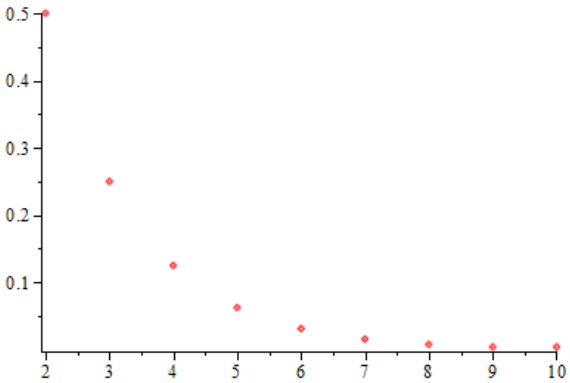


Figure 3.4: An example of the probability distribution for the "Stop with probability $p$" solution concept.

### 3.2.2 Pure Nash Equilibrium

As previously mentioned Nash Equilibria are defined as states where none of the players can get a better outcome by diverting from their strategies. Many algorithms exist which find these equilibria, but the simplest method is to examine each outcome to see if it is a Nash Equilibrium. To do this, the outcome is examined from each of the players' perspective, one at a time. If the player knows that the other player would play the strategy which leads to that outcome, could the player then choose another strategy which has the same or better utility for him? If this is the case the outcome is not a Nash Equilibrium. This is repeated from the other players' perspective. When one outcome has been examined the same examination is repeated for each outcome. The described procedure can be made into the algorithm as seen below in Algorithm 3.1.

**Algorithm 3.1** An algorithm which finds all Nash Equlibria

```
for each outcome (s_I, s_II) do
  for each strategy s_i of player I do
    if u_1(s_i, s_II) ≥ u_1(s_I, s_II) then
      (s_I, s_II) is not an equlibrium
    end if
  end for
  for each strategy s_j of player II do
    if u_2(s_I, s_j) ≥ u_2(s_I, s_II) then
      (s_I, s_II) is not an equlibrium
    end if
  end for
end for
```

This algorithm is not very efficient. It finds all the equilibria by an exhaustive search where it examines all the different possible equilibria. This has an impact on the time complexity. For each of the outcomes, it examines one outcome for each of the strategies of both players. This results in the following asymptotic time complexity $O((n \cdot m)(n+m))$ where $n$ and $m$ and the number of strategies for the players.

### 3.2.3 Regret minimizing solution concepts

The concept of regret minimization is directly tied to the name. First the term regret must be defined. Regret is what you could have gained by acting differently in a given situation. The idea of using this concept for game theory was adopted by P. B. Linhart and R. Radner in 1989 in their article [3]. In the recent years Joseph Y. Halpern and Rafael Pass have adapted this concept by iteration in their article [2]. Regret minimization removes all the strategies which do not have the minimal regret. When iterating regret minimization, the game continually gets smaller for each iteration until no further strategies can be removed.
In the following subsections regret minimization and the iterated version is described. For each of them an algorithm is given, which is then examined.

#### Regret minimization

This solution concept utilizes the regret for removing strategies which have regret above the minimum. The regret is found for each strategy separately and is then compared in order to remove strategies

which do not have the minimal regret. The game used as an example is the same in the article [2]. The game is the Traveler's Dilemma with a maximal bid of 100 and a penalty of 2. The regret of an action is how much more the player could have won, if his response to the specific strategy of the other player instead was the strategy which yielded the highest utility. The best response of a player for this game, is to bid $s_n$ one less than the other player or bid the same if he bids the minimal bid. This gives the best outcome for the player for some bid $s_m$. So for this game the regret is $regret(s_n, s_m) = u(s_m - 1, s_m) - u(s_n, s_m)$ and in the general case $regret(s_n, s_m) = u(s_{n_{best}}, s_m) - u(s_n, s_m)$. From this the regret of a single strategy can be defined. The regret of a strategy is defined as the value of the outcome with the largest regret, which includes the strategy in question. The formal description is $regret(s_n) = max(regret(s_n, s_i)s_i \in S_m)$. From this the regret minimization mechanic can be formalized. It is defined as $rm_p(S_p) = min(regret(s)s \in S_p)$ where $p$ is the player. The strategy which is returned is not unique, since more than one strategy can have the minimal regret. This can now be turned into the algorithm seen in Algorithm 3.2.

**Algorithm 3.2** An algorithm which finds all startegies with minimal regret.

Step 1: Find $u_max$ for each strategy of the other player
Step 2: Find maximal regret for each strategy
Step 3: Find strategies with minimal regret

The algorithm consists of three different steps. To find the time complexity each of the steps must be evaluated. To find $u_max$ all utilities for each of the strategies of the other player are examined in order to find the maximum utility. The entire step then has complexity $mn$, where $m$ and $n$ are the number strategies for player II and player I respectively. The next step has the same complexity, for each of Player I's strategies all utilities are examined by subtracting them from their corresponding maximal utility. The last step is just finding the strategies with the minimal regret which can be done in linear time. This results in the following time complexity: $O(mn + mn + n) = O(mn)$.

#### Iterated regret minimization

The modification Joseph Y. Halpern and Rafael Pass discuss in their article [2], is to iteratively remove strategies which do not have the minimal regret and thereby get a more specific result. This is done by repeatedly calling regret minimization on the game until no more strategies are removed. The complexity of this is related to the number of iterations used. The minimum number of iterations is 1. This is because when no strategies can be removed, then the algorithm simply terminates. If at least one strategy is removed, then the minimum increases to 2. When a strategy has been removed, then the algorithm needs an additional iteration to check whether or not any more strategies can be removed. If no strategies are removed in that iteration the algorithm terminates. This can only be continued for a specific number of iterations and will always terminate. This is ensured by the finite size of the game. In each iteration at least one strategy is removed, so the maximal number of iterations is the same as the number of strategies $n$ of the player.

### 3.2.4 Average regret minimization

A proposed adaptation of the regret minimizing solution concept is to minimize the average regret. The procedure is much like the basic regret minimization, but instead of finding the maximal regret for each of the strategies the average is calculated. The difference from algorithm 3.2 lies in step 2. Instead of finding the single outcome which has the largest regret all the regrets are added together and divided by the number of strategies. This has no impact on the complexity but will have on the

resulting strategies.
As with the normal regret minimization, one can iteratively use average regret minimization to remove strategies. This should also provide more specific results. Since the main algorithm is very identical to regret minimization, and thereby also inherits the complexity and the possible number of iterations.

# Chapter 4

# Analysis

The purpose of this chapter is to examine and specify the scope of this thesis. This is done to clearly state the background for the thesis, the motivation behind it, the methods used and the goal of the thesis.

## 4.1 Background of the thesis

In 1950 J. F. Nash described Nash Equilibrium as a solution concept which could be applied to any game. This was because a result always exists either as a pure or a mixed strategy. However, in order for Nash Equilibrium to be a viable solution concept, all other players are required to be using the Nash Equilibrium as their solution concept. For other games, like the Centipede game or the Traveler's Dilemma, Nash Equilibrium does not provide a meaningful result, which reflects the actions of real life situations. For these problems other solution concepts have been suggested. One of them is regret minimization. This solution produces some very different results from the Nash Equilibrium. A resent modification has been developed by Halpern and Pass in [2]. The solution concept is examined in the article by applying it to different games and examining the behavior. A variation of this solution concept has been suggested by V. Goranko, where the average regret is minimized instead of the maximal regret.

## 4.2 Motivation

The driving force behind this project is, that no examination of the Average Regret Minimization solution concept exists. Additionally has regret minimization not yet been implemented and no empirical results exist. Another reason is to clarify if Average Regret minimization is a better solution than regret minimization based on the maximal regret for each strategy. Performing these analyses would lead to a greater understanding of the usability of this solution concept in real life situations. Depending on the results, this approach could indicate how Iterated regret minimization and Average regret minimization perform compared to Nash Equilibrium.
The results of this analysis would be of interest to people who perform research within this area, but it is of even more importance to people who are using the theory in different applications, since these

results are oriented towards the usage in real situations, thereby having impact on the usability of the different approaches.
My personal interest on the subject of this thesis is based on my background in algorithmic and Artificial Intelligence (AI) courses at the Technical University of Denmark (DTU). Through my work with different areas in informatics and algorithms, game theory has drawn my attention. The ability to predict how a person or entity would react in a situation based on a selection of strategies I find very intriguing and fascinating. Early on in my courses at DTU I become acquainted with the field of game theory, but it was first later on when working with artificial intelligence my real interest for game theory sparked. In combination with my interests in algorithms, it seemed obvious to write a thesis about a combination of these two fields.

## 4.3 Scope

The scope defines the constraints of this project. This will be performed in a hierarchal fashion by gradually becoming more specific. The goal of the scope is to precisely define the subject of this thesis. The general area is game theory, which can be approached from two angles, a game theoretical and an algorithmic approach. Both angles will be taken into account, but the main emphasis will be on the game theoretical analysis.
The main subject of the thesis relates to the analysis and evaluation of how Iterated regret minimization relates to Average regret minimization. The performance and results will be examined on a set of strategic form games which is a subset of those treated in [2]. Since these games are in strategic form they are also considered imperfect information games. The games are also be repeated in order to see how the different solution concepts perform in this situation.
In order to create a foundation for the terms and definitions, a wider survey of the area of game theory is given prior to the analysis of the more advanced solution concepts. When the terms and definitions are settled, the scope can be narrowed to look at the different regret minimization solution concepts.

### 4.3.1 Methodology

In order to accomplish the analyses mentioned in the scope, some fundamental work must be carried out prior to the analyses. First a more wide theoretical analysis of the general area of game theory is carried out in order to create the foundation of all terms and definitions. This analysis contains two parts: a more general description of the general terms and solution concepts, and a separate, and more thorough description of the solution concepts. In the second part, algorithms will be defined for each of the solution concepts in order to have a specific interpretation for each concept. For the different Regret minimization solution concepts, no specific algorithms have yet been defined, meaning that this is also done in this section along with an analysis of the algorithmic properties. These parts are found in chapter 2 and 3.

The theoretical analysis will be used as the foundation for an implementation of a testing tool which is intended as method for the practical testing and analysis. This implementation should encompass algorithms for all the different solution concepts or a possibility to use another implementation of the algorithms.
When the implementation is executed, the attention is turned towards the specific area of this thesis. Where a series of analyses are carried out, the different tests are seen in the list bellow and define the focus of the main area of this thesis.

**Reproduction of test results:** Some of the more interesting results in [2] are reproduced in order to verify them. For some games the parameters will be changed in order further test the behavior beyond what is explained in the article [2].

**Behavior of average regret minimization:** Here the same games are tested but with average regret minimization. The results are then compared with the iterated and the normal version. Furthermore the results are used to compare average regret to maximal regret.

**Iterations of the algorithm:** The number of times the different algorithms need to be iterated is tested and compared.

**Impact of mixed strategies:** Here an experiment is performed in order to determine how many iterations are needed for the average outcome to come close to the expected outcome.

**Solution concepts vs. mixed strategies:** Here the performance of the solution concepts is compared up against some mixed strategies which represent real life situations.

**Solution concepts vs. solution concepts:** Here the different solution concepts play against each other in order to measure their performance.

When all the different experiments have been executed and evaluated an analysis will be written in order to draw a conclusion on the subjects presented in this chapter.

### 4.3.2 Goals

A number of goals can be defined in order to determine what must be accomplished, in order to conclude on the successfulness of this project.

- Establish a wide understanding of the area of game theory.
- Obtain an understanding on how the different solutions concepts work.
- Create an algorithm from the concept of Iterated Regret Minimization and its adaption's.
- Implement a test environment, referred to as the testing tool.
- Reproduce and verify results from [2].
- Test and compare the performance of the solution concepts.
- Evaluate on which solution concept provides the best performance.
- Argue formally for the implications of the results.

The list is not a complete list of everything which is performed within this thesis, only a list of the things which is considered a requirement for this thesis.

# Chapter 5

# Implementation

This chapter contains a description of the implementation of the testing tool. Firstly a description the included functionality and how the implementation has been performed is given. Three elements have been given their own sections, the data structure, creating of test data and the algorithms. However, before the specific elements can be described the requirements for the tool are listed, as shown below.

**Requirements:**

- Use 2-player single move strategic form games.
- Both players must be able to choose from all the different solution concepts.
- Ability to execute the games a fixed number of times.
- Support both pure and mixed strategies.
- Ability to create different games based on their parameters.
- Provide a graphical interface.
- Provide results in an easy and intuitive way.
- Modular implementation which enables an easy way of adding features, more games or solution concepts.

The above list contains the features which must be included in the test tool, in order for it to support the chosen experiments. The procedure of the whole implementation is omitted since it is not considered an important part of this thesis. However, important features are emphasized in the following sections by detailing their specific implementation.

## 5.1 Implementation environment

The implementation environment consists of a short description of the hardware and software used during the implementation and running of the testing tool. The hardware consists of a laptop with

the following specifications: Intel Core2 Duo T5800 2.0 GHz, 4 GB ram, Windows 7 Professional 64-bit. The implementation was carried out in JavaSE-1.6 using Eclipse SDK Version: 3.5.2 as the IDE (integrated development environment). For backup and version control an SVN-Server located on DTU's GBar was used. To utilize the functionality of SVN TortoiseSVN version 1.6.7 was used.

## 5.2 The testing tool

The tool was implemented in order to have a common environment for all the experiments. The testing tool can be divided into two parts, the user interface and the backend mechanics. The user interface takes care of all inputs and outputs along with error handling. The backend mechanics takes care of executing the right algorithms, finding the results of the solution concepts and simulate the execution of the games.
A class diagram of the entire implementation is seen in fig. 5.1. All the associations are single associations and one-way.

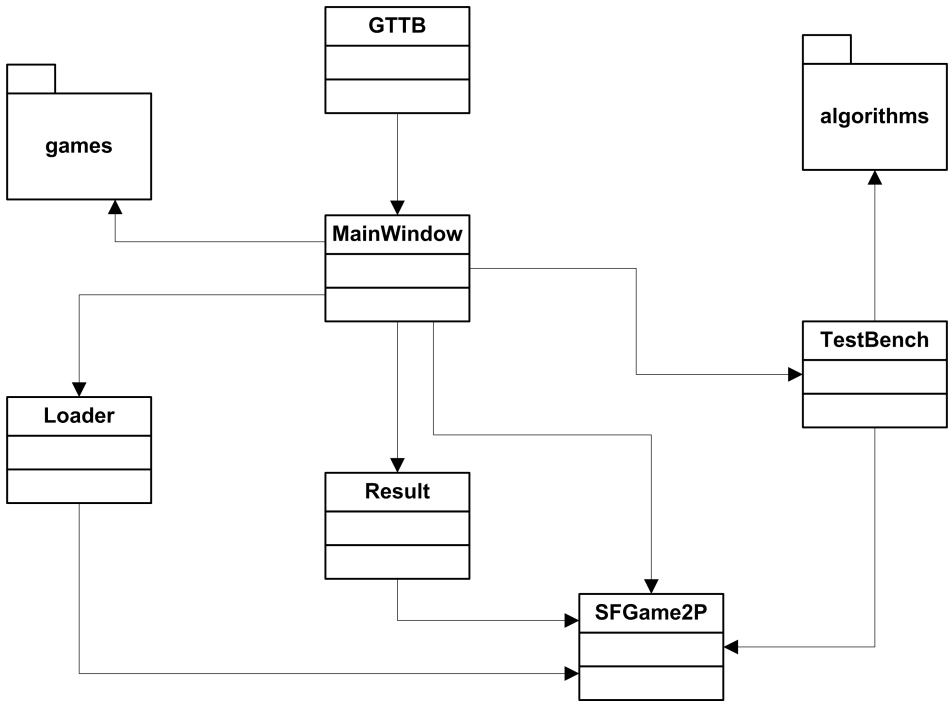


Figure 5.1: A class diagram of the testing tool.

From the class diagram the structure of the implementation can be seen, which serves at the basis for the description of the implementation of the testing tool. The application's name is *GTTB* which is an abbreviation of *Game Theory Test Bench*. Each of the classes are described below with focus on what they are responsible for.

**GTTB:** This class is the main class which opens a single window in which all the testing occurs.

**MainWindow:** Is the class which contains all UI elements and is responsible for all input/output.

**Loader:** Loads a game from a file specified by MainWindow.

**Result:** Data structure for a result of an execution of some game.

**TestBench:** The class responsible for obtaining strategies from the selected solution concepts and executing the game.

**SFGame2P:** The data structure containing a game.

**Games:** This is a package where all the different classes for constructing games are located.

**Algorithms:** This package includes all the different algorithms for the different solution concepts.

The most interesting areas consist of the user interface, data structures, the way games get created (creating test data) and the algorithms which use the different solution concepts. Each of these components are described individually below.

### 5.2.1 The User Interface and its functions

The user interface (UI) was implemented using the Java SWING library. The reason for this choice is that it is easy to use and contains all the required functionality for the implementation. In figure 5.2 a screenshot of the UI is shown.

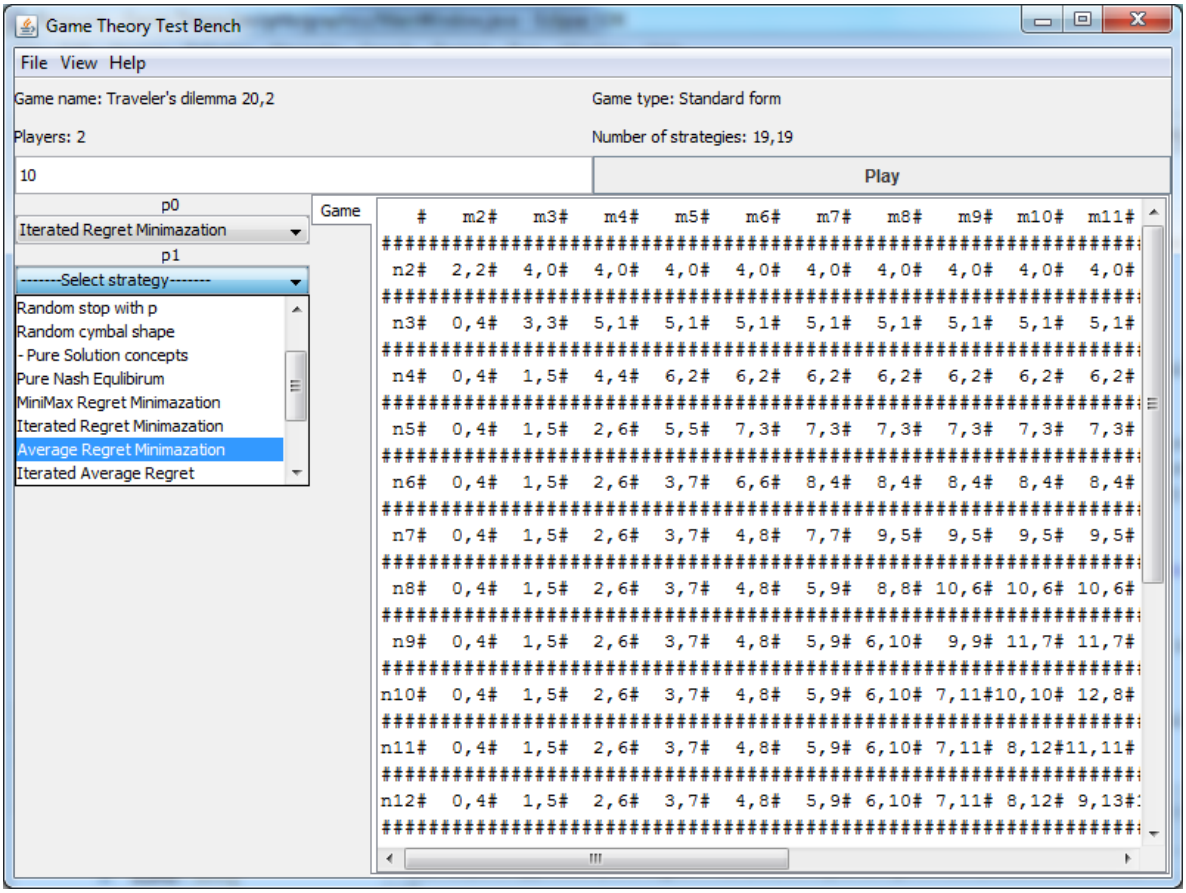


Figure 5.2: The user interface for the testing tool.

The toolbar contains three elements. Firstly the File segment, which contains: Open Game, Load Game Type, Save Game, Export Gambit File and Exit located.
Open Game loads a game from a file, the file format can be found in appendix C. The different lines of the file are includeding the following information.

- The first line is the number of players.
- The next is $U$ if the values of the outcomes are utilities and $C$ if they are costs.
- The next line is how many strategies the different players have.
- The Two next lines are the names of the strategies of the players.
- Finally comes the outcomes of the games. They are stated as pairs of utilities for each of the players.

Load Game Type includes a set of submenu's located for each of the game types described in chapter 3. Clicking on one of the menu items causes a popup to appear, where the parameters of the game can be entered.
Save Results simply saves the results of a game as a .txt file. Export Gambit File saves the current games in a format which can be read by the Gambit [1]. The final menu item is simply to close the application.
The next item in the toolbar is View: This contains options about what information is shown about the game and the result. The first submenu item is an option for how the results of each iteration should be shown, the next is and enable/disable of the game matrix. The final submenu is whether or not the probabilities of the different strategies should be shown. The reason for these features being able to be enabled/disabled is because they can affect the runtime, make the testing tool more/less confusing to use, both of which often happens with using large games.
The final item on the toolbar is Help: The Help menu simply contains a menu item that creates a popup with information about the author and the testing tool.

In the topmost section of the window right below the toolbar information on the name of the game and the number of strategies of the players is displayed. Below this a text field and a button is found. The text field is used to enter the number of iterations of a given game. The button is to play the current game and execute all the iterations; the button appears when a game has been loaded. In order to be able to play the game a solution concept must be chosen for each player. The strategies are chosen from a dropdown list located under the text field. In addition to the strategies a valid number of iterations needs to be selected. Once the play button is pressed an additional tab appears to the right of the dropdown menus, this tab contains the results of the game.

### 5.2.2 Data structures

Throughout the implementation of the testing tool different data structure are used, however, it is the one used for storing games that is the most interesting. As mentioned earlier, a game consists of three components, the strategies, the outcomes and the players. The outcomes are defined as the result of a function which maps a strategy of each player to a set of utilities. The data structure which is used for encompassing all the information is arrays. For each of the players two arrays are given, a two dimensional array to hold the outcomes of that specific player, and an array of strings with the names of the strategies of the player. The entire data structure is then composed of four arrays, where two of

[1] Another program for computing nash equilibriam. Can be found at http://www.gambit-project.org

them are two dimensional. The space complexity is then $nm$ which is the size of the two dimensional arrays.
During an execution of a game are the results of each iteration is recorded. The data which is recorded consists of the strategies of each of the players, the specific outcome for each iteration, the utility gained each iteration of each player and the collected utility from all the iterations.

### 5.2.3 Creating the test data

The different game types each have a class to produce a game representation based on the parameters. In the experiments four different types of games are used, but the centipede have two variations and is therefore implemented with an extra parameter to specify which type of games is wanted. For each of the games the method for creating and name the actions along with the utilities for each of the players is given. The issues with the creation of the different games are based on how the arrays are indexed. In Java indexing starts at index 0. The problem is then how to map from the index to the value of the action. Below the implementation of each game is described.

**Traveler's Dilemma:** The two parameters for this game are the maximal bid and the penalty. The only parameter which has influence on the actions is the maximal bid. An action is to bid a specific value. The number and actions are all the integers in the range $[2, ..., bid_{max}]$. This results in $bid_{max} - 1$ different action named the same as the bid they represents. The action is identical for both the players.
To accurately map from the indices to the bids is simply by adding 2 to the index. For both the utility arrays the utilities can then be calculated by using the function in 3.1.

**Centipede Game:** For the centipede game three parameters are needed. The number of rounds $rounds$, the penalty and the type of centipede game. The class which controls this game first creates the strategies since they are the same for both types of games. A strategy in this representation of a centipede game, is to stop at a certain round. The strategies for the players are not the same for this game type. The strategies for of player I depends on the number of rounds. If the rounds are even the strategies range consists of $[1, 3, ..., rounds - 1]$, and $[1, 3, ..., rounds]$ if the number is odd. For the other player it is the other way around, $[2, 4, ..., rounds]$ for even, and $[2, 4, ..., rounds] - 1$ for odd. To map from the indices to the rounds is therefore different for the two players. For player I it is done by $\frac{rounds}{2} + rounds\%2$ and $\frac{rounds}{2}$ for player II. The strategies are then named by $i \cdot 2 + 1$ and $i \cdot 2 + 2$ for player I and II respectively.
The utilities are then found from the actions. For each outcome, the player who has the lowest strategy is the one to say stop first. If the indices are the same, then it is player I who has stopped first. The specific utility the players depends on the type of game. For exponential games it is simply to give the player who stopped first $2^{round} + 1$ and $2^{round-p}$ to the other player.

**Bertrand Competition:** The action in this game is to bid a value for the wares. The player can bid any amount between $[0, 1, ..., bid_{max}]$. Each player has $bid_{max} + 1$ strategies. The bid of the actions are simply the same as the indices in the arrays. The utilities are assigned by looking at the relation between the bids of the players. The player who bids lowest gets $bid \cdot wares$ and the other 0. If they bid the same they are both assigned $bid \cdot \frac{wares}{2}$.

**Nash Bargaining Game:** The strategies for this game are identical to the Bertrand Competition. They have the same number of strategies and they are named in the same way. The action the strategies represent are again the bids in the game. The utilities are either the same amount as the bid or 0 depending on the sum of the bids.

The class MainWindow makes sure that only valid games are created in accordance to the requirements described in chapter 3.

### 5.2.4 The algorithms

From each of the different solution concepts an algorithm has been created. Each of the algorithms are described in the list below. The description includes how the different algorithms are converted to an implementation and what is added or removed from the algorithms in order to implement them. Each of the algorithms receives a game and the player for which it has to find a strategy. Two of the classes also receive another input. The Cymbal class also gets the strategy which should have the highest probability and the Stop class with probability $p$ gets the probability $p$.

**Mixed strategies:** The implementation of these was straight forward. For the uniform version is it based on the number of strategies. For the two others their extra parameter are what controls the way the probabilities are assigned.

**Nash Equilibrium:** This is implemented by looking at each outcome in turn. First a list of all the different strategies is recorded. For each outcome which is not an Nash Equilibrium the corresponding strategies will be removed from the list. Lastly the result returned by either returning null if no Nash Equilibria is located or an array of all the strategies where the probability is uniformly distributed over all the strategies which is part of an equilibrium.

**Regret Minimization:** The implementation follows the algorithm very closely. When the minimal regret of the strategies is found, an array with the same size as the number of strategies is created. In this the probability is evenly distributed between the strategies with the minimal regret.

**Iterated Regret Minimization:** To iterate this algorithm the non-iterated version is used in a slightly modified version. The modification consists of the following: instead of returning a result, the result is simply recorded in an array and a variable is set to indicate that at least one strategy has been removed. The procedure is then repeated but the strategies removed in the previous iterations are no longer used for the calculations. This is repeated until no more strategies are removed.

**Average Regret Minimization:** The implementation of this solution concept is almost identical to Regret Minimization. The only difference is that the average regret is computed instead of the maximal regret for each strategy.

**Iterated Average Regret Minimization:** This is also uses a modified version of the non-iterated version of the same solution concept.

## 5.3 Testing

Testing and validating the implementation was divided into four areas: The UI, the games creating, the algorithms and the game execution.
The UI contains different features such as open games from files, saving them and converting to other file types. These features were tested by a functional test. This involved running the different feature and assess if it is the right behavior was observed. If some features did not react as expected, it was fixed on the run by adding or changing a feature.
The feature where games were created from some parameters were tested by creating different games

and observe if it was created according to the descriptions in this chapter and in chapter 3.
The algorithms were tested one at a time. The ones based on the solution concepts from the article [2] were tested with the games from the same article. The results can be seen in table B.1 in the appendices. The adapted solution concepts were also tested on these games and the behavior was observed. A more detailed evaluation of the results is given in the next chapter.
The mechanism for running games was tested by running some different games where the results were known prior to the test run. The utilities were then observed and compared manually to the outcomes. Each of the iterations were compared the outcomes of the games and the collected outcome were calculated.

# Chapter 6

# Experiments and evaluation

The experiments found within this thesis can be divided into two different sections and each section can be divided further into subsections. The first main section concerns regret minimization. In this, the solution concept proposed by Halpern and Pass in [2] is tested with in the testing tool. The second main section concerns average regret minimization, here the two solution concepts are compared to each other and to the regret minimization described by halpern. After each of the experiments, an evaluation is given on the results of the given experiments indicated above. The results of the experiments can be seen in appendix B.

## 6.1 Experiments with regret minimization

The main objective of this experiment is to confirm some of the more interesting results from [2]. These are the results from the games defined in *The Games* in chapter 3, which are reproduced. The first objective is to extract and verify the results from Halpern and Pass' article. The results are seen in the appendices in table B.1. How the games are tested, in order to produce the results, are then described; followed by the produced results. Finally, the results are evaluated. Each of the games are handled individually and in turn. The data for each of the experiments can be seen in the appendices B.1

**The Traveler's Dilemma:**
The results from the article of the Traveler's Dilemma are defined for a game with a specific size, but with varying penalties. For a game of size $k = 100$, the magnitude of the penalty can be $p = [2, ..., 100]$ and the article examines the results for $p < 50$ and for $p \geq 50$. For $p \geq 50$ both the regret minimizing solution concepts follow the same strategy as that of the Nash Equilibrium. For $p < 50$, the suggested strategy is closely tied to the specific size of the penalty. The non-iterated regret minimization suggests that all strategies $[100 - 2p, ..., 100]$ minimizes the regret. The iterated version strictly suggests that $100 - 2p + 1$ minimizes the regret. The article [2] has n error where it uses the Traveler's Dilemma as an example. At page 10 it states that Iterated Regret Minimization has $100 - 2p + 1$ as the strategy which minimizes the regret for $p \geq 50$. This should obviously be for $p < 50$ instead as mentioned earlier in the article.

The experiment is carried out by running a series of games and recording the results. To reproduce the results, the penalty parameter is changed to check the results for the different solution concepts. For most of the results in table B.2 found in the appendices, the results from the article holds. It is easy to check whether or not the limit $100 - 2p + 1$ holds for the games, which is does for results with $p < 49$, but when the penalty gets near the limit 50 something happens which the article does not take into account. For $p \geq 49$ the results differs from the article. At $p = 49$ the result of the single iteration regret minimization is the single strategy 3. The reason for this is because of the minimal bid of the players. When the penalty is increased, so are the number of strategies which survive the first iteration. When $p = 49$ strategies should survive, the first iteration is $[100 - 2(49), ..., 100]$, which is all the strategies. Instead of removing the strategies before $100 - 2p$, which in this game does not exist, it does what the second iteration does for games with lower penalty and picks a single strategy instead. The results from the article mentions, that the chosen strategy for games with $p \geq 50$ should be the strategy with the minimal bid. This is not the case; instead it is the second smallest bid. If the minimal bid was 0, then the suggested strategies would be $[2, ..., 100]$.

Table 6.1: The Traveler's Dilemma with a max bid 10 and a penalty of 5.

| TD(10,5) | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 2,2 | 7,-3 | 7,-3 | 7,-3 | 7,-3 | 7,-3 | 7,-3 | 7,-3 | 7,-3 |
| 3 | -3,7 | 3,3 | 8,-2 | 8,-2 | 8,-2 | 8,-2 | 8,-2 | 8,-2 | 8,-2 |
| 4 | -3,7 | -2,8 | 4,4 | 9,-1 | 9,-1 | 9,-1 | 9,-1 | 9,-1 | 9,-1 |
| 5 | -3,7 | -2,8 | -1,9 | 5,5 | 10,0 | 10,0 | 10,0 | 10,0 | 10,0 |
| 6 | -3,7 | -2,8 | -1,9 | 0,10 | 6,6 | 11,1 | 11,1 | 11,1 | 11,1 |
| 7 | -3,7 | -2,8 | -1,9 | 0,10 | 1,11 | 7,7 | 12,2 | 12,2 | 12,2 |
| 8 | -3,7 | -2,8 | -1,9 | 0,10 | 1,11 | 2,12 | 8,8 | 13,3 | 13,3 |
| 9 | -3,7 | -2,8 | -1,9 | 0,10 | 1,11 | 2,12 | 3,13 | 9,9 | 14,4 |
| 10 | -3,7 | -2,8 | -1,9 | 0,10 | 1,11 | 2,12 | 3,13 | 4,14 | 10,10 |

In table 6.1 the Traveler's dilemma with a maximal bid of 10 and a punishment of 5 is shown. From this game it can be seen, why the strategy with the second lowest bid, which has the minimal regret. For the strategies 2 and 3 the action of the opponent which maximizes the regret is if he bids the maximum amount. For all the other strategies, it is the action of bidding just one less than player I which maximizes the regret. Strategy 2 will always have a higher regret than strategy 3 when the opponent chooses the maximal bid. This leads to strategy 3 always being chosen over the first. This behavior continues until the penalty approaches the size of the game. At $p = bid_{max} - 3$ the behavior changes once again, here 2 and 3 initially have the same regret and in the next iteration 2 has the lowest regret and is therefore the only strategy which survives both iterations. With any higher penalty, the only strategy which survives a single iteration is 2.
The behavior of the solution concepts can be described more formally by making an expression for the regret for each set of strategies. For this there exists three situations, both players can bid the same, the first player can bid lower than the other player, or the first player can bid higher than the other. From chapter 3 section "The Games" we have the regret for each of these situations. It can be written as a simple function as seen in equation 6.1.

$$regret_1(n, m) = \begin{cases} (m - 1 + p) - (n + p) & \text{for } n < m \\ (m - 1 + p) - n & \text{for } n = m \\ m - 1 + p - (m - p) & \text{for } n > m \end{cases} \tag{6.1}$$

From equation 6.1 a new expression can be created, which tells the maximal regret for each of the possible strategies for player I. To do this, the situations which maximize the regret must be uncov-

ered. It is easily seen, that the situation which maximizes the regret, is $m = bid_{max}$ for $n < m$. The other situations do not depend on the answers. From this the expression in equation 6.2 can be created.

$$regret_1(n) = max(bid_{max} - n - 1, p - 1, 2p - 1) \tag{6.2}$$

Since $p-1 < 2p-1$ for $p >= 0$ the second expression is never the maximum and is therefore not taken into account. The first part of the expression is descending and the third part is a constant. The limit for where the strategies survive is described by when the two expressions are of equal size. This results in a limit of $limit_{minreget} = bid_{max} - n - 1 = 2p - 1 \rightarrow n = bid_{max} - 2p$. This then proves why the single iteration returns the proposed result.

Table 6.2: The Traveler's Dilemma with a max bid 10 and a penalty of 3.

| TD(10,3) | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 2,2 | 5,-1 | 5,-1 | 5,-1 | 5,-1 | 5,-1 | 5,-1 | 5,-1 | 5,-1 |
| 3 | -1,5 | 3,3 | 6,0 | 6,0 | 6,0 | 6,0 | 6,0 | 6,0 | 6,0 |
| 4 | -1,5 | 0,6 | 4,4 | 7,1 | 7,1 | 7,1 | 7,1 | 7,1 | 7,1 |
| 5 | -1,5 | 0,6 | 1,7 | 5,5 | 8,2 | 8,2 | 8,2 | 8,2 | 8,2 |
| 6 | -1,5 | 0,6 | 1,7 | 2,8 | 6,6 | 9,3 | 9,3 | 9,3 | 9,3 |
| 7 | -1,5 | 0,6 | 1,7 | 2,8 | 3,9 | 7,7 | 10,4 | 10,4 | 10,4 |
| 8 | -1,5 | 0,6 | 1,7 | 2,8 | 3,9 | 4,10 | 8,8 | 11,5 | 11,5 |
| 9 | -1,5 | 0,6 | 1,7 | 2,8 | 3,9 | 4,10 | 5,11 | 9,9 | 12,6 |
| 10 | -1,5 | 0,6 | 1,7 | 2,8 | 3,9 | 4,10 | 5,11 | 6,12 | 10,10 |

The next iteration has some different premises. When looking at the game in table 6.2, it can be seen that the maximal regret changes when removing the colored strategies. The maximal regret for the new two smallest strategies is now when the opponent bids the maximal bid. A new expression for maximal regret can then be defined as seen in equation 6.3. Minimizing this expression leads to a minimum of exactly $max - 2p + 1$. This then proves why this is the result of iterated regret minimization.

$$regret_1(n, m) = \begin{cases} bid_{max} - 1 - n & \text{for } n \leq max - 2p + 1 \\ 2p - 1 & \text{for } n > max - 2p + 1 \end{cases} \tag{6.3}$$

When the penalty is high, then the results change. The easiest way to explain this behavior is by looking at what happens in the second iteration for games with a low penalty. When $p \geq \frac{bid_{max}}{2} - 1$ then the structure of the games change so that it looks like a game where only strategies which has survived the first iteration is present. It has already been proved what the result of regret minimization is on a game of this type, so it is apparent why the result is $min_{bid} + 1$.
When the penalty approaches the maximal bid, the results change again. The result of the regret minimization becomes the lowest strategy possible. This is because, for all the other strategies, the action of the opponent which maximizes the regret for player one is to bid 3. For bid 2 the action is instead to bid the maximal value. This pattern continues for all penalties larger than the limit $p \geq \frac{bid_{max}}{2} - 1$.

Changing maximal bid affects the outcome, however, this is tied to the penalty. When $p < bid_{max}$ then increasing the $bid_{max}$ does not change the behavior of the algorithm. The results have the same number of strategies; they just get pushed forward by an equal amount as the change in the maximal bid. When decreasing the maximal bid the result gets moved backward in a similar manner. It changes

when $bid_{max} < \frac{p}{2} - 1$ which is the same ratio when changing the penalty.

**Summary:** Two differences from the article [2] were found. The first was the limit of what the penalty is when the outcome pattern changes. Instead of $p \geq \frac{bid_{max}}{2}$ is it for $p \geq \frac{bid_{max}}{2} - 1$. The other difference is the suggested strategy for games with a high penalty. Instead of assuming the minimal mid is the second lowest bid. Proofs were given for each of the differences. Furthermore the differences with other parameters for the game were tested. The games were with a high penalty $p \geq bid_{max} - 3$. At this limit, the result of the iterated version changed to the strategy for the minimal bid. The non-iterated version returned both the strategies for the minimal bid and the second to minimal bid.

**The Centipede Game:**
The next results examined are that of the two versions of the centipede game. The two types of the game have different results from the regret minimizing solution concepts, while the Nash Equilibrium remains unchanged, namely stop at the first possible round. The simplest results are for the exponential version. Here, that action that minimizes the regret is the same for both solution concepts, namely to stop at the last possible round. The linear version of the centipede game is more interesting. The results of the iterated version vary depending on whether or not the number of rounds and the penalty is even or odd. The result of the non-iterated version is the same no matter what the values are of the parameters. The result is $[k - p + 1, k - p + 3, ..., k - 1]$. When both parameters are odd, then iterating make no difference and therefore produce the same results. On the other hand, when both the parameters are even, then iterating produces a single unique result $k-p+1$ which minimizes the regret.

*The exponential centipede game* is by far the less interesting. The Nash Equilibrium states that the only possible strategy is to stop at the first possible round. The results also support this empirically. The reason why this happens can be seen by looking at the dominated strategies. Starting at the last strategy for player I, this is dominated by the second last strategy which is in turn dominated by the third last. This continues until only the first strategy remains. A dominated strategy can never be a part of a Nash Equilibrium, and therefore is the only Nash Equilibrium to stop the game at the first possible round.
The results for Regret Minimization do in most cases follow those of the article. Some variations can be seen both with the number of possible strategies for the single iterations version, but also with the final result of the iterated version in some cases. First the result of the regret minimization where it finds more than one strategy is examined. It only occurs when $p = 1$ and the number of rounds is odd. When this is the situation the solution concept finds two strategies which have the same regret. The strategies represent the action of stopping in the two last possible rounds. By looking at the game in table 6.3 and observing the structure of the two last strategies for player I, it can easily be seen why this behavior occurs. The maximal regret for both strategies is 59, and therefore, the only difference between the two strategies does not affect the outcome until the next iteration. In the next iteration is it easy to see, that it is the second last strategy which has the minimum regret because weakly dominates the other strategy. The same structure is evident with size game with an odd number of rounds. If the number of rounds are even, then the strategy of player I is removed, and thereby one strategy has the minimal regret, and it is therefore simple why this behavior does not appear for games with an even number of rounds. From the game it can also be seen why this behavior only appears with games with $p = 1$. With $p > 1$ the second last strategy does no longer have as low a regret as the last and thereby the situation with more strategies with the minimal regret no longer exists. When changing the punishment to higher values, nothing changes in relation to the results of the solution concepts. Even with very high numbers, where the punishment is a 0 utility, the solution concepts are unaffected.

Table 6.3: The Exponential Centipede Game with $k = 9$ rounds and a penalty of 1.

| EC(9,1) | 2 | 4 | 6 | 8 |
|---|---|---|---|---|
| 1 | 5,2 | 5,2 | 5,2 | 5,2 |
| 3 | 4,9 | 17,8 | 17,8 | 17,8 |
| 5 | 4,9 | 16,33 | 65,32 | 65,32 |
| 7 | 4,9 | 16,33 | 64,129 | 257,128 |
| 9 | 4,9 | 16,33 | 64,129 | 256,513 |

The game seen in table 6.3 can also be used to explain the behavior where the result of regret minimization is the strategy with the second highest bid of the game. It can be seen in table B.3 within the appendices, that this behavior only occurs in games with an odd number of rounds. The behavior can be traced back to why the previous behavior only occurred with games with an odd number of rounds. In these situations, the choosing the last strategy means cooperation until the end of the game, if the number of rounds are odd, then the other player's last possible moment to not cooperate is in the second last round (round 8 in the example). This results in a situation where player II always is the one to defect first when player I is playing the last strategy and therefore he can always get a better outcome by choosing the previous strategy, this results in the last strategy being dominated by the previous strategy and is therefore never chosen. This is in fact the same result as the article shows, the difference lies in the implementation of the game in the implementation.
From the above analysis it is observed why the last or second to last strategy is always the solution for regret minimization. Defecting later always has a possibility of getting a higher outcome. For player I, defecting at round $r = 1$ will always have a possibility to get a better outcome, the result can also be worse, but this is not taken into account in regret minimization. The later one stops, the higher are the possible outcomes, and therefore the lower the regret. The only deviation is with the last strategy for a game with an odd number of rounds as explained earlier. The regret of a given strategy can rather easily be found by defining a function. The single event which induces the maximal regret for each strategy is when the opponent chooses the latest round possible to stop. In this situation, the maximal regret is given in equation 6.4.

$$maxregret_1(n) = \begin{cases} (2^{k-2}+1)-(2^n+1) & \text{for } k \text{ odd and } n \neq k \\ (2^{k-2}+1)-(2^{n-2}) & \text{for } k \text{ odd and } n = k \\ (2^{k-1}+1)-(2^n+1) & \text{for } k \text{ even} \end{cases} \tag{6.4}$$

From equation 6.4 it can easily be seen, that in order to minimize the regret, the strategies with $n = k - 2$ are preferred for games with an odd number of rounds, and the strategies with $n = k - 1$ for games with an even number of rounds.

**Summary:** In the case of $p = 1$, the single iteration version of the regret minimization had a deviation from the results in the article. It found an additional strategy in its results. It was only for the non-iterated version and was only a single more strategy, so the loss of precision was kept small. The biggest impact is in the iterations which will be discussed in a later experiment.

*The linear centipede game* has some other results than the exponential version. The Nash Equilibrium remains unchanged. For every game it is still the first possible round one should defect while adopting this solution concept. For most other games, both the regret minimization solution concepts tend to follow the results of the article. There are a few games where the results in table B.4 deviate from

the results in the article. The first are where games with $p = 2$ have a different number of results depending on the number of rounds being even or odd. This relates directly to some of the results for the exponential version of the game. The same argument applies to this deviation as for why regret minimization finds two different strategies with minimal regret in exponential centipede games with an odd number of rounds. The article does not examine all the different types of centipede games. The article describes what happens when either both the punishment and the number of rounds are even or if they both are odd. From the results in table B.4 it can be seen what the behavior is when both the parameters are not either odd or even. When the number of rounds is odd and the punishment even, then the results are the same as for games where both parameters are even. When the number of rounds is even and the punishment odd, then the results follow those of games where both parameters are odd. From this a more precisely definition, of when the iterated solution concept does not yield a more specific result, can be made. The behavior is not tied to the punishment, but only the number of rounds. This behavior can be explained by looking at the game in table 6.4. From this table the pattern of the game is apparent. For this game, the strategies $[7, ..., 13]$ are the result of both the iterated and the non-iterated version of the regret minimization solution concept. The maximal regret for all of these strategies is 4. The regret can be found without using the strategies $[1, ..., 5]$ and thereby removing these strategies does not yield a different result.

Table 6.4: The Linear Centipede Game with $k = 13$ rounds and a penalty of 5.

| LCG(13,5) | 2 | 4 | 6 | 8 | 10 | 12 |
|---|---|---|---|---|---|---|
| 1 | 7,2 | 7,2 | 7,2 | 7,2 | 7,2 | 7,2 |
| 3 | 3,8 | 9,4 | 9,4 | 9,4 | 9,4 | 9,4 |
| 5 | 3,8 | 5,10 | 11,6 | 11,6 | 11,6 | 11,6 |
| 7 | 3,8 | 5,10 | 7,12 | 13,8 | 13,8 | 13,8 |
| 9 | 3,8 | 5,10 | 7,12 | 9,14 | 15,10 | 15,10 |
| 11 | 3,8 | 5,10 | 7,12 | 9,14 | 11,16 | 17,12 |
| 13 | 3,8 | 5,10 | 7,12 | 9,14 | 11,16 | 13,18 |

The regret can also be described in a more formal matter. To do this, first an expression for the regret in any given outcome must be created. There exists two different outcomes, either player I stops earlier or after player II. When the opponent stops later, then the regret is what he could have won if he stopped just before the opponent which yields a regret of $(m-1)-n$, If on the other hand, the opponent stops earlier than player I, then his regret is $(m-1)-(m-p)$. This can be contracted into the regret formula seen in equation 6.5.

$$regret_1(n,m) = \begin{cases} (m-1)-n & \text{for } m \geq n \\ (m-1)-(m-p) = p-1 & \text{for } m < n \end{cases} \tag{6.5}$$

For each strategy, there exists two schemes which the opposing player can play in order to maximize the regret of the first player. These schemes can be either to play the last possible strategy or to play the action precisely one lower than the first player. The last possible round the opposing player can stop depends on whether or not the number of rounds is even, it can be either $k$ for even numbered games and $k-1$ for odd. With the two schemes in mind a formula for the maximal regret for each strategy can be devised. This is done by looking at which strategy $m$ would maximize the regret for each strategy $n$. For the first situation, the larger $m$ is, the larger the regret. The next expression does not depend on the strategies but remains the same. This can then be contracted into the expression seen in equation 6.6 which defines the maximal regret for each possible $n$.

$$maxregret_1(n) = \begin{cases} max((k-1)-n-1, p-1) & \text{for odd k} \\ max(k-n-1, p-1) & \text{for even k} \end{cases} \tag{6.6}$$

From this it can then be analyzed, that for low $n$ the regret depends on the number of rounds, but at some limit, the regret changes to the constant $p-1$. It can then be seen, that all the strategies at this limit and from there on have the same minimal regret. The limit is where the max changes to $p-1$. This is $k-p+1$ and $k-p$ depends on the number of rounds.
From the expressions $k-p+1$ and $k-p$ can it then be analyzed how regret minimization reacts when changing the two parameters since all strategies $n >= k-p$ or $n >= k-p+1$ have minimal regret. From this it can then be seen, that by changing the number of rounds, the number of strategies is unaffected, but when changing the punishment the number of strategies that survive the first iteration of regret minimization is increased for every 2 added to the punishment. When a new iteration is run, then the results vary depending on the punishment. The formal explanation for this, is that, when the punishment is odd, the strategies removed in the first iteration do not have any influence on the maximal regret for each of the possible strategies, and therefore do not change the outcome by running more iterations. When the punishment is even, then iterating changes the result. Looking at the game in table 6.5, the behavior can be explained.

Table 6.5: The Linear Centipede Game with $k = 13$ rounds and a penalty of 6.

| LCG(13,6) | 2 | 4 | 6 | 8 | 10 | 12 |
|---|---|---|---|---|---|---|
| 1 | 8,2 | 8,2 | 8,2 | 8,2 | 8,2 | 8,2 |
| 3 | 3,9 | 10,4 | 10,4 | 10,4 | 10,4 | 10,4 |
| 5 | 3,9 | 5,11 | 12,6 | 12,6 | 12,6 | 12,6 |
| 7 | 3,9 | 5,11 | 7,13 | 14,8 | 14,8 | 14,8 |
| 9 | 3,9 | 5,11 | 7,13 | 9,15 | 16,10 | 16,10 |
| 11 | 3,9 | 5,11 | 7,13 | 9,15 | 11,17 | 18,12 |
| 13 | 3,9 | 5,11 | 7,13 | 9,15 | 11,17 | 13,19 |

The game in table 6.5 has the same result for at single iteration of regret minimization as the game in 6.4, but iterating further does produce a different result in the game with an even penalty. This can be seen by looking at the maximal regret for each of the strategies for player I. Initially the maximal regret for strategies $[7, .., 13]$ are all 5, but when the strategies $[1, .., 5]$ are removed, the regret of strategy 7 is now 4 while the regret for the other strategies remains unchanged. This is because the regret of strategy 7 depends on strategy 1 in to first iteration. This is the same for other games with an even penalty. The regret of the lowest strategy which survives the first iteration always depends on strategy 1.
The article does not explain what happens when $p \geq k$. Looking at what happens when $p$ approaches $n$ this is easy explained, the number of strategies which survive the first iteration become greater and greater until all of the strategies should survive. The game where all the games should have survived, as a drastically changed outcome. When this happens, the result of regret minimization is the same as the Nash Equilibrium, namely to stop at the first possible round. This can be attributed to the structure of the game. When $p \geq k$, then when the opponent stops at the first possible round, the action maximizes the regret. This is because the punishment is at least as much as the amount a player can win by continuing to the end. This is also supported by the empirical evidence from the results in table B.3 in the appendices.

**Summary:** In the linear centipede game, the only deviation from the article was that the result of the solution concepts was not tied to both the parameters, but only whether the penalty was even or odd. When the penalty was odd, iterating would not produce a more precise result, so for games with a high penalty were the results are a lot less precise. It will be seen in a later experiment what impact this has on the expected utilities.

**The Bertrand Competition:**

The result of the Bertrand competition is for a specific game where the maximal bid is 200$ and the number of wares for sale are a 100 pieces. There are two Nash Equilibrium in this game 0 and 1. Regret minimization suggests a very different approach. A single iteration exposes 100 and 101 as the two strategies which minimize the regret and another iteration singles out 100 as the best strategy.
To reproduce the results and verify empirically, a single game is being played and the result is recorded. Since the article only refers to the game with a specific set of parameters it is a very quick to test. First the strategies suggested by the Nash Equilibria are examined. From the test results it is seen, that not only 0 and 1 are included in the Nash Equilibria, but also 2. This can rather simply be tested by looking at the game in strategic form. Since the game table is rather big (201x201) it is not possible to show it in its complete form, but in table 6.6 a small section is seen. From this is it easy to see that $(2, 2)$ is a weak Nash Equilibrium. Turning the attention towards the other solution concept it is seen that the experimental results follow the results from the article perfectly.

Table 6.6: Small section from the Bertrand Competition Game from the article.

| BC(200,100) | 0 | 1 | 2 | 3 | ... | 201 |
|---|---|---|---|---|---|---|
| 0 | 0,0 | 0,0 | 0,0 | 0,0 | ... | 0,0 |
| 1 | 0,0 | 50,50 | 100,0 | 100,0 | ... | 100.0 |
| 2 | 0,0 | 0,100 | 100,100 | 200,0 | ... | 200,0 |
| 3 | 0,0 | 0,100 | 0,200 | 150,150 | | |
| ⋮ | ⋮ | ⋮ | ⋮ | | | |
| 201 | 0,0 | 0,100 | 0,200 | | | |

The next results in appendix B.5 are used for examining the behavior of the solution concepts when changing the value of the parameters. When changing the number of items for sale, nothing is changed with the results of the algorithms. This is not that surprising seeing that the only impact the number of items is a factor which the outcomes are multiplied. For all $no_{items} > 0$ the results remain unchanged. 0 is not examined since it will result in a games with 0 utility for all outcomes.
Changing the maximal bid only changes the result of the single iteration version of regret minimization. When the maximal bid is an odd number, the result is found in the first iteration. The reason can be explained by looking at the maximal regret for a game with parameters $maxbid > 2$ and $no_{items} > 0$. $n$ is the bid of the first player and $m$ is the bid of the other. For $m > 1$ the best response for the first player is to always bid $m - 1$. For $m = 1$ the best response is to bid 1 as well. A special case is for $m = 2$ where bidding 2 and 1 yields the same result. This can be seen from table 6.6. The regret can be defined four different ways depending on the value of $n$ and $m$. This can be seen in equation 6.7.

$$regret_1(n, m) = \begin{cases} (m - 1 - n)no_{items} & \text{for } m > n \\ (\frac{n}{2} - 1)no_{items} & \text{for } m = n > 1 \\ (m - 1)no_{items} & \text{for } m < n \\ 0 & \text{for } m = n = 0, 1 \\ \frac{no_{items}}{2} & \text{for } m = 1, n = 0 \\ 0 & \text{for } 0 = m < n \end{cases} \tag{6.7}$$

Equation 6.7 can then be contracted into a more simple equation which defines the maximal regret for player I. This can be done by looking at which value of $m$ gives the highest regret for each $n$. The two things which can cause the most regret, is either when the opponent bids the maximal amount or when he bids one less than you. When he bids maximum, the regret is $(max_{bid} - 1 - n)no_{items}$ and when he bids one less it is $((n - 1) - 1)no_{items}$. This leads to the function for finding the maximal regret as seen in equation 6.8.

$$maxregret_1(n) = max((max_{bid} - 1 - n)no_{items}, (n-2)no_{items}) \tag{6.8}$$

It can now be analyzed which strategy gives the minimal regret for a given game. Looking at a game with a maximal bid of 200 and considering not only integers as possible strategies leads to 100.5 as the unique value of the bid which minimizes the regret. The two integer values which minimize the regret for this game is 100 and 101. For a game with an odd number for the maximal bid, for example 199, the minimal regret is an integer number, and therefore the solution concept finds it in a single iteration when using games with an odd number for the maximal bid.
The value of the bid that minimizes the regret has a constant ratio in relation to the size of the maximal bid. This can be seen empirically from the test results, and can be found by minimizing the function. The minimum is where the two regrets are the same, which then can be used to find the relation to the size of the maximal bid.

$$(max_{bid} - 1 - n)no_{items} = (n-2)no_{items} \rightarrow fracmax_{bid} + 12 = n \tag{6.9}$$

It is easily seen, that the strategies proposed by the regret minimization can lead to a better outcome than that of the Nash Equilibria. Following the Nash Equilibrium, the maximal utility is $max(u_0(NE)) = 2no_{items}$ where the strategies from regret minimization have a possibility to $max(u_0(RM)) = \frac{max_{bid}}{2} no_{items}$.

**Summary:** The only difference from the analysis to the article was the results of the Nash Equilibrium. This was because the Nash Equilibrium used in the implementation was using weak equilibria as well. All other results followed those of the article. By changing the maximal bid a new behavior was observed. When the value was odd a single iteration of regret minimization gave the final result containing only a single strategy.

**Nash Bargaining Game:**
The results for Nash Bargaining Game are rather simple, since it only has a single parameter, and for the results of the article, this parameter is $b_max = [99, 100]$. All the different strategies are included in the Nash Equilibrium. For a game with $b_max = 100$, the non-iterated regret minimization produces two possible strategies, 50 and 51. Another iteration removes 51 so that 50 is the only strategy which minimizes the regret. For a game with $b_max = 99$, the game finds 50 as the only strategy with minimal regret, and thereby iteration gives no further information.
To reproduce the results from the article, the solution concepts must be used on games with a maximum bid which can be both even and odd. From the test results in table B.5 in appendix B.1 is it seen, that all the tests fit into the results from the article, so no deviation has been discovered.
The behavior of the solution concepts can be explained in the same way it was explained for the Bertrand Competition, but first the Nash Equilibria will be found. Looking at a small game with the maximal bid of 4 is it very easy to find all Nash Equilibria. The game can be seen in table 6.7. The Nash Equilibria is easily identified as all outcomes in the diagonal composed of $[(4,0),(3,1),(2,2),(1,3),(0,4)]$. For bigger or smaller games the same structure is held, and therefore can all strategies can be justifiable by using this solution concept.

Regret minimization gets a completely different result. Here the same arguments as for Bertrand Competition apply. So for games with an odd number as the maximal bid, the regret minimization finds a single strategy in the first iteration. The regret of an outcome can be found by looking at equation 6.10.

Table 6.7: A Nash Bargaining game with a max bid of 4.

| NB(4) | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0,0 | 0,1 | 0,2 | 0,3 | 0,4 |
| 1 | 1,0 | 1,1 | 1,2 | 1,3 | 0,0 |
| 2 | 2,0 | 2,1 | 2,2 | 0,0 | 0,0 |
| 3 | 3,0 | 3,1 | 0,0 | 0,0 | 0,0 |
| 4 | 4,0 | 0,0 | 0,0 | 0,0 | 0,0 |

$$regret_1(n,m) = \begin{cases} bid_{max} - m - n & \text{for } m+n <= bid_{max} \\ bid_{max} - m & \text{for } m+n > bid_{max} \end{cases} \tag{6.10}$$

From this it can be seen, that there are two situations which can maximize the regret, where either the opponent bids the minimal bid, or he bids so that the sum is exactly one higher than the limit. The regret in these situations can be seen in 6.11.

$$maxregret_1(n) = (bid_{max} - n, n-1) \tag{6.11}$$

From this it can be seen that the regret is at its minimum when the two parts of the function are the same, which leads to a minimum at $n_{regretmin} = (\frac{bid_{max}+1}{2})$. From this it can be seen, that when the maximal bid is an odd number, then a single integer has the minimal regret, and two minima exists when the maximal bid is even. The next iteration will then have to choose between the two strategies, and the lowest then has the minimal regret.

**Summary:** For this game no deviations were observed. The results of the experiments were exactly the same as in the article of Halpern and Pass [2]. No further experiments were conducted that were not covered by the article.

## 6.2 Experiments with average regret minimization

The adapted solution concept is now tested. The analysis is divided into two sections. The first is how the iterated behave in comparison with the non-iterated version. Followed by an analysis on how the concept relates to the two regret minimization solution concepts.

### 6.2.1 Comparison of average regret minimization, the iterated version and iterated regret minimization

Looking to the empirical evidence in the tables in section B.2 of the appendices, it is quickly observed, that for no game in the experiments does iteration of average regret minimization yield any different results than that of the non-iterated version. All the results of the experiments give either one or two strategies which have the optimal average regret. The games where iteration could bring a more specific result are those which have a result of two strategies. An example of this behavior could be

the Traveler's Dilemma. In this game, all valid games with $p < \frac{bid_{max}}{2} - 1$ have a result consisting of two strategies.

Table 6.8: A Traveler's Dilemma with a max bid of 8 and a penalty of 2.

| TD(8,2) | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| 2 | 2,2 | 4,0 | 4,0 | 4,0 | 4,0 | 4,0 | 4,0 |
| 3 | 0,4 | 3,3 | 5,1 | 5,1 | 5,1 | 5,1 | 5,1 |
| 4 | 0,4 | 1,5 | 4,4 | 6,2 | 6,2 | 6,2 | 6,2 |
| 5 | 0,4 | 1,5 | 2,6 | 5,5 | 7,3 | 7,3 | 7,3 |
| 6 | 0,4 | 1,5 | 2,6 | 3,7 | 6,6 | 8,4 | 8,4 |
| 7 | 0,4 | 1,5 | 2,6 | 3,7 | 4,8 | 7,7 | 9,5 |
| 8 | 0,4 | 1,5 | 2,6 | 3,7 | 4,8 | 5,9 | 8,8 |

In table 6.8 a small Traveler's Dilemma is given. For this game, average regret minimization finds strategy 4 and 5 to be optimal for player I. In the first iteration both of these strategies have the same average regret of $\frac{12}{7}$ which also turns out to be the minimum. The same happens in the second iteration were both have an average regret of $\frac{3}{2}$. There seems to be some connection between the average regret in the first and the second iteration since iteration does not remove any more strategies in any situation. Looking closer at the game a pattern emerges, there is a relation between the sum of the utilities of a strategy and the average regret. The difference between two strategies' collected regret and their utility sum is the same. This can be seen in table 6.9 where the game is represented, as well as the sum of utilities and the collected regret.

Table 6.9: TD(8,2) with the collected regret and sum of utilites.

| TD(8,2) | 2 | 3 | 4 | 5 | 6 | 7 | 8 | Sum of utilities | Collected regret |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 2,2 | 4,0 | 4,0 | 4,0 | 4,0 | 4,0 | 4,0 | 26 | 15 |
| 3 | 0,4 | 3,3 | 5,1 | 5,1 | 5,1 | 5,1 | 5,1 | 28 | 13 |
| 4 | 0,4 | 1,5 | 4,4 | 6,2 | 6,2 | 6,2 | 6,2 | 29 | 12 |
| 5 | 0,4 | 1,5 | 2,6 | 5,5 | 7,3 | 7,3 | 7,3 | 29 | 12 |
| 6 | 0,4 | 1,5 | 2,6 | 3,7 | 6,6 | 8,4 | 8,4 | 28 | 13 |
| 7 | 0,4 | 1,5 | 2,6 | 3,7 | 4,8 | 7,7 | 9,5 | 26 | 15 |
| 8 | 0,4 | 1,5 | 2,6 | 3,7 | 4,8 | 5,9 | 8,8 | 23 | 18 |

If this is true, then the result of average regret minimization can be found much easier than the algorithm described in section 3. Two different theorems are needed in order to prove that the correlation is true. A more formal definition of these correlations is given in Theorem 6.2.2 and 6.2.1.

**Theorem 6.2.1.** *$u_{sum}(s_1) = u_{sum}(s_2)$ if and only if $regret_{average}(s_1) = regret_{average}(s_2)$*

**Theorem 6.2.2.** *Given a set of strategies $[s_1, ..., s_n]$ with different utilities, for these the following holds:*
*$u_{sum}(s_1) < ... < u_{sum}(s_n)$ if and only if $regret_{average}(s_1) > ... > regret_{average}(s_n)$*

First 6.2.1 is proved. The theorem states, that if two strategies have the same utility sum, it implies that they also have the same average regret, and that the same average regret implies they also have the same utility sum. To prove this, a generic game is needed. For this game, player I has $n$ strategies which is named $S = [s_1, ..., s_n]$. For two strategies $s_1, s_2 \in S$ we know $u_{sum}(s_1) = \sum_{i=1}^{m} u(n, i)$ and $regret_{average}(s_i) = \sum_{i=1}^{m} u(n_{max}, i) - u(n, i)$ where $n_{max}$ indicates an imaginary strategy where for which holds $u(n_{max}, m) = max(u(i, m))$ where $i \in [1, .., n]$. The connection can then be proven as seen bellow.

*Proof.*
$regret_{average}(s_1) = regret_{average}(s_2) \Leftrightarrow$
$\sum_{i=1}^{m} u(n_{max}, i) - u(s_1, i) = \sum_{i=1}^{m} u(n_{max}, i) - u(s_2, i) \Leftrightarrow$
$\sum_{i=1}^{m} u(n_{max}, i) - \sum_{i=1}^{m} u(s_1, i) = \sum_{i=1}^{m} u(n_{max}, i) - \sum_{i=1}^{m} u(s_2, i) \Leftrightarrow$
$\sum_{i=1}^{m} u(s_1, i) = \sum_{i=1}^{m} u(s_2, i) \Leftrightarrow$
$u_{sum}(s_1) = u_{sum}(s_2)$ □

The proof for the other theorem is of the same nature as proof 6.2.1 and is therefore omitted. The two theorems can then be used to explain why it does not make sense to iterate the average regret minimization. This is because strategies of equal sum will have the same regret relating to any other strategy. After the first iteration $n_{max}$ changes and so does the regret, but since the utility sum is unchanged, so is the relation between the two strategies' average regret. From the observations of the relation between the average regret and the sum, the behavior in the different games can be explained. An expression can be defined, which tells which strategies have the smallest average regret. If $u_{sum}(n) = max(u_{sum}(n_i), n_i = 2, ..., bid_{max})$ then $n$ has the minimal average regret. Since the expression can have more than one solution $n$ is necessarily unique. This expression can be applied to any game in strategic form in order to find the strategies with minimal average regret. An algorithm which uses this to find the minimal average regret will have the same time complexity $O(nm)$, since it still has to examine all outcomes.

**Comparison between average regret and iterated regret minimization**

In the next experiment the results will be compared to those of the basic regret minimization, with single and multiple iterations. In the tables in section B.2 of the appendices, the results of the experiment are described. In the tables the results of average regret minimization are given along with the iterated version. Since the results of the iterated version are the same for all the games, these results of the iterated version are not discussed any further. It is seen that for all the games, they have similar results as to those of regret minimization.

**Travelers's Dilemma:** For this game, the results are more precise than the non-iterated version of regret minimization, but they are a little less precise than the results of the iterated version. It can be seen, that the results are closely tied to those of iterated regret minimization, it only has a single strategy more in the result.

**Exponential Centipede Game:** Here the results are very similar. The results of the iterated version are identical to those of the average version. Only a little difference is seen with the non-iterated version, where it finds two strategies, the average regret minimizations instead finds the specific strategy which iterated regret minimizations finds.

**Linear Centipede Game:** This is the game for which the results contain the most differences from those of the two regret minimization solution concepts. It can be seen, that the results are still very similar to those of iterated regret minimization. In the all cases it is at least as precise as the iterated regret minimization, and in the cases where it returns more than two strategies, average regret minimization returns a more precise result consisting of only two strategies. The implications of this result will be examined further in a later experiment where the expected utilities are examined.

**Bertrand Competition:** For most games, the results are identical to those of the iterated regret minimization. Only in the special case where $p = 1$ and the maximal bid is odd the results only match that of the non-iterated version. This leads to a more precise result than the non-iterated version in almost every Bertrand Competition.

**Nash Bargaining Game:** Here average minimization produce the same results as the non-iterated version of regret minimization. The results of the iterated version are slightly more precise, because it removes one more strategy.

#### Evaluation

In the following list the most important conclusions are mentioned. The conclusions are drawn from the analysis and experiments of this section.

- From the analysis it can be seen, that due to a constant connection between the average regret and the sum of the utilities of a strategy, it is possible to find the strategies with a minimal average regret by using a simplified algorithm. The new algorithm is easier to implement, but keeps the same complexity as the algorithm described in chapter 5.
- Due to the connection between the strategies, with the minimal average regret, iterations do not make sense for the average regret minimization solution concept. From this it can be concluded, that average regret minimization does only need a single iteration to obtain a result which is only a little less precise in some cases.
- Average regret minimization produces results very similar to those of iterated regret minimization. Only in a few cases does it produce results which contain more strategies and thereby loses some precision. The implication of the differences in the result will be analyzed in a later experiment.

### 6.2.2 Iterations of the algorithms

It was seen in the previous section, that average regret minimization only needs a single iteration to find the optimal result. The experiments of this section are made in order to evaluate how many differences there is in the iterations of the different solution concepts. This is because regret minimization and average regret minimization have the same time complexity.
By browsing through the results in appendix B.3, it can be seen, that iterated regret minimization uses a maximum of 3 iterations for the games examined. From the algorithm, it is known that the last iteration is used to check whether or not any more strategies can be removed. This means, that only two iterations are used for removing strategies. This leads to the question: "If average regret minimizations produce results of similar quality, then how much better is the computation?". To answer this, it must first be examined whether or not a game exists where iterated regret minimization iterates more than 3 times. A modified version of the game in table 6.8 can be used to verify this. In table 6.10 the modified version of the game is seen. The modification is marked in bold and italic font. This little modification causes the iterated regret minimization to have an additional iteration. The modification is simple to raise the utility by one. The maximal regret for each strategy is also seen for each of the iterations in the table. For each of the utilities the strategies consist of an additional iteration can be added. This leads to the maximal number of iterations being identical to the number of strategies of either player who has the lowest number of strategies. It implies that iterated regret minimization uses up to a linear number of iterations.

Table 6.10: A modified Traveler's Dilemma with a max bid of 8 and a penalty of 2.

| TD(8,2) | 2 | 3 | 4 | 5 | 6 | 7 | 8 | $i_1$ | $i_2$ | $i_3$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2,2 | 4,0 | 4,0 | 4,0 | 4,0 | 4,0 | 4,0 | 5 | | |
| 3 | 0,4 | 3,3 | 5,1 | 5,1 | 5,1 | 5,1 | 5,1 | 4 | | |
| 4 | 0,4 | 1,5 | 4,4 | 6,2 | 6,2 | 6,2 | 7,2 | 3 | 2 | 1 |
| 5 | 0,4 | 1,5 | 2,6 | 5,5 | 7,3 | 7,3 | 7,3 | 3 | 2 | 2 |
| 6 | 0,4 | 1,5 | 2,6 | 3,7 | 6,6 | 8,4 | 8,4 | 3 | 3 | |
| 7 | 0,4 | 1,5 | 2,6 | 3,7 | 4,8 | 7,7 | 9,5 | 3 | 3 | |
| 8 | 0,4 | 1,5 | 2,6 | 3,7 | 4,8 | 5,9 | 8,8 | 3 | 3 | |

From the results of this experiment and the results of the first experiment, it can now be analyzed how much computation time can be spared by using average regret minimization instead of iterated regret minimization. The way the algorithm works, it implies that the fewer strategies it removes each iteration, the more time it has to examine each of the remaining utilities. The following list encompasses each of the games and the number of strategies examined.

**Traveler's Dilemma:** With this game it is seen, that regret minimization removes $bid_{max} - 2 - 2p$ in the first iteration. In the next iteration, it removes all but one strategy. This results in the total number of strategies examined being $bid_{max} + 2p$. From this it is easy to see that the higher the punishment the more strategies are examined more than once. Since average regret minimization only examines each strategy once, there is an improvement of $2p$. This is only true for games where iterated regret minimization have 3 iterations.

**Exponential centipede game:** For this game there is very little improvement to be gotten from switching to average regret minimization. This is because the iterated regret minimization in most cases finds the result in a single iteration. The few cases where it does take an additional iteration, it is only to remove a single strategy. So the number of strategies that average regret minimization examines less are only 1.

**Linear centipede game:** The number of strategies that iterated regret minimization examines depends on the parameters of the game. The larger the punishment is the less strategies are removed in the first iteration. From the first experiment we know that the strategies removed in the first iteration can be either those who represent rounds lower than $k-p$ or $k-p+1$ where $k$ is the number of rounds. In the case where $p$ is an odd number, no more strategies are removed in the next iteration. Average regret minimization does in this case remove more strategies. It removes all but two strategies. The implications of the different results are examined in a later experiment. The other case is when $p$ is even. In this case are all but one strategy is removed in the second iteration. This leads to the total number of strategies examined $\frac{k+p}{2} + 1$. The $+1$ strategy is from the last iteration where it verifies that no more strategies can be removed.

**Bertrand competition:** The number of examined strategies depends on the maximal bid. If it is odd, then all but one strategy is removed in a single iteration, which leads to the same number of strategies examined by both iterated and average regret minimization. When the maximal bid is even, then iterated regret minimization removes one strategy less in the first iteration. In the second iteration it then removes a single iteration more. This leads to some situations where average regret minimization will examine a single strategy less than iterated regret minimization.

**Nash bargaining game:** The difference between the two solution concepts is the same for the Nash Bargaining Game as for Bertrand Completion. If the maximal sum is even, then average regret minimization saves the examination of a single strategy. When the maximal sum is odd they examine the same number of strategies.

For all of the different games, iterated regret minimization iterates at least one more time than average regret minimization. This is because of the final iteration being performed in order to confirm that no more strategies can be removed. The number of strategies examined in this iteration is easily found by simply looking at the number of strategies in the result. The only game where the result of iterated regret minimization is not just a single strategy is in the linear centipede game. In the case where the result is $\frac{k-p}{2}$ strategies, the last iteration examines the $\frac{k-p}{2}$-strategies in the last iteration without removing any. For all the other games, the last iteration only examines the single remaining strategy.

#### Evaluation

From this experiment and analysis it can be seen, that Average Regret Minimization uses less iterations than Iterated Regret minimization, but for the games used in this thesis the differences are not that great because iterated regret minimization only iterates a maximum of 3 times. It is seen that games exist for which iterated regret minimization iterates more than 3 times, so for these games there are more improvement to be found. In order to finally conclude, that average regret minimization is at least as good as iterated regret minimization, a comparison of the expected outcome of the different strategies they produce is required. This analysis is performed in a later experiment.

### 6.2.3 Impact of mixed strategies

Here it will be examined how many times a game must be executed in order to make the average outcome of the games reflect the expected outcome. This experiment is performed in order to see how many times the games need to be repeated in the upcoming experiments. The game which is used is the Traveler's Dilemma with a maximal bid of 100 and a punishment of 2. Player I uses a completely uniform random strategy and player II always uses the highest possible strategy. The number of times the game is repeated increases logarithmically in the form of $[1, 10, 100, ..., 100000]$. Each of these experiments are repeated and the maximum and minimum average utility is recorded. They are repeated until no new minima or maxima is encountered in ten consecutive experiments. In figure 6.1 the maximal deviation can be seen when increasing the number of iterations the number of iterations.

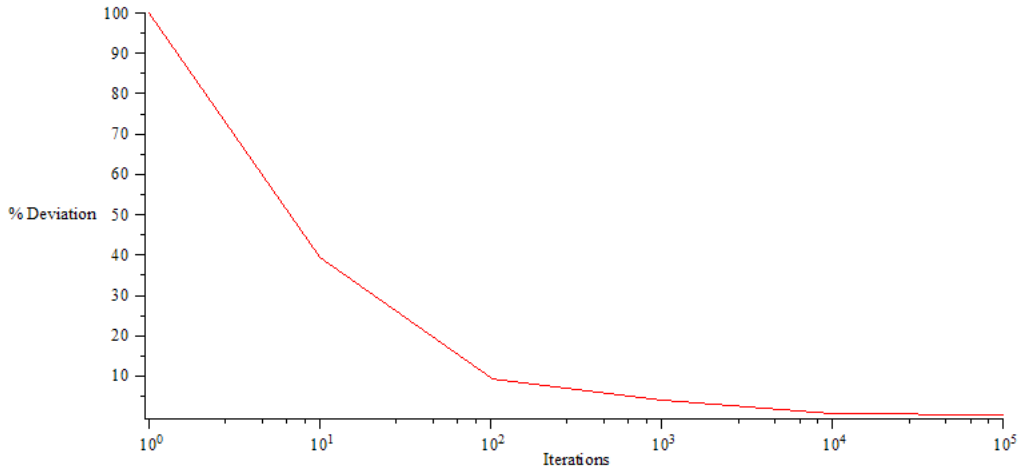


Figure 6.1: Deviation from the mean.

From this experiment it can be concluded, that in order to get under a 0.5% deviation from the expected outcome the average must be measured over 100000 iterations. This result can then be used for the next experiments where the different solution concepts are tested against either other or against some random strategies.

### 6.2.4 Performance versus mixed strategies

Here each of the solution concepts play against a player which plays different mixed strategies. This experiment is conducted in order to have some empirical evidence for the expected utility of the different solution concepts when playing against a real life opponent. All the experiments have been completed using the average over 100000 iterations. When the result of a solution concept contains more than one strategy, a mixed strategy id played which uniformly randomizes between the strategies suggested by the solution concept.

Before the experiment can be evaluated, an analysis of how to compare two different outcomes is needed. The problem itself is rather simple: "How to determine if one outcome is better than another?". In order to compare outcomes some formalities are need. The two outcomes are called $a$ and $b$. Each of them consist of two utilities $(u_{a1}, u_{a2})$ and $(u_{b1}, u_{b2})$. The notation $a > b$ defines that outcome $a$ is strictly better than outcome $b$. This is when both utilities are higher in one of the outcomes than the other. All the games tested in this thesis are non-cooperative games. Therefore the outcomes do not need to be strictly better, the only requirement is $(u_{a1} > u_{b1})$ in order for $a$ to be a better outcome for player I.
Another way a player can think is to try and minimize the outcome of the opponent. In these cases the player is indifferent to his own outcomes. Instead of remodeling the algorithms the outcomes can be changed. To simulate this behavior the utilities of player I can be set to the opposite of player II, i.e. $u_{a1'} = -u_{a2}$. This is the same as happens in zero-sum games.
A third way a player can think to optimize the sum of the utilities of the players. Simulating this behavioris done med remodeling the utilities by $u_{a1'} = u_{a1} + u_{a2}$.

The results of this experiment can now be evaluated. The results are seen in the appendices in section B.5. The results can be divided into two subgroups, one where the solution concepts compete against an opponent which plays mixed strategies. In the first the probability is uniformly distributed between all pure strategies, and in the other the probabilities are higher for some of the strategies which have a possibility of giving a better utility.

#### Uniform mixed strategy

When the different solution concepts are played out against a completely uniform random opponent, the results vary depending on the game. The results are seen in table B.14 of the appendices.

**Traveler's Dilemma:**
Looking at the result for the two Traveler's Dilemma it can be seen, that using one of the three regret minimizing solution concepts provide a much better outcome for both players. The outcome of iterated regret minimization is the best with a very small margin over average regret minimization. Looking at the average utility of the opponent an unexpected behavior is observed. In the case where player I

adopts a regret minimizing strategy the random player will get a better expected utility. From table B.15 it can be seen, that it is still better to use one of the solution concepts when the opponent plays the mixed strategy. So when playing Traveler's Dilemma against a uniform random opponent the best solution concept is either iterated regret minimization or average regret minimization. When taking the algorithm into consideration average regret minimization is the better choice due to the fewer iterations used.

**Exponential centipede game:**
The results for the exponential centipede game are not very interesting. For these games the proposed strategy from each of the solution concepts are very similar, which is also expressed by the average utilities. This is because of the strategies being identical or very similar for each of the solution concepts. Because of this, the recommended solution concept is not based on the result, but on the algorithm. regret minimization and average regret minimization share the same number of iterations and are therefore recommended for this game.

**Linear centipede game:**
The linear centipede game provides some more interesting results. For games with even penalty, the result is the same for all the regret minimizing solution concepts, and playing random uniform yields a worse average utility. When encountering games with an odd penalty it is better to use average regret minimization. This is because of the difference in the strategies proposed by the two solution concepts. The higher the penalty, the higher the difference, therefore for a high penalty the best solution concept is average regret minimization. It improves the average utility by 12.5% but the utility of the opponent decreases with 59.1%. But the average utility is still a better solution as long as it increases for player I.

**Bertrand Competition:**
The results of the Bertrand Competition lean towards iterated regret minimization and average regret minimization. These solution concepts produce the same strategies and therefore also very similar expected utilities. Using regret minimization has a lower expected utility, but is very small. Playing one of the solution concepts in all cases provides a better expected utility than playing random. Because average regret minimization uses less iterations it is preferred from iterated regret minimization.

**Nash Bargaining Game:**
The results of Nash Bargaining Game are very trivial. All the solution concepts produce equally good average utilities. And therefore regret minimization or average regret minimization are the suggested solution concepts. Using a random strategy only provides worse utility and is therefore not recommended.

**Non-Uniform Mixed strategy:**

Now the opponent is no longer playing a uniform random mixed strategy. Instead it adopts a mixed strategy where strategies which can give a higher utility have a higher probability. Each of the following paragraphs describes the random strategy for a type of game and the results of the experiments. The results are seen in the same sections as for random uniform.

**Traveler's Dilemma:**
For the Traveler's Dilemma, the utilities increase the higher you bid, it is therefore natural to adopt a strategy where higher bids have higher probability. To simulate this behavior the cymbal algorithm is utilized. The parameter is the number of strategies of the opposing players. Changing to the weighted random strategy increases the average utility in all cases. The magnitude of the increase depends on the parameters of the game and what solution concepts are used. The average utility is higher for both players. The average utility is still highest for the opponent in all cases. For $TD(100, 2)$ the increase in the utility is the same for all solution concepts and players. When increasing the penalty the expected increase in utilities changes. For $TD(100, 20)$ the increase is much bigger for player I than the opponent. Player II only has the largest increase in the expected utility when player I plays regret minimization. When player I plays iterated regret minimization or average regret minimization he only increases his average utility by 1.5.

**The centipede games:**
The centipede game employs a different random strategy. It is the random strategy where there is a probability $p$ for stopping each round. A single game is selected for each type of centipede game. For both types of games the result is indifferent to the choice of solution concept. It is seen, that for low $p$ player II gets a better average utility than players I. This is because player I always plays a strategy where he stops later. When $p$ is low the opponent has a high probability of stopping earlier than player I resulting in a better outcome. The outcome is better for both players when the opponent has a high probability $p$ for continuing. This is because of the game mechanic, which ensures that the utilities are increasing along with the number of rounds.

**The Bertrand Competition:**
The Bertrand Competition uses the cymbal algorithm as well, where the parameter is $\frac{bid_{max}}{2}$. When the opponent uses this strategy none of the expected utilities of player I are affected. The only change is that the opponent has a large increase in his expected utilities. This is because it still has the same probability for player II to bid higher and lower than player I, therefore his expected outcome is not affected. When player II bids less then player one, there is a higher possibility that he bids close to Player I which yields a better outcome.

**The Nash Bargaining Game:**
The Nash Bargaining Game uses the same random strategy as the Bertrand Competition with parameter $\frac{sum_{max}}{2}$. This has the same effect as for the Bertrand Competition. Player I's expected utilities remain unaffected while the opponent increases his outcomes. The reason for this is the same behavior for the Bertrand Competition.

**Evaluation**

It can be seen, that in all cases average regret minimization has at least as good expected utility as iterated regret minimization. When playing against a mixed strategy it is in no case better to play mixed yourself. In all cases this yields a worse average outcome and can therefore not be recommended. This experiment also proves, that when playing the games used throughout this thesis against a random opponent, average regret minimization is the preferred solution concept because it has at least as good an outcome as iterated regret minimization and uses less iterations to solve the games.

### 6.2.5 Solution concepts versus solution concepts

Here the performance of each of the solution concepts is measured when they are playing against other solution concepts in the different games. The reason for this experiment is to evaluate the performance of each of the solution concepts against the other solution concepts. Comparing the result of each of the games it is seen, that average regret minimization's average utility is in most cases at least as good as the others. The few situations where the average utilities of average regret minimization are worse than the opponents is mostly when competing against NE. Here the player which plays NE gets a better utility in all other games than the Nash Bargaining Game. This is because the Nash Equilibria for these games selects a strategy where one of the lowest bids are selected or to stop in the first rounds. The other strategies tend to select higher bid or to stop later. It is therefore natural that bidding low would yield a higher average utility than the opponent. If the player tries to minimize the outcome of the other player this strategy is the right choice, but if the player tries to optimize his own utility, then it is better to select another solution concept. The reason why NE does not have a higher outcome than the opponent in the Nash Bargaining Game, is because the Nash Equilibrium selects any possible strategy. This results in a uniformly distributed random strategy which in the cases yield a utility of 0. In the case where he bids less, the outcomes are uniformly distributed between all the different possibilities, where the other player always gets the maximum of these.
A game for which average regret minimization does not perform as well as the other solution concepts is the Nash Bargaining Game. Here it actually has a worse expected utility than iterated regret minimization. This is because the solution concepts find two different strategies which are optimal and iterated regret minimization only finds one. For $sum_{max} = 100$ the two strategies are 50 and 51. iterated regret minimization only has 50 as the strategy. Half the time average regret minimization will choose 51 which results in an utility of 0 for both players. This is also seen in the results, where the average utility is half of the utility of iterated regret minimization. Regret minimization produces the same strategies as average regret minimization and therefore has the same average utilities.

It can then be seen, when playing against the other solution concepts, that for all games, except Nash Bargaining, average regret minimization is recommended. For the Nash Bargaining game it is instead iterated regret minimization.

# Chapter 7

# Conclusion

Throughout this thesis many subjects have been touched. First the more general areas of game theory have been described in order to establish a foundation for the experiments, the games and solution concepts used. On that background, the games and solution concepts were described and algorithms were created for each of the solution concepts. The algorithms were then implemented along with the games in a testing tool which was used for all the experiments.
Two different goals were set for the experiments: To verify the results from [2] by producing empirical and theoretical results, and to determine the relation between iterated regret minimization as defined by Halpern and Pass in [2] and average regret minimization by V. Goranko. The results of the verification was that all games in general followed the results of [2]. However, most of the games had a slight variation with some of the results. A summary of the variations can be seen in the list bellow. Moreover a formal argument was given for the behavior of each of the solution concepts of the different games.

**Traveler's Dilemma:** The limit of the penalty when the outcome pattern changes. This is the suggested strategy for games with a high penalty.

**Exponential Centipede Game:** In the case of $p = 1$, the single iteration version of the regret minimization had a deviation from the results in the article.

**Linear Centipede Game:** The only deviation from the article was that the result of the solution concepts was not tied to both the parameters.

**Bertrand Competition:** The only difference from the analysis to the article was the result of the Nash Equilibrium.

**Nash Bargaining Game:** For this game no deviations was observed.

For average regret minimization a range of experiments was performed in order to clarify, if average regret minimization could be a better solution concept compared to iterated regret minimization. The different experiments are given in the list below.

- Relation between iterated and non-iterated average regret minimization.

- Relation between average regret minimization and the original regret minimization solution concept.
- Compared iterations and results of average regret minimization and the original regret minimization solution concept.
- Compared expected utility of the different solution concepts against different mixed strategies.
- Compared expected utility of the different solution concepts against each other.

From these experiments it can then be concluded, that it does not make sense to iterate over average regret minimization, since it produces the same results as the non-iterated version in all cases.
Within the experiments were a better algorithm for average regret minimization created. It had the same time complexity but were much simpler than the one described in chapter 3.
For all the games used throughout this thesis average regret minimization produces results very similar or better than those of iterated regret minimization. So it can be concluded, that average regret minimization is an optimization of iterated regret minimization, because regretminimization and average regret minimization have the same time complexity. However, for the games in this thesis the advantages of average regret minimization were very small, because in these games iterated regret minimization is used at most in 3 iterations. A game was presented were more iterations were used by iterated regret minimization, and it was proven that games exists where several iterations are used.

The goals stated in section 4 were all fulfilled.

# Referencess


[1] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1991.

[2] Joseph Y. Halpern and Rafael Pass. Iterated regret minimazation: A new solution concept. *Cornell University*, 2009.

[3] P. B. Linhart and R. Radner. Minimax-regret strategies for bargaining over several variable. *Journal of Economic Theory 48*, 1989.

[4] J. Niehans. Zur preisbildung bei ungewissen erwartungen. *Scbweizerische Zietschrift für Volkswirtschaft und Statistik 84 (5)*, 1948.

[5] Noam Nisan, Tim Roughgarden, Éva Tardos, Vijay V. Vaziran, and Christos H. Papadimitrioui. *Algorithmic Game Theory*. Cambridge Univerity Press, 2007.

[6] L. J. Savage. The theory of statistical decision. *Journal of the American Statistical Association*, 1951.

[7] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

# Appendix A

# Abbreviations

Table A.1: Abbreviations of the algorithms

| Solution concept | Abbreviation |
|---|---|
| Nash equilibrium | NE |
| Regret Minimazation | RM |
| Iterated Regret Minimazation | IRM |
| Average Regret Minimazation | ARM |
| Iterated Average | IARM |

Table A.2: Abbreviations of the games

| Solution concept | Abbreviation | Parameters |
|---|---|---|
| Traveler's Dilemma | $TD(mv, p)$ | mv = Max Value, p = Punishment |
| Exponential Centipede Game | ECG(r,p) | r = Rounds, p = Punishment |
| Linear Centipede Game | LCG(r,p) | r = Rounds, p = Punishment |
| Bertrand Competition | BC(mb,i) | mb = Maximal Bid, i = number of Items |
| Nash Bargaining | NB(mv) | mv = Maximal Value |

# Appendix B

# Test results

## B.1 Test - Reproduction of results

### B.1.1 Results from article

Table B.1: The results from [2] of the games.

| Game | NE | RM | IRM |
|---|---|---|---|
| TD$(100, p < 50)$ | 2 | [100-2p,100] | 100-2p+1 |
| TD$(100, p \geq 50)$ | 2 | $-\|\|-$ | $-\|\|-$ |
| ECG$(k, p)$ | 1 | k | $-\|\|-$ |
| LCP(k,p), k and p even | 1 | [k-p+1,k-p+3,...,k-1] | k-p+1 |
| LCP(k,p), k and p odd | 1 | [k-p+1,k-p+3,...,k-1] | $-\|\|-$ |
| BC(200,100) | [0,1] | [100,101] | 100 |
| NB(100) | [0,...,100] | [50,51] | 50 |
| NB(99) | [0,...,99] | 50 | $-\|\|-$ |

### B.1.2 Traveler's Dilemma

Table B.2: TD test results

| Game | NE | RM | IRM |
|---|---|---|---|
| TD(100,2) | 2 | [96,...,100] | 97 |
| TD(101,2) | 2 | [97,...,101] | 98 |
| TD(50,2) | 2 | [46,...,50] | 47 |
| TD(51,2) | 2 | [47,...,51] | 48 |
| TD(150,2) | 2 | [146,...,150] | 147 |
| TD(151,2) | 2 | [147,...,151] | 148 |
| TD(101,3) | 2 | [95,...,101] | 96 |
| TD(100,1) | [2,...,100] | [98,...,100] | [98,...,100] |
| TD(100,3) | 2 | [94,...,100] | 95 |
| TD(100,4) | 2 | [92,...,100] | 93 |
| TD(100,5) | 2 | [90,...,100] | 91 |
| TD(100,20) | 2 | [60,...,100] | 61 |
| TD(100,30) | 2 | [40,...,100] | 41 |
| TD(100,48) | 2 | [4,...,100] | 5 |
| TD(100,49) | 2 | 3 | 3 |
| TD(100,50) | 2 | 3 | 3 |
| TD(100,51) | 2 | 3 | 3 |
| TD(100,96) | 2 | 3 | 3 |
| TD(100,97) | 2 | [2,3] | 2 |
| TD(100,98) | 2 | 2 | 2 |
| TD(100,100) | 2 | 2 | 2 |

### B.1.3 Exponential centipede game

Table B.3: ECG test results

| Game | NE | RM | IRM |
|---|---|---|---|
| ECG(10,1) | 1 | 9 | 9 |
| ECG(15,1) | 1 | [13,...,15] | 13 |
| ECG(20,1) | 1 | 19 | 19 |
| ECG(25,1) | 1 | [23,...,25] | 23 |
| ECG(30,1) | 1 | 29 | 29 |
| ECG(30,2) | 1 | 29 | 29 |
| ECG(25,2) | 1 | 23 | 23 |
| ECG(30,3) | 1 | 29 | 29 |
| ECG(25,3) | 1 | 23 | 23 |
| ECG(30,6) | 1 | 29 | 29 |
| ECG(25,6) | 1 | 23 | 23 |
| ECG(30,7) | 1 | 29 | 29 |
| ECG(25,7) | 1 | 23 | 23 |
| ECG(30,30) | 1 | 29 | 29 |
| ECG(25,30) | 1 | 23 | 23 |

### B.1.4 Linear centipede game

Table B.4: LCG test results

| Game | NE | RM | IRM |
|---|---|---|---|
| LCG(10,2) | 1 | 9 | 9 |
| LCG(11,2) | 1 | [9,11] | 9 |
| LCG(20,2) | 1 | 19 | 19 |
| LCG(21,2) | 1 | [19,21] | 19 |
| LCG(30,2) | 1 | 29 | 29 |
| LCG(31,2) | 1 | [29,31] | 29 |
| LCG(10,3) | 1 | [7,9] | [7,9] |
| LCG(11,3) | 1 | [7,...,11] | [7,...,11] |
| LCG(20,3) | 1 | [17,19] | [17,19] |
| LCG(21,3) | 1 | [17,...,21] | [17,...,21] |
| LCG(30,3) | 1 | [27,29] | [27,29] |
| LCG(31,3) | 1 | [27,...,31] | [27,...,31] |
| LCG(30,10) | 1 | [21,...,29] | 21 |
| LCG(30,20) | 1 | [11,...,29] | 11 |
| LCG(30,25) | 1 | [5,...,29] | [5,...,29] |
| LCG(30,26) | 1 | [5,...,29] | 5 |
| LCG(30,27) | 1 | [3,...,29] | [3,...,29] |
| LCG(30,28) | 1 | [3,...,29] | 3 |
| LCG(30,29) | 1 | [1,...,29] | [1,...,29] |
| LCG(30,30) | 1 | 1 | 1 |
| LCG(30,31) | 1 | 1 | 1 |
| LCG(30,100) | 1 | 1 | 1 |
| LCG(30,101) | 1 | 1 | 1 |
| LCG(31,20) | 1 | [11,...,31] | 11 |
| LCG(31,25) | 1 | [5,...,31] | [5,...,31] |
| LCG(31,26) | 1 | [5,...,31] | 5 |
| LCG(31,27) | 1 | [3,...,31] | [3,...,31] |
| LCG(31,28) | 1 | [3,...,31] | 3 |
| LCG(31,29) | 1 | [1,...,31] | [1,...,31] |
| LCG(31,30) | 1 | 1 | 1 |
| LCG(31,31) | 1 | 1 | 1 |
| LCG(31,100) | 1 | 1 | 1 |
| LCG(31,101) | 1 | 1 | 1 |
| LCG(30,1) | [1,...,29] | 29 | 29 |

### B.1.5 Bertrand competition

Table B.5: BC test results

| Bertrand competition | NE | RM | IRM |
|---|---|---|---|
| BC(200,100) | [0,...,2] | [100,101] | 100 |
| BC(100,100) | [0,...,2] | [50,51] | 50 |
| BC(300,100) | [0,...,2] | [150,151] | 150 |
| BC(199,100) | [0,...,2] | 100 | 100 |
| BC(201,100) | [0,...,2] | 101 | 101 |
| BC(200,75) | [0,...,2] | [100,101] | 100 |
| BC(200,50) | [0,...,2] | [100,101] | 100 |
| BC(200,2) | [0,...,2] | [100,101] | 100 |
| BC(199,75) | [0,...,2] | 100 | 100 |
| BC(201,75) | [0,...,2] | 101 | 101 |
| BC(200,1) | [0,...,2] | 100 | 100 |
| BC(201,1) | [0,...,2] | [100,101] | 100 |

### B.1.6 Nash Bargaining Game

Table B.6: NB test results

| Nash Bargaining | NE | RM | IRM |
|---|---|---|---|
| NB(50) | [0,...,50] | [25,26] | 25 |
| NB(100) | [0,...,100] | [50,51] | 50 |
| NB(99) | [0,...,99] | 50 | 50 |
| NB(101) | [0,...,101] | 51 | 51 |
| NB(200) | [0,...,200] | [100,101] | 100 |

## B.2 Test - Average regret minimization

### B.2.1 Traveler's Dilemma

Table B.7: TD test results for ARM and IARM

| Game | ARM | IARM |
|---|---|---|
| TD(100,2) | [96,97] | −\|\|− |
| TD(101,2) | [97,98] | −\|\|− |
| TD(150,2) | [146,147] | −\|\|− |
| TD(101,3) | [95,96] | −\|\|− |
| TD(100,1) | [98,99] | −\|\|− |
| TD(100,3) | [94,95] | −\|\|− |
| TD(100,4) | [92,93] | −\|\|− |
| TD(100,5) | [90,91] | −\|\|− |
| TD(100,20) | [60,61] | −\|\|− |
| TD(100,30) | [40,41] | −\|\|− |
| TD(100,48) | [4,5] | −\|\|− |
| TD(100,49) | 2,3 | −\|\|− |
| TD(100,50) | 2 | −\|\|− |
| TD(100,51) | 2 | −\|\|− |
| TD(100,96) | 2 | −\|\|− |
| TD(100,97) | 2 | −\|\|− |
| TD(100,98) | 2 | −\|\|− |
| TD(100,100) | 2 | −\|\|− |

### B.2.2 Exponential centipede game

Table B.8: ECG test results for ARM and IARM

| Game | ARM | IARM |
|---|---|---|
| ECG(10,1) | 9 | −\|\|− |
| ECG(15,1) | 13 | −\|\|− |
| ECG(20,1) | 19 | −\|\|− |
| ECG(25,1) | 23 | −\|\|− |
| ECG(30,1) | 29 | −\|\|− |
| ECG(30,2) | 29 | −\|\|− |
| ECG(25,2) | 23 | −\|\|− |
| ECG(30,3) | 29 | −\|\|− |
| ECG(25,3) | 23 | −\|\|− |
| ECG(30,6) | 29 | −\|\|− |
| ECG(25,6) | 23 | −\|\|− |
| ECG(30,7) | 29 | −\|\|− |
| ECG(25,7) | 23 | −\|\|− |
| ECG(30,30) | 29 | −\|\|− |
| ECG(25,30) | 23 | −\|\|− |

### B.2.3 Linear centipede game

Table B.9: LCG test results

| Game | ARM | IARM |
|---|---|---|
| LCG(10,2) | 9 | −\|\|− |
| LCG(11,2) | 9 | −\|\|− |
| LCG(20,2) | 19 | −\|\|− |
| LCG(21,2) | 19 | −\|\|− |
| LCG(30,2) | 29 | −\|\|− |
| LCG(31,2) | 29 | −\|\|− |
| LCG(10,3) | [7,9] | −\|\|− |
| LCG(11,3) | [7,9] | −\|\|− |
| LCG(20,3) | [17,19] | −\|\|− |
| LCG(21,3) | [17,19] | −\|\|− |
| LCG(30,3) | [27,29] | −\|\|− |
| LCG(31,3) | [27,29] | −\|\|− |
| LCG(30,10) | 21 | −\|\|− |
| LCG(30,20) | 11 | −\|\|− |
| LCG(30,25) | [5,7] | −\|\|− |
| LCG(30,26) | 5 | −\|\|− |
| LCG(30,27) | [3,5] | −\|\|− |
| LCG(30,28) | 3 | −\|\|− |
| LCG(30,29) | [1,3] | −\|\|− |
| LCG(30,30) | 1 | −\|\|− |
| LCG(30,31) | 1 | −\|\|− |
| LCG(30,100) | 1 | −\|\|− |
| LCG(30,101) | 1 | −\|\|− |
| LCG(31,28) | 3 | −\|\|− |
| LCG(31,29) | [1,3] | −\|\|− |
| LCG(31,30) | 1 | −\|\|− |
| LCG(31,31) | 1 | −\|\|− |
| LCG(31,100) | 1 | −\|\|− |
| LCG(31,101) | 1 | −\|\|− |
| LCG(30,1) | 29 | −\|\|− |

### B.2.4 Bertrand competition

Table B.10: BC test results for ARM and IARM

| Bertrand competition | ARM | IARM |
|---|---|---|
| BC(200,100) | 100 | −\|\|− |
| BC(100,100) | 50 | −\|\|− |
| BC(300,100) | 150 | −\|\|− |
| BC(199,100) | 100 | −\|\|− |
| BC(201,100) | 101 | −\|\|− |
| BC(200,75) | 100 | −\|\|− |
| BC(200,50) | 100 | −\|\|− |
| BC(200,2) | 100 | −\|\|− |
| BC(199,75) | 100 | −\|\|− |
| BC(201,75) | 101 | −\|\|− |
| BC(200,1) | 100 | −\|\|− |
| BC(201,1) | [100,101] | −\|\|− |

### B.2.5 Nash Bargaining Game

Table B.11: NB test results for ARM and IARM

| Nash Bargaining | ARM | IARM |
|---|---|---|
| NB(50) | [25,26] | −\|\|− |
| NB(100) | [50,51] | −\|\|− |
| NB(101) | 51 | −\|\|− |
| NB(200) | [100,101] | −\|\|− |

## B.3 Test - Iterations of the algorithms

Table B.12: The results and iterations used by IRM and corresponding results for ARM

| Game | Iterations | IRM | ARM |
|---|---|---|---|
| TD(100,2) | 3 | 97 | [96,97] |
| TD(100,20) | 3 | 61 | [60,61] |
| TD(100,49) | 2 | 3 | 2 |
| ECG(25,1) | 3 | 23 | 23 |
| ECG(30,1) | 2 | 29 | 29 |
| ECG(30,6) | 2 | 29 | 29 |
| ECG(25,6) | 2 | 23 | 23 |
| LCG(20,2) | 2 | 19 | 19 |
| LCG(21,2) | 3 | 19 | 19 |
| LCG(30,20) | 3 | 11 | 11 |
| LCG(30,25) | 2 | [5,...,29] | [5,7] |
| BC(200,100) | 3 | 100 | 100 |
| BC(199,100) | 2 | 100 | 100 |
| NB(100) | 3 | 50 | [50,51] |
| NB(101) | 2 | 51 | 51 |

## B.4 Test - Randomness impact

Table B.13: The deviasion from the expected collected utility when increasing the number of iterations

| | 1 | 10 | 100 | 1000 | 10000 | 100000 |
|---|---|---|---|---|---|---|
| Minimum average utility | 4 | 33.0 | 47.95 | 50.79 | 52.53 | 52.77 |
| Maximum average utility | 101 | 73.9 | 56.97 | 54.81 | 53.42 | 53.19 |
| Expected average utility | 52.98 | 52.98 | 52.98 | 52.98 | 52.98 | 52.98 |
| Maximum deviasion | 100% | 39.49% | 9.49% | 4.13% | 0.85% | 0.40% |

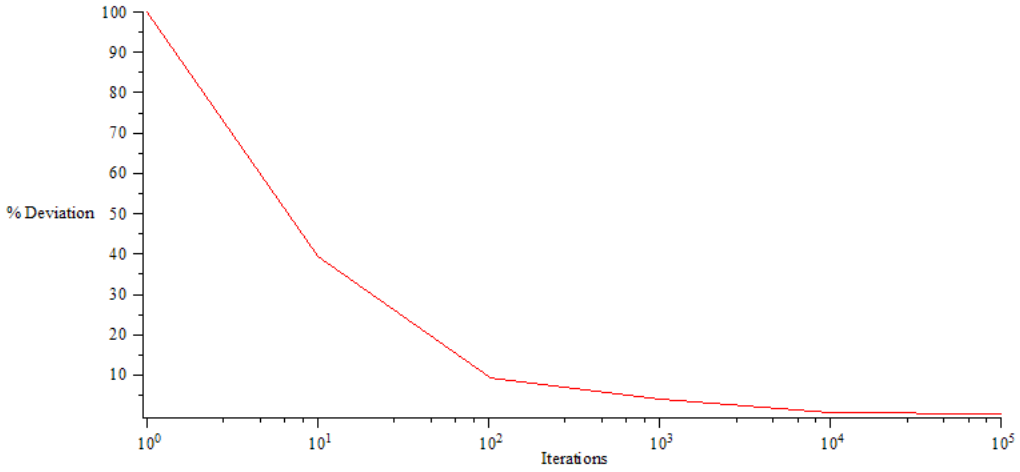


Figure B.1: Deviation from the mean.

## B.5 Test - Versus random opponent

Table B.14: The average outcome when playing against an uniformly random opponent.

| Game | NE | RM | IRM | ARM |
|---|---|---|---|---|
| TD(100,2) | $3.98, 0.02$ | $48.84, 52.64$ | $49.13, 52.85$ | $49.08, 52.76$ |
| TD(100,20) | $21.80, -17.80$ | $36.54, 59.84$ | $39.14, 47.09$ | $38.98, 46.74$ |
| ECG(15,1) | $5.00, 2.00$ | $3101.87, 4460.26$ | $3110.54, 2726.98$ | $3130.25, 2728.10$ |
| ECG(25,1) | $5.00, 2.00$ | $18.56 \cdot 10^5, 26.53 \cdot 10^5$ | $18.65 \cdot 10^5, 16.25 \cdot 10^5$ | $18.68 \cdot 10^5, 16.28 \cdot 10^5$ |
| ECG(30,1) | $5.00, 2.00$ | $95.22 \cdot 10^6, 83.24 \cdot 10^6$ | $94.88 \cdot 10^6, 82.46 \cdot 10^6$ | $95.43 \cdot 10^6, 83.31 \cdot 10^6$ |
| ECG(30,6) | $5.00, 0.00$ | $73.23 \cdot 10^6, 48.57 \cdot 10^6$ | $72.24 \cdot 10^6, 48.83 \cdot 10^6$ | $72.44 \cdot 10^6, 48.84 \cdot 10^6$ |
| ECG(25,6) | $5.00, 0.00$ | $13.96 \cdot 10^5, 95.10 \cdot 10^4$ | $14.03 \cdot 10^5, 95.19 \cdot 10^4$ | $14.03 \cdot 10^5, 94.72 \cdot 10^4$ |
| LCG(20,2) | $4.00, 2.00$ | $12.09, 13.70$ | $12.08, 13.68$ | $12.09, 13.69$ |
| LCG(21,2) | $4.00, 2.00$ | $12.03, 13.83$ | $12.10, 13.70$ | $12.11, 13.71$ |
| LCG(31,10) | $12.00, 2.00$ | $18.00, 24.70$ | $18.68, 21.99$ | $18.62, 21.98$ |
| LCG(30,20) | $22.00, 2.00$ | $21.80, 27.11$ | $23.70, 16.98$ | $23.62, 17.03$ |
| LCG(30,25) | $27.00, 2.00$ | $24.48, 26.13$ | $24.37, 26.20$ | $27.42, 10.71$ |
| BC(200,100) | $99.72, 41.65$ | $4978.97, 2518.41$ | $4990.70, 2482.28$ | $4994.65, 2482.97$ |
| NB(100) | $16.74, 17.22$ | $25.32, 12.37$ | $25.23, 12.60$ | $25.29, 12.42$ |

Table B.15: The average outcome when playing two uniform random players againg each other.

| Game | Random uniform vs Random Uniform |
|---|---|
| TD(100,2) | $34.39, 34.39$ |
| TD(100,20) | $34.41, 34.34$ |
| ECG(25,1) | $38.52 \cdot 10^4, 48.72 \cdot 10^4$ |
| ECG(30,1) | $10.62 \cdot 10^6, 8.49 \cdot 10^6$ |
| ECG(30,6) | $82.59 \cdot 10^5, 44.39 \cdot 10^5$ |
| ECG(25,6) | $19.74 \cdot 10^4, 39.64 \cdot 10^4$ |
| LCG(20,2) | $9.24, 9.04$ |
| LCG(21,2) | $9.49, 9.49$ |
| LCG(30,20) | $22.12, 20.81$ |
| LCG(30,25) | $24.75, 23.22$ |
| BC(200,100) | $3336.95, 3305.92$ |
| NB(100) | $16.81, 16.83$ |

Table B.16: The average outcome when playing against an random opponent which tends to play high bids.

| Game | NE | RM | IRM | ARM |
|---|---|---|---|---|
| TD(100,2) | $4.00, 0.00$ | $65.31, 68.91$ | $65.41, 68.86$ | $65.52, 68.90$ |
| TD(100,20) | $22.00, -18.00$ | $56.69, 67.94$ | $59.16, 48.30$ | $59.09, 47.74$ |

Table B.17: The average outcome when playing against an random opponent who tends to play strategies in the middle.

| Game | NE | RM | IRM | ARM |
|---|---|---|---|---|
| BC(200,100) | $99.98, 0.019$ | $4953.04, 3383.75$ | $5015.56, 3311.31$ | $4993.35, 3322.82$ |
| NB(100) | $14.63, 21.28$ | $25.33, 16.55$ | $25.45, 17.01$ | $25.06, 16.40$ |

## B.6 Test - Strategies versus strategies

The average outcome when playing the different solution concepts against each other.

Table B.18: The average outcome of ECG(15,1) with different chances $p$ to continue.

| Game | NE | RM | IRM | ARM |
|---|---|---|---|---|
| $p = 0.99$ | $5.0, 2.0$ | $15473.07, 19326.48$ | $15487.42, 7823.98$ | $15468.73, 7814.143$ |
| $p = 0.75$ | $5.0, 2.0$ | $3261.49, 4360.39$ | $3304.89, 2202.40$ | $3288.75, 2198.84$ |
| $p = 0.50$ | $5.0, 2.0$ | $368.51, 550.78$ | $384.61, 378.64$ | $380.75, 377.79$ |
| $p = 0.25$ | $5.0, 2.0$ | $22.43, 43.15$ | $20.15, 37.62$ | $21.58, 39.00$ |
| $p = 0.10$ | $5.0, 2.0$ | $5.93, 12.86$ | $6.09, 13.18$ | $6.04, 13.08$ |

Table B.19: The average outcome of LCG(31,10) with different chances $p$ to continue.

| Game | NE | RM | IRM | ARM |
|---|---|---|---|---|
| $p = 0.99$ | $12.0, 2.0$ | $32.71, 27.93$ | $30.04, 21.99$ | $30.09, 21.99$ |
| $p = 0.75$ | $12.0, 2.0$ | $9.06, 18.49$ | $9.17, 18.05$ | $9.19, 18.06$ |
| $p = 0.50$ | $12.0, 2.0$ | $5.01, 15.01$ | $5.01, 14.99$ | $5.00, 14.98$ |
| $p = 0.25$ | $12.0, 2.0$ | $3.67, 13.67$ | $3.66, 13.66$ | $3.67, 13.67$ |
| $p = 0.10$ | $12.0, 2.0$ | $3.22, 13.22$ | $3.22, 13.22$ | $3.22, 13.22$ |

Table B.20: TD(100,2)

| TD(100,20) | NE | RM | IRM | ARM |
|---|---|---|---|---|
| NE | $2, 2$ | $22.00, -18.00$ | $22.00, -18.00$ | $22.00, -18.00$ |
| RM | $-18.00, 22.00$ | $73.21, 73.10$ | $42.46, 79.49$ | $41.46, 79.52$ |
| IRM | $-18.00, 22.00$ | $79.49, 42.46$ | $61.00, 61.00$ | $50.52, 70.48$ |
| ARM | $-18.00, 22.00$ | $79.54, 41.43$ | $70.50, 50.50$ | $60.30, 60.20$ |

Table B.21: ECG(25,1)

| ECG(25,1) | NE | RM | IRM | ARM |
|---|---|---|---|---|
| NE | $5.0, 2.0$ | $5.0, 2.0$ | $5.0, 2.0$ | $5.0, 2.0$ |
| RM | $4.0, 9.0$ | $16.78 \cdot 10^6, 20.98 \cdot 10^6$ | $16.78 \cdot 10^6, 20.99 \cdot 10^6$ | $16.78 \cdot 10^6, 20.98 \cdot 10^6$ |
| IRM | $4.0, 9.0$ | $16.78 \cdot 10^6, 83.89 \cdot 10^5$ | $16.78 \cdot 10^6, 83.89 \cdot 10^5$ | $16.78 \cdot 10^6, 83.89 \cdot 10^5$ |
| ARM | $4.0, 9.0$ | $16.78 \cdot 10^6, 83.89 \cdot 10^5$ | $16.78 \cdot 10^6, 83.89 \cdot 10^5$ | $16.78 \cdot 10^6, 83.89 \cdot 10^5$ |

Table B.22: ECG(30,1)

| ECG(30,1) | NE | RM | IRM | ARM |
|---|---|---|---|---|
| NE | $5.0, 2.0$ | $5.0, 2.0$ | $5.0, 2.0$ | $5.0, 2.0$ |
| RM | $4.0, 9.0$ | $66.97 \cdot 10^7, 53.69 \cdot 10^7$ | $26.84 \cdot 10^7, 53.69 \cdot 10^7$ | $26.84 \cdot 10^7, 53.69 \cdot 10^7$ |
| IRM | $4.0, 9.0$ | $67.22 \cdot 10^7, 53.69 \cdot 10^7$ | $26.84 \cdot 10^7, 53.69 \cdot 10^7$ | $26.84 \cdot 10^7, 53.69 \cdot 10^7$ |
| ARM | $4.0, 9.0$ | $67.15 \cdot 10^7, 53.69 \cdot 10^7$ | $26.84 \cdot 10^7, 53.69 \cdot 10^7$ | $26.84 \cdot 10^7, 53.69 \cdot 10^7$ |

Table B.23: LCG(21,2)

| LCG(21,2) | NE | RM | IRM | ARM |
|---|---|---|---|---|
| NE | $4.0, 2.0$ | $4.0, 2.0$ | $4.0, 2.0$ | $4.0, 2.0$ |
| RM | $3.0, 5.0$ | $21.50, 21.50$ | $21.50, 21.50$ | $21.50, 21.50$ |
| IRM | $3.0, 5.0$ | $22.0, 20.0$ | $22.0, 20.0$ | $22.0, 20.0$ |
| ARM | $3.0, 5.0$ | $22.0, 20.0$ | $22.0, 20.0$ | $22.0, 20.0$ |

Table B.24: LCG(30,20)

| LCG(30,20) | NE | RM | IRM | ARM |
|---|---|---|---|---|
| NE | 22.0, 2.0 | 22.0, 2.0 | 22.0, 2.0 | 22.0, 2.0 |
| RM | 3.0, 23.0 | 27.50, 27.50 | 11.0, 31.0 | 11.0, 31.0 |
| IRM | 3.0, 23.0 | 30.07, 13.75 | 11.0, 31.0 | 11.0, 31.0 |
| ARM | 3.0, 23.0 | 30.09, 13.72 | 11.0, 31.0 | 11.0, 31.0 |

Table B.25: LCG(30,25)

| LCG(30,25) | NE | RM | IRM | ARM |
|---|---|---|---|---|
| NE | 27.0, 2.0 | 27.0, 2.0 | 27.0, 2.0 | 27.0, 2.0 |
| RM | 3.0, 28.0 | 25.96, 26.11 | 26.00, 26.00 | 6.90, 30.04 |
| IRM | 3.0, 28.0 | 25.92, 26.04 | 26.07, 26.00 | 6.93, 30.00 |
| ARM | 3.0, 28.0 | 29.13, 9.51 | 29.14, 9.51 | 12.00, 24.50 |

Table B.26: BC(200,100)

| BC(200,100) | NE | RM | IRM | ARM |
|---|---|---|---|---|
| NE | 27.92, 27.63 | 100.1, 0.0 | 100.08, 0.0 | 100.116, 0.0 |
| RM | 0.0, 99.94 | 5025.05, 4996.19 | 2503.64, 7486.58 | 2503.99, 7486.22 |
| IRM | 0.0, 100.11 | 7489.21, 2500.99 | 4994.15, 4994.15 | 4994.15, 4994.15 |
| ARM | 0.0, 99.77 | 7497.34, 2492.86 | 4994.15, 4994.15 | 4994.15, 4994.15 |

Table B.27: NB(100)

| NB(100) | NE | RM | IRM | ARM |
|---|---|---|---|---|
| NE | 16.85, 16.91 | 12.34, 25.62 | 12.36, 25.41 | 12.23, 25.47 |
| RM | 25.48, 12.24 | 12.52, 12.52 | 24.97, 24.97 | 12.46, 12.46 |
| IRM | 25.60, 12.53 | 25.05, 25.05 | 50.0, 50.0 | 24.95, 24.95 |
| ARM | 25.56, 12.25 | 12.59, 12.59 | 24.95, 24.95 | 12.55, 12.55 |

# Appendix C

# File format sample file

```
2
U
2 2
S C
S C
4,4 1,5
5,1 2,2
```