Aalto University
School of Science
Degree Programme of Engineering Physics and Mathematics

Markus Kärki

Computing pure-strategy punishments in repeated games

Master´s thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology in the Degree Programme in Engineering Physics and Mathematics.

Espoo, December 16, 2015

Supervisor: Professor Harri Ehtamo
Instructor: Dr. Kimmo Berg

Aalto University School of Science	ABSTRACT OF THE MASTER'S THESIS

Author: Markus Kärki

Title: Computing pure-strategy punihsments in repeated games

Title in Finnish: Rangaistusten laskeminen puhtailla strategioilla toistetuissa peleissä

Degree Programme: Degree Programme in Engineering Physics and Mathematics

Major subject: Systems and operations research	Minor subject: Finance

Chair (code): Mat-2

Supervisor: Prof. Harri Ehtamo	Instructor: Dr. Kimmo Berg

Abstract:

A repeated game is a decision making situation where players interact over and over again. They are used to model long-term relationships, cooperation and competition in a rational manner.

In repeated interaction a basic solution concept is subgame-perfect equilibrium. It a special case of Nash equilibrium and requires players to play Nash equilibrium in all possible situation during the game. Often many subgame perfect equilibria exist and computing those are as complex as in single-shot games, where players meet and choose an action only once [Borgs et. Al, 2010].

Worst equilibria allows a computationally efficient way to check, if an arbitrary solution is an equilibrium solution of a game [Abreu, 1986]. Berg and Kitti [2011] have introduced a method for computing all the equilibrium solutions of a game, assuming that the worst equilibrium payoffs are known. This is because rational players can only play an equilibrium solution. The worst equilibrium is the strongest punishment, which the players can use for threaten each other and force others to play another equilibrium.

It is an open question, whether it is possible to solve these punishment strategies in practice. This work examines computing threat points and strategies first with unlimited and then with bounded rationality meaning bounded computing capacity. An algorithm for optimal punishments with pure strategies is introduced. It is suitable for an arbitrary number of players and strategies and it accepts also unequal discount factors. In the end the performance of the algorithm is analyzed and limits to finding punishment strategies with given computing capacity is discussed.

Date: 16.12.2015	Language: English	Number of pages: 60

Keywords: Game theory, Repeated games, Subgame-perfect equilibria, Computational complexity, Algorithmic game theory

Aalto-yliopisto Perustieteiden korkeakoulu	DIPLOMITYÖN TIIVISTELMÄ
Tekijä: Markus Kärki	
Työn nimi: Rangaistusten laskeminen puhtailla strategioilla toistetuissa peleissä	
Title in English: Computing pure-strategy punishments in repeated games	
Tutkinto-ohjelma: Teknillisen fysiikan ja matematiikan tutkinto-ohjelma	
Pääaine: Systeemi- ja operaatiotutkimus	Sivuaine: Rahoitus
Opetusyksikön (ent. professuuri) koodi: Mat-2	
Työn valvoja: Prof. Harri Ehtamo	Työn ohjaaja(t): TkT Kimmo Berg

Tiivistelmä:

Toistetulla pelillä tarkoitetaan päätöksentekotilannetta, jossa samat pelaajat kohtaavat toisensa yhä uudelleen. Toistettuja pelejä käytetään, jotta voidaan mallintaa rationaalisella tavalla pitkäaikaisia vuorovaikutussuhteita ja niissä tapahtuvaa kilpailua ja yhteistyötä.

Toistetussa vuorovaikutuksessa vakiintunut ratkaisukäsite on osapelitäydellinen tasapaino, joka vaatii pelaajaa toimimaan Nashin tasapainoperiaatteen mukaisesti kaikissa tilanteissa. Usein toistetussa pelissä tällaisia tasapainoja on useita ja niiden laskeminen on yhtä vaikeaa kuin kertapelissäkin, jossa pelaajat kohtaavat päätöksentekotilanteen vain kerran [Borgs et. Al, 2010].

On todistettu, että pelaajien kannalta huonoimpien tasapainoratkaisujen tunteminen mahdollistaa muiden tasapainoehdokkaiden tarkistamisen laskennallisesti tehokkaasti [Abreu, 1986]. Lisäksi Berg ja Kitti [2011] ovat kehittäneet laskentamenetelmän, jolla pelin tasapainoratkaisut voidaan löytää, mikäli huonoimmat tasapainoratkaisut tunnetaan. Tämä perustuu siihen, että rationaaliset pelaajat voivat päätyä ainoastaan tasapainoratkaisuun ja huonoin tasapainoratkaisu on voimakkain uhka, jota pelaajat voivat käyttää painostuskeinona pysyäkseen muissa tasapainoratkaisuissa.

On kuitenkin avoin kysymys, pystytäänkö uhkausstrategioita käytännössä ratkaisemaan. Tässä työssä tarkastellaan uhkausstrategioiden laskemista ensin rajoittamattoman ja sitten rajoitetutun rationaalisuuden, eli käytännössä rajoitutetun laskentakapasiteetin näkökulmasta. Työssä esitellään algoritmi, jolla voidaan laskea voimakkaimmat puhtailla strategioilla aikaansaadut uhat annetulle kertapelille ja tarkastellaan sitä, millaisia uhkausstrategioita voidaan löytää annetun laskentakapasiteetin rajoissa.

Päivämäärä: 16.12.2015	Kieli: Englanti	Sivumäärä: 60

Avainsanat: Peliteoria, Toistetut pelit, Osapelitäydellinen tasapaino, Laskennan kompleksisuus, Laskennallinen peliteoria

Preface

When I started this thesis project, I had absolutely no idea, how challenging and long this project would eventually be. I started with game theory and optimization and ended up in the theory of computation and algorithms. During this time I have had exciting time while I lived a year in Munich, sit a backseat of a glider aircraft teaching, organized an international internship-program, Stratos, and so on. I would like to thank my supervisor and instructor for patience and flexibility during this long project.

Supervisor of the work, Professor Harri Ehtamo, pushed me towards to the goal and gave supporting guidance. Instructor Kimmo Berg for entered passionately into all questions related to work and gave advice full of deep expertise. I would like to thank them. In addition, I would like to thank people in Laboratory of System Analysis for interesting discussions and fun time in work and in spare time.

Friends and family I would like to thank for support, but also for making a good balance between work and rest. Active student life in PIK, Remburssi, Akaflieg Munich, Instituutti did not speed up my work with the thesis. They made the time memorable experience, which I look back with warmth in future.

Contents

1 Introduction

1.1 Background and motivation

Repeated games are a common model for long-term competition and cooperation. They are widely used in modeling economic interactions, for example competition between firms [Abreu, 1986]. This study focuses on games with infinite horizon and perfect information. The repeated games explain the observed behavior better than finite games, because they predict more cooperation [Mailath and Samuelson, 2006]. The model can also be interpreted as uncertain horizon. It is a suitable model, if the players do not know when the game is going to end, but there is a constant probability for ending after every round. The applications can also be found in computer science, where games models of systems, logics, automata or computational complexities [Björklund, 2005].

The theory of repeated games builds on a seminal work of Abreu (see Abreu [1988], Abreu [1986] and Abreu et al. [1990]). He showed that all the equilibrium behavior is reduced to simple strategies, which comprise only the initial strategy and extremal punishments for each player. Without this simplification ensuring if a strategy is an equilibrium solution, would be an infinite task. Abreu has shown that all equilibrium outcomes are possible to represent using only *simple strategies* and extremal punishments for each player [Abreu, 1988]. The extremal credible punishment has an important role in the computation of other equilibrium solutions.

Abreu's theory builds a framework for computing the equilibrium solutions of repeated games, but does not solve the problem directly. The numerical computation of equilibrium payoffs is a much studied problem. There are several algorithms for this problem including the work of Cronshaw [1997], Judd et al. [2003], Salcedo and Sultanum [2012], Burkov and Chaib-draa [2010] and Abreu and Sannikov [2013]. All of these assume correlated strategies, which simplifies the computation. These studies just compute the payoff and they do not discuss strategies behind these solutions. Berg and Kitti [2012] presented an algorithm to compute all the pure-strategy equilibrium solutions, both payoffs and corresponding game plays. The extremal punishments are assumed to be known and Berg [2013] presents a simple algorithm for computing the extremal equilibrium solutions.

Regardless of the fundamental role of extremal equilibrium payoffs, there is no systematic analysis or algorithms to find the extremal payoffs of repeated

games. This study answers this problem and presents an algorithm to find extremal punishments. The earlier research considers only individual cases [Abreu, 1986] or the different models like the game with imperfect monitoring Gossner and Hörner [2010].

Payoffs are discounted so the payoffs far in the future have a smaller weights in average payoff. The discount factor has a major effect to equilibrium solutions. The discount factor comes often from interest rate, but it is also commonly interpreted as subjective patience of a player. The minimax payoff and the Nash equilibria of the stage game set bounds to extremal punishment, but there has not been a way to find the exact minimum in the general case. The problem turns out to be an infinite horizon combinatorial optimization problem. There are no general algorithms for this kind of problem, so the presented branch-and-bound algorithm is quite pioneering in this sense.

1.2 Research objectives

The aim of this study is to present a systematic analysis of minimal equilibrium payoffs of the repeated games. The existence of equilibrium solution is guaranteed [Abreu, 1986], but there is no proof for the computability of the solution for general game. The extremal equilibrium outcomes may be extremely complex [Nachbar and Zame, 1996]. This study introduces a branch-and-bound algorithm to compute the minimal equilibrium payoffs and strategies. The algorithm finds only pure strategies, which are possible to present in the finite form. Together with Berg and Kitti [2012] work there is now an algorithm for computing the pure-strategy equilibriums for arbitrary N-player stage game with possible unequal discount factors. Although there is no good theoretical performance bound for the algorithm, but excessive numerical testing well indicates typical performance.

1.3 Structure of work

The structure of the work is the following. First, in Section 2, there is a review to the literature and theory of repeated games. Section 3 focuses on the extremal equilibria in different games and divides the games into the classes by the extremal equilibrium. The existence and computability of the solution is also covered in this section. Sometimes the extremal equilibrium outcomes are complex and it is not possible to solve these analytically. Section 4 describes an algorithm for these hard cases. The limitations and performance

of the algorithm are analyzed in Section 5 with the examples of a Cournot game and a simple matrix game called anti-No Conflict. The study ends up in the conclusions in Section 6.

2 Repeated games

A game is a decision making situation where the player chooses one of his alternative actions and these choices determine the players' payoffs. The choices of the players are called *actions*. The game is called single-shot game or stage game if there is just one decision and no repeated interaction. The classical example of single-shot game is the game called prisoner's dilemma (PD), which is presented in a matrix form in following way:

	L	R
T	$3,3$	$0,4$
B	$4,0$	$1,1$

One player chooses the row, and another chooses the column simultaneously, without knowing the choices of the other players. The first number in the matrix is the payoff for player 1 and the second is player's payoff. The dilemma is that player 1 gets a better payoff by choosing B and the player 2 by choosing R regardless the other player's action. The joint optimum solution (T,L) is dominated by other strategies and thus never reached by rational players. PD is widely used to explain behavior in decision problems such as the tragedy of commons, arm races [Majeski, 1984] and pollution.

Repeated game means that the stage game is played finitely or infinitely many times consecutively. Every stage game can be seen as a start of new infinite game and these games are called *subgames*. The sequence of stage game outcomes is called a path. A game with infinite horizon is a suitable model for many everyday decisions because often relationships last long time or may not have a predetermined end. Players discount their future payoff by their discount factors δ_i, $i = \{1, .., n\}$, $\delta_i \in (0, 1)$. This is a common assumption in economic models and mathematically necessary for keeping payoffs bounded. The discount factor δ can be also interpreted as probability that the interaction continues. This study concentrates on games with *perfect monitoring* meaning that a player can observe the other's actions between every stage game.

A strategy of a player is a complete plan how to choose an action after an arbitrary game history. If the strategy defines a certain action for every

situation, it is called *pure strategy*. Alternatively, a strategy may only describe the probabilities of different actions. This kind of strategy is known as *mixed strategy*. Third and the most general class of strategy are *correlated* strategies. The player observes a signal and players can therefore coordinate their actions by those signals. Mixed and correlated strategies are a common assumption in the studies of the field. The main weakness of these strategies is that players are assumed to observe not only the past outcomes, but also the strategies of each other.

Each player has one strategy and the *Nash equilibrium* is a strategy combination, where no player has an incentive to change his strategy assuming other players does not change their strategies. There is one pure Nash equilibrium in the previous Prisoner's dilemma example: (B,R). If the game is repeated, we also consider that there are no profitable deviations at any point of the infinitely long game. If this holds, the strategy is said to be *subgame perfect*. Although, the stage game and its equilibria may be simple, the repeated version of the game may provide countless number of equilibrium solutions. That is because players can punish and reward each other by their choices.

This section starts with an introduction and a mathematical formulation of repeated games. Then the basic solution concept, Nash equilibrium, is explained and generalized to repeated games. The end of the section reviews *simple strategy*-approach and numerical methods for computing equilibria.

2.1 Stage game in normal form

Stage game G is presented in normal form

$$G = \{N, A, u\} \tag{1}$$

where N=$\{1,...,n\}$ is the set of players, who choose their actions simultaneously from their available actions $A_i, i \in N$. The sets A_i are assumed to be finite. The action profiles are $A = A_1 \times ... \times A_n$ and an action profile leads to a payoff vector $u : A \to R^n$, which contains a payoff for each player. The payoffs are often written in a matrix form. The minimax is the payoff, which the player can ensure although all other players would try to minimize his payoff. A player has no incentive to accept any payoff lower than minimax. The minimax payoff in the prisoner's dilemma is 1 for both players. It is thus the absolute lower bound for any equilibrium solution, if it is assumed that the players can observe what the other player has done [Gossner and Hörner,

2010]. The minimax pure-strategy payoff of stage game is formally defined

$$\underline{v_i} = \min_{a_{-i}} \max_{a_i} u_i(a_i, a_{-i}). \tag{2}$$

2.2 Formulation of repeated game

A repeated game consists on stage games, which are played successively. The formal definition is

$$G^{\infty}(\delta), \tag{3}$$

where G is a stage game and δ is the discount factor which is needed to compute the payoffs.

The players observe and remember the past actions, which are called the history of the game. The history is denoted $A^k = \times_k A$, where k is the length of the history. When the game begins the history is empty: $A^0 = \{\emptyset\}$. Each player has a strategy $\sigma_i : A^k \rightarrow A_i, k = 0, 1, 2...$ The strategy tells the player how to choose an action after the current history. Note that the strategy must define action for arbitrary history, not only for the history that is realized, if the strategy is followed. This kind of strategy requires *perfect monitoring* ergo players must perceive the previous action profiles. The previous definition of strategy restricts on pure strategies because history lead to the exact action not the probability distribution over actions. [Mailath and Samuelson, 2006, page 19].

The strategy profile is $\sigma = \{\sigma_1, ..., \sigma_n\}$ and it determines the outcome of the game unambiguously. Players discount their future payoffs by a constant player specific discount factor $\delta_i \in (0, 1)$. The interpretation of the discount factor is that the player keep recent payoffs more valuable than those which are realized far in the future. The average discounted payoff is defined

$$U_i(\sigma) = (1 - \delta_i) \sum_{k=0}^{\infty} \delta_i^k u_i(\sigma(A_k)) \tag{4}$$

Average payoff lies always in the convex hull of action profile payoffs. This convex hull is called as the set of feasible payoffs and a point in the hull is a *feasible payoff*. $U_i(\sigma)$ denotes the average payoff from finitely or infinitely length path and $u_i(a)$ is the payoff from a single action profile a. The sum is scaled by $(1-\delta_i)$ to make repeated game payoff correspond to the stage-game payoffs. Otherwise, the discount factor would affect the range of possible average payoffs. The action profiles of a 2x2 matrix game are denoted by the following table.

	L	R
T	a	b
B	c	d

This allows a simple way for writing outcomes. If the strategy tells to repeat action profile d, which is the Nash equilibrium of the single shot PD, the outcome path is denoted d^∞. Alternatively, if action profile a is played two times and then the players alter between c and b the outcome path is $aa(cb)^\infty$. The outcome paths plays important role in further analysis. The following notation is used. The outcome action profile in subgame t is denoted $q(t)(\sigma)$ and the infinite stream of action profiles is denoted $Q(\sigma)$ or just Q when there is no risk of confusion.

2.2.1 The Nash equilibrium

The Nash equilibrium (NE) is the most fundamental solution concept in game theory. The strategy profile σ^* is a Nash equilibrium of the game if for all players i and strategies σ

$$U_i(\sigma^*) \geq U_i(\sigma_i, \sigma^*_{-i}), \tag{5}$$

where the strategies of players, excluding player i, is denoted by $\sigma_{-i} = \{\sigma_1, ..., \sigma_{i-1}, \sigma_{i+1}, ..., \sigma_n\}$. The condition (5) means that no player can improve her payoff by changing her strategy alone. There may be many Nash equilibria in the same game. On the other hand, it is possible that there are no pure strategy equilibria at all. Nash [1951] proved in his famous theorem, that every finite stage game has at least one Nash equilibrium if mixed strategies are considered. Computing Nash equilibria for a game is a widely studied and fundamental problem. The task is hard for a general stage game, and for more advanced model such as repeated or stochastic games the task may be cumbersome.

To keep notation in the next chapters compact, the set of all strategy profiles is denoted $\Sigma = \Sigma_1 \times ... \times \Sigma_n$, where Σ_i is the set of possible strategies for player i. In addition, the set of all equilibrium strategies is Σ_p and naturally $\Sigma_p \subset \Sigma$. The notations of this section follows the notation used by Abreu.

The payoff from the best deviation is denoted

$$v_i^*(a) = \max_{a_i} u_i(a_i, a_{-i}) \tag{6}$$

If a is a Nash equilibrium,

$$v_i^*(a) = u_i(a) \tag{7}$$

for all players $i \in N$.

2.3 Subgame-perfect equilibrium

Nash equilibrium is not a sufficient condition to ensure equilibrium behavior in repeated games. Although the players has no incentive to deviate from the strategy at the first round, there may be profitable options to deviate later. The proper solution concept is *subgame perfect equilibrium* (SPE). The strategy profile σ' is a subgame-perfect equilibrium if for all histories $\mathcal{A} = \bigcup_{k=0}^{\infty} A^k$, σ' is a Nash equilibrium of the subgame. $\mathcal{A}$ contains also the histories with are never realized, if the strategy σ' is followed. [Mailath and Samuelson, 2006, page 23].

The intuition behind SPE is clear, but the definition does not help much when looking for solution for an infinitely repeated game. Checking whether σ is an equilibrium strategy involves of comparing it with the infinite number of strategies because a player can deviate in any subgame or combination of subgames. The simple strategy approach solves this problem ingeniously.

2.4 Equilibrium with simple strategies

The question, whether a strategy profile σ_0 is a subgame perfect equilibrium or not, is very nontrivial. The strategy profile induces a path Q_0. One has to check not only whether there are profitable deviations in initial path Q_0, but also all possible combinations of deviations. The task is cumbersome, but the seminal work of Abreu simplifies this analysis of equilibrium solutions. He proved that the worst perfect equilibria exist for each player and all equilibrium behavior is supported by threat of this worst punishment. Abreu reduced his examination by introducing simple strategies [Abreu, 1988].

The definition of simple strategy $\sigma(Q_0, Q_1, ..., Q_n)$ is following: The strategy consists of the set of paths, where Q_0 is an outcome path, which the players follow initially. If player i deviates from the strategy, then the other players punish him by Q_i. If the player deviates from his punishment path, the punishment starts over. If a punishing player deviates, the others start punish him. The only exception is that the deviations are not punished if at least 2 players deviate simultaneously. In this case, players just continue on the original path. Abreu restricts only to games with Nash equilibrium in one-shot game. The assumption is only technical to ensure the existence of SPE in the repeated game. It is no restrictive assumption: If a SPE exists, we can apply the theory anyway. It does not matter if there is not a single-shot Nash equilibrium.

A simple strategy is a subgame-perfect equilibrium if and only if it satisfies condition

$$U_j(Q_i^t) \geq (1-\delta_j)v_j^*(Q_i^t) + \delta_j U_j(Q_j) \ \forall i,j \in N, \forall t \in \mathbb{N} \tag{8}$$

The complete proof is presented in Abreu [1988]. However, the idea is simple. For every player it shall be optimal to follow the path Q_i at every time step rather than deviate in one round and get the punishment after that. The condition is sufficient and also necessary because otherwise a player can gain profit by deviating from the simple strategy. The practical consequence is, that it is enough to check that there are no profitable one-shot deviations for any player. One-shot deviation is a strategy where a player deviates once from his supposed action, but follows strategy again after that.

Next, we define an *optimal penal code*. The optimal penal code is a vector of strategy profiles $(\sigma_1^-, ..., \sigma_n^-)$, such that for all i $\sigma_i^- \in \Sigma_p$ and $v_i^-(\sigma_i^-) = \min\{U_i(\sigma) \| \sigma \in \Sigma_p\}$. A simple penal code is called *optimal simple penal code* if it is an optimal penal code. A suboptimal punishment $\{Q_1, ..., Q_n\}$ may discard path Q_0, which would be an equilibrium with the optimal punishments. Too small, infeasible punishment does not discard all the non-equilibrium solutions.

The *simple penal code* $(Q_1, ..., Q_n)$ consist of strategy profiles $(\sigma(Q_1, Q_1, ..., Q_n), ..., \sigma(Q_n, Q_1, ..., Q_n))$. The only difference between simple strategy profiles is the path, which is followed initially. A simple penal code is also the optimal simple penal code if Q_i is the optimal penal code. This means that optimal simple penal code is just a simple strategy, which produces the optimal punishment path.

The main lemma of Abreu [1988] proves that if a path Q_0 is a subgame-perfect equilibrium, then

$$(1-\delta_i)v_i^*(Q_0^t) \leq U_i(Q_0^t) - \delta_i v_i^- \tag{9}$$

The condition is again both, necessary and sufficient. This means, that we can easily check, if an arbitrary path Q, is possible to produce by an equilibrium strategy. All that is needed is payoffs from the optimal simple penal codes. The number of inequalities to check depends on the complexity of Q. The example path $aa(cd)^\infty$ needs just four checks for each player. However, nothing guarantees that the optimal simple punishments are possible to find and execute by a finite-time algorithm.

The punishment path σ_i is sometimes called penal code because it consists on a sequence of actions. Both proof are based on incentive compatibility

(IC) condition, which applies definition of Nash equilibrium (5) in repeated games.

$$(1-\delta_i)u_i(a^*) + \delta_i v \geq (1-\delta_i)u_i(a, a^*_{-i}) + \delta v_i^- \forall i \in N, a \in A, \qquad (10)$$

where δ_i is the discount factor of player i, a^* is an action chosen among the strategy and v is the continuation payoff if the strategy is followed forever. If v_i^- is the payoff from optimal punishment payoff, the condition is necessary and sufficient for all equilibrium paths.

2.5 Finite automata representation of a strategy

Strategies in a repeated game can be presented in a simple way by grouping histories into equivalence classes. A member of each class has an identical continuation strategy. This kind of strategy is called automata. It consists of a set of states W, an initial state w_0, an output function f and a transition function τ.

An output function $f : W \to A$ tells which action players should choose in each state. A transition function $\tau : A \times W \to W$ describes the state in each state after each action profile. This can be presented as a graph form.

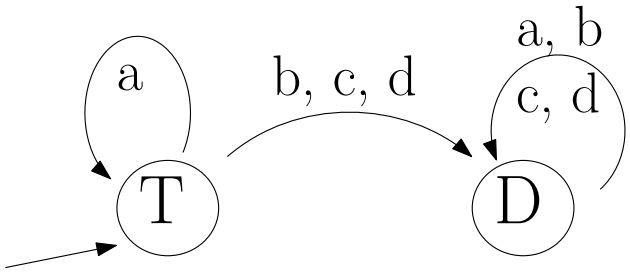

Figure 1: An automaton for playing trigger strategy in prisoner's dilemma

Figure 1 presents an automaton for the row players. The automaton implements well-known trigger strategy in prisoner's dilemma. Transfer function is presented by arcs and states by nodes. Initially, the player cooperates (T) and repeat this action if the outcome is action a. Otherwise, the automaton goes to another state, where the action D is played. There is no way out from the second state, so the punishment lasts for the rest of the game. [Mailath and Samuelson, 2006, page 29]

2.6 The graph representation of all equilibrium solutions

Berg and Kitti [2012] present a method for computing all equilibrium solutions and payoffs in a repeated game. The idea is based on Abreu [1988] framework, where a strategy is simplified to an outcome path and threat of the punishment payoff. Their algorithm needs punishment payoffs for each player as input, but provides no method for finding them. The number of different equilibrium paths is often infinite, but at least sometimes all equilibrium paths can be collected into one finite graph. Figure 2 presents an example of a graph computed using method of Berg and Kitti. Note that this graph does not directly tell the strategies behind these outcome paths. All we know is that any infinitely long outcome path, which is possible to construct by following the graph, results from at least one, but possibly many equilibrium strategies. It is always possible to produce an outcome path of an equilibrium play via simple strategies. It is done by using the outcome path as an initial path and then punishing deviating players by optimal simple punishment paths. In some cases, we can numerically compute a graph, which contains all the equilibrium outcomes of a game with a given discount factor. That is remarkable achievement when the number of equilibrium paths is infinite.

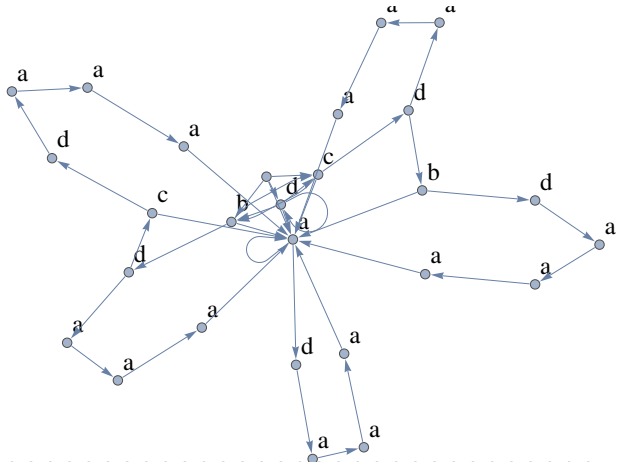

Figure 2: The graph of all equilibrium paths in PD with $\delta = 0.51$

The novel idea behind the algorithm is doing tree search for possible outcome paths and finding *elementary paths*, which produce all the equilibrium solutions. The computation starts by dividing search tree branches into the

first action feasible (FAF) and first action infeasible (FAI). The search stops if all the open branches are classified. The definition of the first action feasible path p is

$$\begin{array}{rl} con(p) \leq con(f(p)), & \text{if length of p} \geq 2 \\ con(p) \leq v^{-}, & \text{if length of p} = 1 \end{array} \tag{11}$$

and for infeasible paths

$$con(p) > \overline{v} \tag{12}$$

where $con_i(p)$ is the continuation payoff after the path p. It means, that it is the lowest average payoff, which player i can accept after playing path p, without taking account possible payoff demands coming from the subpaths of p. Moreover, $f(p)$ refers to the last action of the path p. The continuation payoff is the smallest value of v (Inequality 10), which is large enough to satisfy the condition.

$$con_i(a) = \frac{(1-\delta_i)}{\delta_i}((u_i(a^*, a^*_{-i}) - u_i(a)) + v_i^- \tag{13}$$

Recursion can be used to compute continuation payoff of a path longer than one action

$$con_i(p) = \delta_i^{-1}(con_i(p^{k-1}) - (1-\delta_i)u_i(a)). \tag{14}$$

If a path cannot be sorted to FAF or FAI, it is called neutral and the children of this path are examined. The target is that all the branches can be sorted to FAF or FAI category. It is possible that a FAF path contains FAI paths as subpaths. Paths that do not contain infeasible parts are called *elementary paths*. In practice, infeasible paths are pruned out during a novel graph constructing process.

Continuation payoffs are used to cut branches in the algorithm. In this work the definition is slightly different, so that also feasibility of subpaths of p is checked while the continuation payoff is computed.

2.6.1 Measuring the equilibrium set

When the equilibrium paths are known, the corresponding equilibrium payoffs can be easily computed. The *payoffs set* consist of all equilibrium payoffs of the game. Pure strategies produce a set of discrete points. Mixed- and correlated strategies produce continuous sets, because the realized outcome is random. The number of different equilibrium paths and payoffs is often infinite although the graph may be finite. Payoff sets are typically fractals. The set are constructed as self-affine set, which explains fractal nature. In

practice the fractal nature is easy to understand when the equilibrium paths are presented as a graph. In a graph it is possible to go through the same nodes over and over again. Figure 3 presents an approximation of equilibrium payoffs in PD.

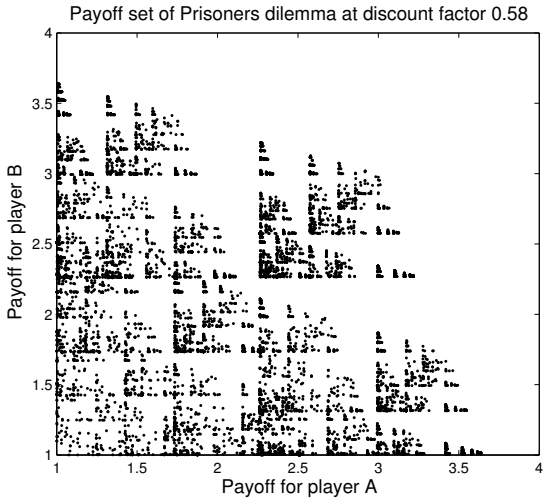

Figure 3: The fractal set of equilibrium payoffs in PD with $\delta = 0.58$

The fractal payoff sets were discovered first by Berg and Kitti [2014]. The fractal form offers a clever way to measure "the number of the equilibrium" payoffs or the density of the set. The general measure used for fractal density is *Hausdorff dimension*, which generalizes the concept of dimension to real numbers. The Hausdorff dimension is defined

$$\dim_H(V) = \inf\{\beta \geq 0 : \mathcal{H}_\beta(V) = 0\}, \tag{15}$$

where

$$\mathcal{H}_\beta(V) = \lim_{\epsilon \to 0} \inf\{\sum_{i=1}^{\infty} \|W_i\|^\beta : V \subset \bigcup_{i=1}^{\infty} \|W_i\| \leq \epsilon \text{ for all } i\} \tag{16}$$

Note that the Hausdorff dimension of a finite set is zero. Hausdorff dimension can be computed from a graph by using the formula [Mauldin and Williams, 1988].

$$\dim_H(V) = -\log \rho(D) / \log \delta \tag{17}$$

where $\rho(D)$ is the largest eigenvalue of the graphs adjacency matrix. The Hausdorff dimension can be possible computed exactly if the players have equal discount factor, $\delta \leq 1/2$. With larger discount factors, the payoffs may overlap and the formula (17) should be considered as an upper bound. Extensions to estimating bounds for fractal dimension in overlapping case exists (see Das and Ngai [2004] and Edgar and Golds [1999]).

2.7 Folk theorems in repeated games

Folk theorems are a class of important theoretical results related to repeated games. The main result was known in game theoretical community some time before they were proved or published, and that is why they are called Folk theorems. The main claim is that any feasible payoff point is attainable by an equilibrium strategy assuming a discount factor sufficiently close to one. The result is well studied nowadays and there is a bunch of proofs for different game classes. The first proofs consider on games with no discounting at all (Friedman [1971]). The theory was completed by Fudenberg and Maskin [1986], who extended the proof for discounted games assuming the games satisfies the *full dimensionality condition*. Abreu et al. [1994] and Wen [1994] introduced less restrictive conditions for the folk theorem. The first is called NEU-condition, and the latter introduces the concept of an effective minimax payoff.

Full dimensionality, NEU and an effective minimax are all sufficient conditions of Folk theorem in repeated games. Full dimensionality is the tightest one. A game is full dimensional if and only if the dimension of feasible payoffs is the same as the number of players. See Section 3.2.4 to find a common example of non-full dimensional game called matching pennies. NEU, Nonequivalent utilities, is less restrictive and often necessary condition for Folk theorem. It demands that players' utilities are not equivalent in all action profiles nor equivalent after a linear transform. Formally, there is no $c, d \in \mathbb{R}$ (d>0) such that

$$u_i(a) = c + du_j(a) \ \forall a \in A \tag{18}$$

Effective minimax is defined

$$u_i(w^i) = \min_a \max_{j \in A_s} \max_{a_j} u_i(a_j, a_{-j}) \tag{19}$$

Effective minimax is an essential concept especially, if some players have equivalent utilities, but there are at least two subsets of players with different utilities.

The practical interpretation of Folk theorem is that in society, a tribe or other social systems every feasible outcome, even those, which seem to be harmful for everyone, is possibly a result from the rational decision making of individuals. The only condition is that the outcome is better than leaving the society (minimax action).

The original form says only that every single payoff point is achieved some discount factor, which possibly is different for different points. The latest results spread the idea by proving, that there are games, where all the feasible payoffs are attainable with every discount factor assuming the factor is high enough (Berg and Karki [2013], Stahl [1991]).

Although the folk theorem ensures the existence of a countless number of equilibrium strategies, finding or playing one is not necessarily an easy task [Borgs et al., 2010]. As we will see in the next chapter, computing an equilibrium strategies is generally a hard task. This is also the case with repeated games and subgame perfect equilibria. Folk theorems builds on the worst credible punishment, which sometimes called also a threat point. Because finding this point is a complex task, the equilibrium payoff set for players with bounded rationality and computing capacity differs from the payoff set for perfectly rational players.

2.8 Computational complexity of Nash and subgame-perfect equilibria

Computing subgame-perfect equilibria is generally a hard task. The relevant questions are, whether the solution exists and is it computable at all, if we allow only pure strategies. If the answer is positive for both, we still need an efficient algorithm for computation. An efficient way to compute optimal punishment is also the key to computing the other equilibria. If it turns out that there is no way to compute equilibria in efficient manner and bounded computing capacity, it is also sets any practical implications under suspect.

Finding equilibrium in one-shot game is PPAD-complex (Daskalakis et al. [2009]). See Table 1 for descriptions of complexity classes, which are discussed in this section. The solution exists by Nash [1951], but is not necessarily easy to find. Even in special cases such 2 player m×m game (Chen and Deng [2006]) and computing an ϵ-equilibrium are PPAD-complex (Chen et al. [2006]).

There are only few special class of games, which are known to be com-

Complexity class	Name	Description
NP	Non-deterministic polynomial time	Given solution can be verified as a solution in polynomial time
NP-hard	Non-deterministic polynomial time -hard	Problems in this class are at least as hard as the hardest problems in NP
NP-complete	Non-deterministic polynomial time -complete	Class of problems which contains the hardest problems in NP
P	Polynomial	Class of problems, which can be solved in polynomial time
PPAD	Polynomial Parity Arguments on Directed graphs	A complexity class closely related to NP, but solution is known to exist
PLS	Polynomial Local Search	A complexity class closely related to NP, but solution is known to exist

Table 1: Description of complexity classes

putable in polynomial time. One is anonymous game, where approximate Nash equilibrium is computable in polynomial time (Daskalakis et al. [2007]) and another is symmetric congestion game (Daskalakis et al. [2009]), where even pure Nash equilibrium is polynomially computable. General congestion game belongs to PLS-class.

Repeated games are generally not easier than one-shot games. Borgs et al. [2010]] proves that finding Nash equilibria of (k + 1)-player infinitely repeated games is as hard as finding Nash equilibria of k-player one-shot games. They also prove that computing the threat point of the game is an NP-hard task. In their setting in mixed strategies the threat equals minimax payoff.

Note that this work focuses on pure subgame-perfect equilibria. However, computing SPE must be at least as hard as NE, because every SPE is always also NE. Finding a threat point with pure strategies seems to be a computationally exhaustive task, with no guaranteed fast convergence in the worst case. Anyway, problems of this kind can be tackled with an efficient heuristic with good performance in a typical case.

2.9 Numerical methods for computing equilibria

Since the fixed-point characterization of equilibrium was published by Abreu [1986, 1988], a wide range of algorithms for computing equilibria has been published. Most of the algorithms assume correlated strategies, which makes the payoff set convex. The algorithm for computing payoff set was published by Abreu et al. [1990] and it is known as APS algorithm. One well-known implementation applies linear programming to solve the problem (Judd et al. [2003]). This algorithm was upgraded later (Abreu and Sannikov [2013]) and this upgraded algorithm is implemented and compared to our algorithm later in this work.

APS algorithm starts from the set of all feasible payoffs W_0 then computes iteratively sets $W_1, W_2, .., W_{n+1} = B(W_n)$ until convergence ($W_{n+1} = W_n$). The set-valued operator B is defined

$$B(W) = co\{v | \exists w \in W, a \in A \text{ s.t. } v = (1-\delta)v^*(a)+\delta w \text{ and } \delta(w-P(W)) \geq (1-\delta)h(a)\} \tag{20}$$

where

$$P_i(W) = \min\{w_i | w \in W \text{ for some } w\} \tag{21}$$

and

$$h(a) = v^*(a) - u(a). \tag{22}$$

The operator is monotonic and the computation algorithms implement and approximate this operator. The main restriction is that they can only handle games with two players and equal discount factors.

The set of pure-strategy equilibrium payoffs differs significantly from the set of correlated strategies. Where the pure strategies produce a set of discrete payoffs, the correlated strategies produce areas with no gaps or holes. The reason is that the correlated strategies produce a random path, with the infinite number of possibilities. Correlated strategies not only fill the gaps between pure strategies, but also new equilibria appear. The extremal equilibria may also be different with pure and correlated strategies.

Berg and Kitti [2011] introduced an algorithm for computing all the pure equilibrium strategies. Having all the strategies represented as a graph makes it possible to plot equilibrium payoffs [Berg and Kitti, 2014]. The payoff sets are fractals and hard to compute, if only pure strategies are allowed [Berg and Kitti, 2014]. Their algorithm takes punishment payoffs as an input, but finding those payoffs remains open question. Algorithm of Berg and Kitti works for an arbitrary game matrix with arbitrary many players and unequal discount factors are possible. This is a rare feature among the numerical

algorithm for computing equilibria. The disadvantage is that computing times may explode when the size of the game or discount factor increases.

3 Minimal equilibria

3.1 Definition and motivation

The focus of this section is finding the optimal penal codes for repeated games. The punishment paths and payoffs are player specific, which makes search somewhat more complicated. The optimal punishment means that it produces the worst equilibrium payoff for a player. One could also state that optimal punishment is the mildest punishment, which makes deviations unprofitable. This is an important aspect in games with imperfect monitoring when deviations are unobservable and player never knows for sure if the other player has deviated. If deviations are observable, there is no need to really execute any punishment, so long they are credible. The whole set of equilibrium is possible to construct from the threat of the worst punishment as seen in Section 2.4.

3.2 Infinitely repeated games

The absolute lower bound for all equilibrium payoffs is the stage games minimax payoff. The intuition behind this is that player can just choose his minimax action and receive the minimax as an average payoff. If a strategy gives a smaller payoff than the minimax payoff, the deviation is profitable and the strategy cannot be an equilibrium. Whether minimax payoff is possible to get by an equilibrium strategy, depends strongly on the discount factors. The smallest equilibrium payoff lies always between the minimax payoff and stage game Nash equilibrium payoff. Folk theorems are the major theorems about equilibria in repeated games. The theorem proves that with certain conditions all the feasible payoffs, that weakly dominates the minimax, is possible to realize as a Nash equilibrium. See Friedman [1971] for classic form and Fudenberg and Maskin [1986] for games with discounting. It assumed that players have not equivalent utilities in all actions profiles (Abreu et al. [1994]). Otherwise the minimax payoff may be impossible to reach.

In the next sections the games are divided into four categories and examine the condition, when the minimax payoff is reached. Algorithm in Section 4

focuses especially to cases where the minimax payoff is not an equilibrium in stage-game and solutions may be extremely complex. These games are found in categories B and C.

Figure 4 illustrates relationships between categories. The main point is dividing games to B and C categories. The games in A and D are just trivial cases inside of these main categories.

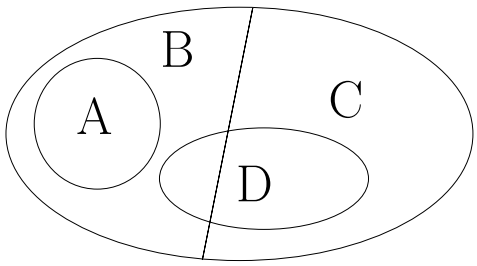

Figure 4: Relationships between categories

3.2.1 Category A

Definition of the category: Minimax is a Nash equilibrium

The stage game has a Nash equilibrium giving the minimax payoff for player i. There fore the minimax payoff is the punishment and repeating the minimax action is the punishment path for player i. The Nash equilibrium of stage game is also equilibrium in repeated game as we see straight from IC. By definition $v_i^*(a_N) = u_i(a_N) \forall i$, if $a_N \in A$ is a Nash equilibrium of the stage game. Then

$$v_i^*(a_N) = u_i(a_N) \geq \frac{\delta_i}{1-\delta_i}(U_i(a_N) - v_i^-) \tag{23}$$

The path a_N^∞ can be used as a punishment and substitute $v_i^- = U_i(a_N)$. Thus the condition (23) is reduced to trivial inequality (24)

$$0 \geq 0, \tag{24}$$

which is trivially true. A well-known example of this class is the prisoner's dilemma, which equilibrium solution (B,R) leads to the minimax payoff (1,1).

3.2.2 Category B

Definition of the category: $u_j(a_{\text{minimax},i}) \geq u_j(a_{\text{minimax},j}) \forall i, j \in N$

The stage game's minimax action profile gives any other player greater payoff than her minimax. Repeating minimax actions is a subgame perfect equilibrium if players are sufficiently patient. If minimax action is not unique, it is sufficient to find one action for each player so that condition is satisfied.

An example is a discrete form of Cournot game, which is used an example in Section 5.1. The minimax action profile for player i is denoted $\underline{a}_i$. The incentive condition (10) can be written in form

$$v_i^*(a_j) - u_i(a_j) \geq \delta(U_i(\underline{a}_j) - \underline{v}_i + v_i^*(a_j) - u_i(a_j)) \tag{25}$$

The definition of this class by this, new notation is

$$U_i(\underline{a}_j) \geq \underline{v}_i \forall j. \tag{26}$$

The left side of equation (25) is surely non-negative. The stationary repeating of minimax actions satisfies IC always if

$$\delta_i \geq \frac{v_i^*(a_j) - u_i(a_j)}{v_i^*(a_j) - u_i(a_j) + U_i(\underline{q}_j) - \underline{v}_i} \forall i \times j, i \neq j \tag{27}$$

Low discount factors may also allow the minimax payoff although the condition (27) does not hold. Anyway, this happens only with some individual discount factors, not their neighborhood, if path p is non-stationary, meaning not repeating a single action. Payoff $U_i(p)$ depends on the discount factor (see equation (4)). Generally, low discount factors may lead to very nonstationary penal codes.

3.2.3 Category C

Definition of the category: $u_j(a_{\text{minimax},i}) < u_j(a_{\text{minimax},j})$ for some $i, j \in N$

The stage game's minimax action profile gives some other player smaller or equivalent payoff than her minimax payoff. The punishment payoff is always greater than minimax payoff and punishment paths tend to be very complex. An example is a three-player game

C

	L	*R*
T	1,1,1	0,0,0
B	0,0,0	0,0,0

D

	L	*R*
T	0,0,0	0,0,0
B	0,0,0	1,1,1

With equal discount factors the only equilibrium is repeating action with the payoff (1,1,1), but with an unequal discount factor the worst equilibrium payoffs may be close to minimax. With discount factors (0.15, 0.65, 0.95) the lower bounds are close to zero (10^{-12}, 10^{-3}) for players 1 and 2 and 0.49 for player 3. The best-found punishment paths are $d^{14}a^{\infty}$ (for 1 and 2) and $(ba)^{\infty}$ (for 3) by the the algorithm. No player accepts zero payoff, but all payoffs above that are reachable. Note that, because the utilities are equivalent for all player, the minimax payoffs are never reached [Wen, 1994, Chen, 2008, Chen and Takahashi, 2012].

3.2.4 Category D

Definition of the category: No pure strategy equilibrium

There are some games where it is impossible to construct pure strategy subgame-perfect equilibrium. The well-known example is a game called matching pennies, which presented below.

	L	R
T	$-1,1$	$1,-1$
B	$1,-1$	$-1,1$

In this game minimax payoff is one the both players, which strictly dominates the set of feasible payoffs. However, it is possible that there are pure subgame-perfect equilibria, although the stage game has no pure equilibria. A simple example is following game

	L	M	R
T	$1,2$	$-1,3$	$0,0$
C	$0,0$	$-1,-1$	$3,-1$
B	$0,0$	$0,0$	$2,1$

Paths (a^{∞}, i^{∞}) can be used as simple penal code, if discount factor $\delta_i \geq 1/4$ $\forall i$. The punished players gets average payoff $U_1(a^{\infty}) = U_2(i^{\infty}) = 1$ and the punishing player gets payoff $U_2(a^{\infty}) = U_1(i^{\infty}) = 2$. Both payoff points ((1,2) and (2,1)) strictly dominates the minimax payoff (0,0).

Necessary and sufficient condition for pure-strategy equilibria is left outside of this work. Anyway, to find pure-strategy equilibria, there must be feasible payoffs, which weakly dominates minimax payoff. That is because minimax is the absolute lower bound for any equilibrium behavior with pure strategies. The sufficient condition is a pure equilibrium in a stage-game. The sufficient and necessary condition is still an open question.

3.3 Computability of the optimal punishment

An important question related to any numerical algorithm is whether the solution exists and is it possible to find by the algorithm. The existence of extremal equilibrium has been proved (Abreu [1986]), but it is not obvious if it is possible to find with any numerical algorithm. Nachbar and Zame [1996] stated that there are equilibrium strategies, which are not possible to implement by finite automata. Are there such simple strategies? Is it possible that extremal equilibria are of this type? Could such a strategy be a credible threat if players have no method to know about it? These questions are open, but we concentrate on strategies with finite complexity. At least we know that there are some games, where it is possible to represent all the equilibrium solutions as a finite directed graph [Berg and Kitti, 2011]. Unfortunately, there are no conditions for games where this finite representation is possible.

A common way to overcome the problems with infinitely complexity is adopt a model with bounded rationality [Simon, 1972, Rubinstein, 1986]. Kalai and Stanford [1988] claimed that every subgame-perfect equilibrium is possible to approximate by finite automata, if ϵ-equilibria are accepted. ϵ-equilibrium means that there may be profitable deviations, but the possible gain from deviation is always smaller than ϵ. The equilibrium condition 5 changes to form

$$U_i(\sigma') \geq U_i(\sigma_i, \sigma'_{-i}) - \epsilon, \forall i \tag{28}$$

Their theorem says, that for every equilibrium σ^* there is a strategy σ' with finite complexity, so that

$$|U_i(\sigma') - U_i(\sigma^*)| < \epsilon, \forall i \tag{29}$$

and σ' is a subgame-perfect ϵ-equilibrium. The reverse does not hold. Every ϵ-equilibrium is not close to subgame-perfect equilibrium, thus searching optimal strategy from the set of finite strategies may be fruitless in some games.

If it is possible to present all the equilibrium strategies in a graph, we can apply the dynamic programming argument to find out the form of punishment paths. The problem is thus finding the extremal discounted infinite path from the graph. An infinite path in a finite graph arrives sooner or later at a node, where it has visited before. At this point Bellman's principle says that, the optimal path must continue in the same way as the previous visit in the same node. We always know that the optimal path begins with some start part and then ends up in an infinite loop. This idea is exploited

in generation method of infinite paths in the algorithm. Madani et al. [2010] has examined an efficient algorithms for this kind of problem.

The algorithm of this work can be interpreted as algorithm for ϵ-equilibria. There is a small ϵ in inequalities to prevent numerical rounding problems. The selected epsilon is many decades smaller than precision needed to compute numerical examples in this work. It thus likely that found paths are also subgame-perfect equilibria, not only epsilon equilibria. On the other hand, it would be easy to modify the algorithm for searching epsilon equilibria with arbitrary epsilon.

3.4 Extremal equilibria in different games

Assumptions like perfect monitoring and pure strategies were made in the previous section. This section reviews literature about extremal equilibria in different games. The question about the worst punishment has not so fundamental role in games with the finite horizon, but extremal equilibria are a well-studied subject in several game classes. The optimal punishment strategies and payoffs vary much between the game classes.

3.4.1 Normal-form game

Finding equilibria in normal-form game is a long-standing problem. Computing the equilibria of a normal-form game is a computationally hard problem for arbitrary game (Daskalakis et al. [2009]), but for small games it is trivial, because we can just check every action profile separately. The extremal equilibrium is found by picking the smallest payoffs from the sets of equilibria.

3.4.2 Finitely repeated games

A finitely repeated game does not differ a much from a single-shot game if perfect information is assumed. Strategies can be enumerated, and the game is possible to present in the normal form. If there are multiple pure-strategy equilibria, the equilibrium behavior will not be trivial anymore. Benoit and Krishna [1985] makes a systematic search for the worst equilibria and proves a folk theorem for the game class. They also state that equilibria in finitely repeated games approach the infinite case if the game is sufficiently long.

3.4.3 Correlated and mixed strategies

Pure strategies are a special case in mixed strategies where probability of one action is one and others action are zero. Correlated strategies are yet wider generalization where players can alter their strategies by a public signal. If a player decide not to alter his strategy, his strategy belong to mixed or pure strategies.

Assuming players can synchronize their actions by a public correlation machine, also correlated strategies are possible. A correlated strategy is a function $s : R \times H \rightarrow \Delta(A)$. $R \in (0, 1)$ is a signal observed by a player and in certain cases all players can observe the same signal. The model has been popular, and it makes the payoff set convex. Plenty of studies present methods for computing the payoff set with correlated strategies. Judd et al. [2003] presents one algorithm for computing the payoff set. Abreu and Sannikov [2013] together with Salcedo and Sultanum [2012] extend this method. The algorithm gives inner and outer approximation to the set. The extremal points of the set are computed. The extremal equilibria payoffs may be lower than when restricting to just pure strategies.

Solving the equilibrium strategies with mixed strategies is still an open question. There are at least two difficulties. 1) The natural assumption is that the other players observe only the outcome, not the strategy. The other player cannot detect a deviation like with pure strategies. 2) One could use mixed strategies that are not equilibria in the stage game. This equivalent in pure strategies where an equilibrium path may contain actions that are not equilibria in a stage game. The problem is worse in mixed strategies because there is the infinite number of mixed strategies in each stage game.

3.4.4 Imperfect and private monitoring

There are two additional models for repeated interaction, where perfect monitoring is not assumed. The models are imperfect and private monitoring. The method for constructing equilibria differs much from method used under public monitoring. In these models, players do not perceive action, but only an action dependant random variable P. Abreu et al. [1990] presented a method for constructing equilibria of such game. Private monitoring means that player knows only his own actions. This is the case for example in price competition between firms when the firms know only their own prices. Gossner and Hörner [2010] proved that equilibrium payoffs may be even under the minimax payoff, which not possible in games with perfect monitoring.

4 The algorithm description

This section contains the detailed description of the algorithm. The algorithm is suitable for an arbitrary game with a finite number of players and action profiles. The discount factors may be different for each player. Finding optimal punishment is formulated as an integer programming problem and solved using a branch and bound algorithm. The formulation and general branch and bound algorithm are introduced in Section 4.1. The detailed description of the algorithm is found after that beginning from Section 4.2.

4.1 Formulation of the problem

Searching the optimal punishment is basically an optimization problem.The problem is formulated as an infinite horizon combinatorial problem and solved using a *branch and bound* -algorithm. The problem is formulated below.

$$\begin{array}{ll} \min_{\sigma} & \sum_{i=1}^{N} U_i(\sigma_i) \\ \text{s.t.} & (1-\delta_i)\Delta_i(\sigma_j^k) \leq \delta_i(U_i(\sigma_j^{k+1}, \delta_i) - U_i(\sigma_i, \delta_i)) \forall k \in \mathbb{N}, i, j \in N \end{array} \tag{30}$$

The solution, σ, includes one infinite punishment path for each player. Paths are searched in finite form, so that they consist on finite start and finite loop, which is repeated infinitely. An example path could be *aabaad* and the loop consist of n last actions. Thus the infinite path could be $aab(aad)^{\infty}$. The best found punishment gives an upper bound for the global minimum of the problem. The algorithm gives also lower bounds for the punishment payoffs. The lower bound for the payoff is the minimum of lower bounds of open branches.

4.1.1 General branch and bound algorithm

Branch and bound is a general algorithm for discrete optimization. It was described by Land and Doig [1960]. The idea behind the algorithm is enumerating all feasible solutions and do a tree-like search for them. The general steps of a branch and bound algorithm are following:

1. *A bounding function* Gives the lower bound for the best solution in a given subspace

2. *A strategy for selecting* the next subproblem
3. *A branching rule* subdividing the subspace if the subspace cannot be discarded

Infinite horizon does not change the basic principles of the algorithm, but needs to be taken account at stopping condition. A common condition is to stop, if the best found solution gives the objective function value equal to lower bound. In infinite version a solutions close enough to the lower bound (difference $< \epsilon$) stops the algorithm.

4.2 Description of the algorithm

The algorithm is based on generating all finite length start parts of paths and discarding the ones that either have profitable deviations or do not provide a small enough payoff. The idea of the algorithm is demonstrated in the next simple example. A 2x2 stage game has four possible actions profiles: a, b, c and d. The initial branches are {a,a}, {a,b}, {a,c},..., {d,d}, which contain the first action profiles for each player. If the branch {a,b} cannot be discarded its children are added to the of open branches (B). The children are {aa,ba}, {aa,bb},{aa,bc},...,{ad,bd}. The algorithm returns the bounds for the punishment payoffs and stops when the bounds go under the limit. The lower and upper bounds for the branch's payoff are needed also for pruning the set of open branches.

The algorithm finds a punishment path for each player, and they are denoted by $p = \{p1, p2, \ldots, pn\}$. Each branch b in the search tree contains the beginning of the punishment path for each player. For example, a branch $b = (ab, cc)$ in a two-player game means that $p1 = ab$, $p2 = cc$, player 1's punishment path starts with ab and player 2's starts with cc. The branching is done by adding one action profile for each player, so each node gets $4^2 = 16$ children in a game with two players and four action profiles. However, most of the branches can be cut soon, which often prevents the rapid growth of the search tree.

The initialization step creates all the one-length path combinations. The current lower and upper bounds are denoted by lb and ub, respectively. For a branch b, the lower and upper bounds are $lb(b)$ and $ub(b)$.

The lower bound is the sum of the payoff from branch and continuation payoff, which is computed by IC condition (10). The algorithm tries to complete the branch to an infinite path. If an equilibrium path is found, it is

used as an upper bound for the minimum equilibrium payoff of the branch. The infinite paths are generated by adding following loops: if the path is $abad$, we examine the paths $aba(d)^\infty$, $ab(ad)^\infty$, $a(bad)^\infty$ and $(abad)^\infty$.

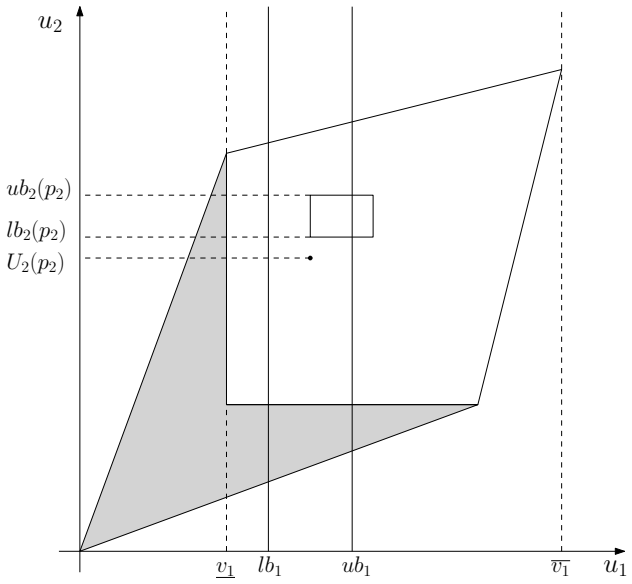

Figure 5: Difference between upper and lower bounds as function of discount factor.

The next example demonstrates branch cutting and stopping conditions. Let $\underline{v} = 1$, $\delta = 1/2$, $ub_2(b) = 2.5$ and $u(a) = 5$. The figure (4.2) visualizes the notation. There is an infinite equilibrium path with payoff 2.5 ($ub_2(b)$), which is also the upper bound of the punishment payoff. Now, we need not consider paths starting with action profile a since they give too high payoffs, i.e., the smallest payoff after starting with a and continuing with the minimax payoff is $(1 - \delta)5 + \delta 1 \geq 3.5$. This is an example of cutting the branch because of too large payoff. The cutting condition for non-feasible paths is different. If the path demands a continuation payoff, which is either impossibly large or so large that the paths payoff exceed the current upper bound, the path is cut off.

Assume that $u_2(c) = u_1(b) = 1$ equilibrium payoffs, which are got by paths $\sigma_1 = c^\infty$ and $\sigma_2 = b^\infty$. The payoff 1 is now the new upper bound. We can stop the algorithm since the minimum payoff cannot be below the minimax value. Thus, c^∞ is the punishment path for player 1 and 1 is the minimum payoff. The lower bound is the smallest lower bound in the set of branches or minimax, which is greater.

Algorithm 1 presents the main steps, which are explained more thoroughly in Sections 4.4-4.6. Step 2a consider the branch selection problem. Step 2b checks that the punished player has no incentive to deviate from the examined path. Step 2b computes the minimum continuation payoff requirements after the punishment path. It checks that the branch gives lower payoff than the current upper bound. Otherwise, the path cannot be the minimum. The step also checks that the continuation payoff requirements are not too high. If these conditions are satisfied, then the payoffs for different loops are computed and the bounds are updated. The algorithm stops when the bounds are within a given tolerance.

Input: Stage game in normal form, discount factors δ_i
Result: Bounds for $v^-(\delta)$, best found feasible path

```
1  Initialize the set of open branches B;
2  Set ub ← v̄ and lb ← v̲;
3  while ub_i − lb_i < ε, ∀i do
4      1) b ← pick the next branch from B;
5      2) for every player i do
6          for k = 1 → |pi| do
7              a) Check that the punished player does not deviate from pi at
               stage k;
8              if there is a deviation then Goto Step 1;
9              b) Compute the minimum continuation requirements after path
               pi for all players;
10             if the lowest payoff from the branch is higher than ub_i then Goto
               Step 1;
11             else if any continuation payoff requirement is too high then
               Goto Step 1;
12             c) Compute the players' payoffs for a path qk = pi^{k−1}(pi_{k−1})^∞;
13         end
14     end
15     3) Compute the bounds;
16     4) Update B;
17 end
```

Algorithm 1: Compute the minimum paths and payoffs

4.3 Step 1: Selecting the next branch

The heuristic for the next branch selection is one key issue in algorithm performance. This phase may also require significant amount of computing power. There is no method for guessing, which branches are the most potential punishments. There may be a huge number of local optima and totally different paths may produce payoffs of the same kind. The most important decision is the search method. The common alternatives are breadth-first search (BFS) and depth-first-search (DFS). In addition a heuristic or randomizing could be a good alternative.

Breadth-first-search is used in the examples of this work. The redeeming feature of this method is reliability, but it may not be the fastest search method. The typical problem of DFS and heuristics is that they may stick in one branch (and its sub-branches) and never reach the global optima. A suitable heuristic could still reduce the computing times dramatically.

4.4 Step 2a: Checking deviations

This step ensures that the punished player does not deviate from the examined path. For example, if the path starts with *abca* and the player could get better payoff by deviating from c to a, then the outcome would be $(aba)^\infty$. The punishment starts always over after the deviation and player deviates every time at time same point. The path starting with *abca* cannot be an equilibrium path, because deviating is profitable for the player. Let $M(k)$ (defined in formula (31)) be the payoff of the punished player if he always deviates from the kth action profile

$$M(k) = \frac{1-\delta_i}{1-\delta_i^k}\left[U_i(pi^{k-1}) + \delta_i^{k-1} v_i^*(f(pi_{k-1}))\right]. \tag{31}$$

The notation $f(pi_{k-1})$ refers to the first action of the path pi_{k-1}. The deviation is not profitable if $M(k)$ is smaller than or equal to the payoff that the player receives when path pi is followed. Now, if any $M(k) \geq ub_i$ then also the path pi gives a higher payoff than ub_i, and the branch cannot form the punishment path. Thus, if the following condition does not hold

$$ub_i \geq \frac{1-\delta_i}{1-\delta_i^k}\left[U_i(pi^{k-1}) + \delta_i^{k-1} v_i^*(f(pi_{k-1}))\right], \quad \forall k, \tag{32}$$

then the branch can be cut.

4.5 Step 2b: Computing minimum payoff requirements

Let con_i^j denote the minimum continuation payoff of player i that is required after the path pj. The v is solved from Inequality (10) and the smallest v satisfying the condition is

$$con_i^j = \max_{k=1,\ldots,|pj|} \left[(1-\delta_i)v_i^*(f(pj_{k-1})) + \delta_i lb_i(par(b)) - U_i(pj_{k-1})\right] / \delta_i^{|pj|-k+1}, \tag{33}$$

where $par(b)$ is the parent node of branch b and $|pj|$ is the length of the path. The index k goes through all the possible stages where the player can deviate, and the deviation is followed by the punishment payoff of the parent node $lb_i(par(b))$. Now, the lower bound of branch b is

$$lb_i(b) = (1-\delta_i)U_i(pi) + \delta_i^{|pi|} \max[con_i^i, lb_i], \tag{34}$$

where $U_i(pi)$ is the payoff of player i from path pi. The lower bound is reached, if there is an equilibrium code which payoff is exactly con_i^i. If the lower bound of the branch is higher than the global upper bound, i.e., $lb_i(b) \geq ub_i$, then the branch can be cut. Thus, the path should be examined only if

$$con_i^i < \delta_i^{-|pi|}(ub_i - (1-\delta_i)U_i(p_i)). \tag{35}$$

The computation of *con* makes possible to cut the infeasible branches. If a path requires too high continuation payoff that cannot be achieved in the game, then the branch can be cut. The path can be an equilibrium only if

$$con_i^j \leq \overline{v}_i, \quad \forall i, j. \tag{36}$$

If either of the conditions (35) and (36) does not hold, the branch is cut.

4.6 Step 3: Computing the bounds

The lower bound of b was computed in Step 2b. The upper bound of b can be computed with the paths $qk = pi^{k-1}(pi_{k-1})^\infty$ from Step 2c. Now, it is possible to check that these paths have none profitable deviations for any player. The order of path checking is important. The worst case is that all the combinations of loop lengths must be checked. It is best start from loops giving the smallest payoffs and check others only if needed.

The checking feasibility of path combination $q = \{q1, q2, ..., qn\}$ consists on following steps.

1. Payoffs from loops are computed in phase 2c in Algorithm 1
2. If $U_i(pj_{k-1}^{\infty}) < con_i^j$ for any $i \times j$, the loop k is discarded. However, the path is not necessarily feasible, because con_i^j is just a lower bound for demanded continuation payoff.
3. The loop with smallest payoff is chosen for every player and the feasibility of path combination is checked. The condition is

$$U_i(qj) \leq (1-\delta_i)v_i^*(qj^k) + \delta_i U_i(qi) \tag{37}$$

   The total number of checks is n^2k because every player's punishment must be feasible for everyone in every phase k.
4. If the path in Step 3 is feasible, the payoffs $U_i(qi)$ are used as punishment. Otherwise, all the loop combinations must be checked separately. This brute-force approach is needed seldom in games examined in this study.

4.7 Step 4: Updating B

If the upper bound $\sum_i ub_i(b) < \sum_i ub_i$, then we update $ub_i = ub_i(b)$, and remove the branches b' from B that satisfy condition

$$lb_i(b) \geq ub_i, \quad b \in B. \tag{38}$$

This phase is known as pruning in the branch-and-bound algorithm. The current branch b is removed from B in Step 1. Moreover, we add the children of b to B. The lower bounds of the parent is saved and used as a initial value of child's lower bound. We leave for future research how to examine only the relevant children and how to go through the branches in a good order. In this paper, we pick the branches in Step 1 in the first in last out order.

The lower bounds are updated by $lb_i = \min\ lb_i(b)$, $b \in B$. If $ub_i - lb_i < \epsilon$, for all i, the algorithm stops. If $ub_i \neq lb_i$, then the found punishment path for player i is feasible but may be suboptimal. We also terminate the algorithm if the length of the punishment path reaches a given limit.

5 Numerical examples

5.1 Duopoly game

The first example is the duopoly game from Abreu [1988]. The game is a matrix version of classic Cournot duopoly where the player decide the amount of production. The price and profit are determined by the markets. The actions and payoffs are found in following table:

	L	M	H
L	$10, 10$ (a)	$3, 15$ (b)	$0, 7$ (c)
M	$15, 3$ (d)	$7, 7$ (e)	$-4, 5$ (f)
H	$7, 0$ (g)	$5, -4$ (h)	$-15, -15$ (i)

The firms have three output levels: low (L), medium (M) and high (H). The nine action profiles are denoted by letters a to i, and the stage game Nash equilibrium is e, i.e., (M, M), giving payoff 7. The joint optimum (10,10) is not an equilibrium solution in the stage game. The minimax payoff is $\underline{v}_i = 0$, and thus for all discount factors it holds that $0 \leq v_i^-(\delta) \leq 7$, $i = 1, 2$.

The punishment paths and payoffs are computed with different discount factors. The found upper bounds are presented in Figure 6. It is seen that the minimum payoff is the stage game Nash equilibrium when the discount factors are low and close to the minimax value with high discount factors. The games between those and unequal discount factors leads to more complex solution and are computationally harder. The punishment paths are simple for both low and high discount factor values. The stage game Nash equilibrium is repeated and the punishment path is e^∞ when the discount factor is low, and the minimax is repeated (c^∞, b^∞), when the discount factor is high. The punishment paths are more complicated for discount factors between 0.1 and 0.5, even though the minimum payoff is close to zero. For example, the best found paths for $(\delta_1, \delta_2) = (0.4, 0.8)$ are $p1 = c^\infty$ and $p2 = hhhdadgd^\infty$ giving payoffs 0 and $1.5 \cdot 10^{-4}$, which may be suboptimal.

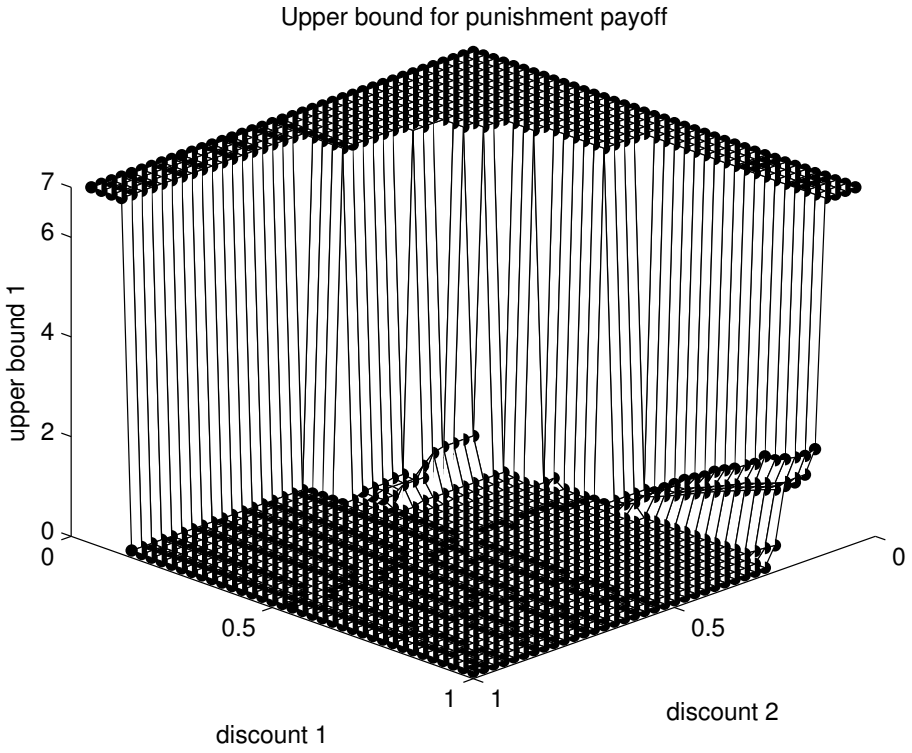

Figure 6: The upper bounds for the minimum payoffs in duopoly.

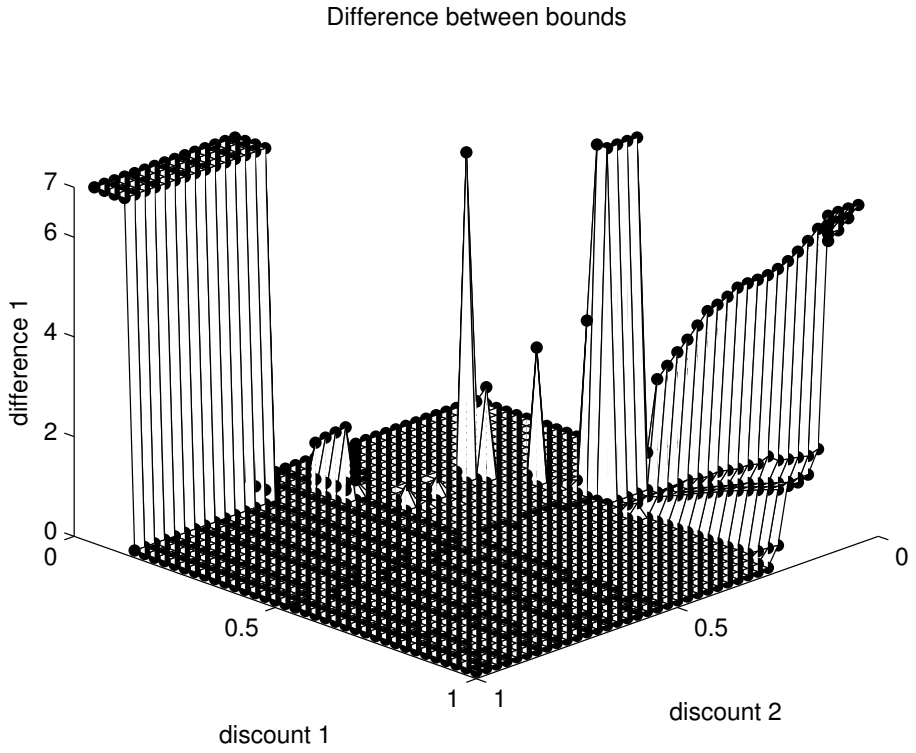

Figure 7: The difference between the bounds in duopoly.

The difference between the upper and lower bounds are shown in Figure 7. We can see that the algorithm converges in most cases but it has problems when one player has a small discount factor and the other has a high value. In these cases, the algorithm has to go through a lot of paths that have no profitable deviations. Finally, the convergence of the bounds as a function of the path length is demonstrated in Figure 8 for $(\delta_1, \delta_2) = (0.4, 0.8)$. We can see that the differences are small already for short path lengths and they converge faster for the player with a smaller discount factor value. This is a general phenomenon since the punishment paths are typically shorter. The reason is that only the first payoffs matter when the discount factor is small.

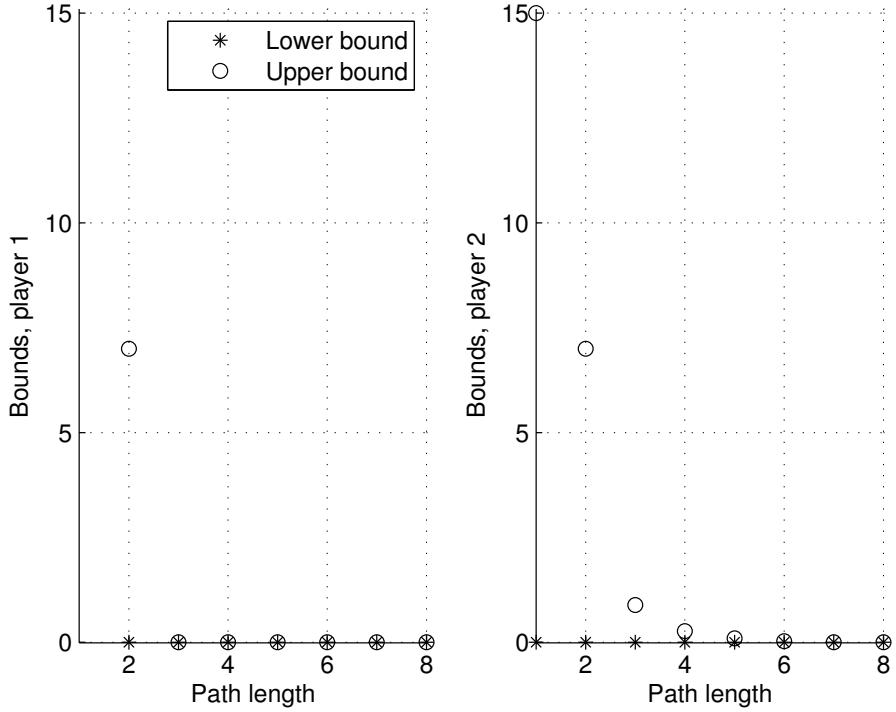

Figure 8: Development of the bounds when $(\delta_1, \delta_2) = (0.4, 0.8)$.

5.2 Anti-No Conflict Game

The second example called anti-no conflict game is good example of a game with complex punishment paths. The game matrix is following:

	L	R
T	$5, 5$	$4, 3$
B	$3, 4$	$2, 2$

The game has a dominant Nash equilibrium (T, L). The minimax value is 4, but the corresponding action profiles (T, R) and (B, L) cannot be played repeatedly since they give a smaller payoff than 4 to the punishing player. The upper bounds and the differences between the bounds are shown in Figures 9 and 10.

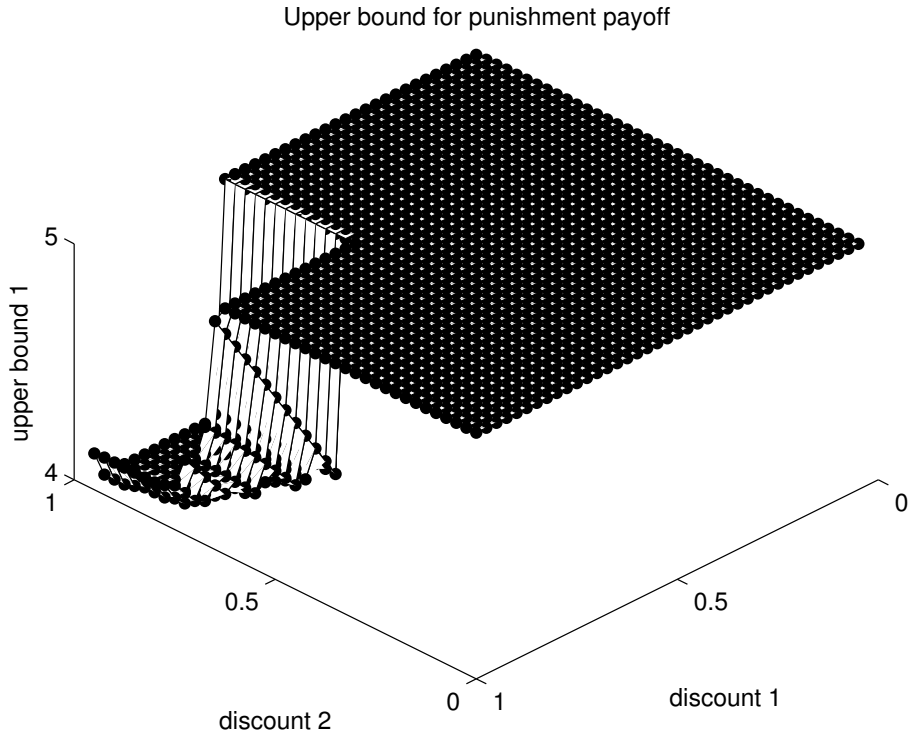

Figure 9: The upper bounds for the minimum payoffs in anti-no conflict game.

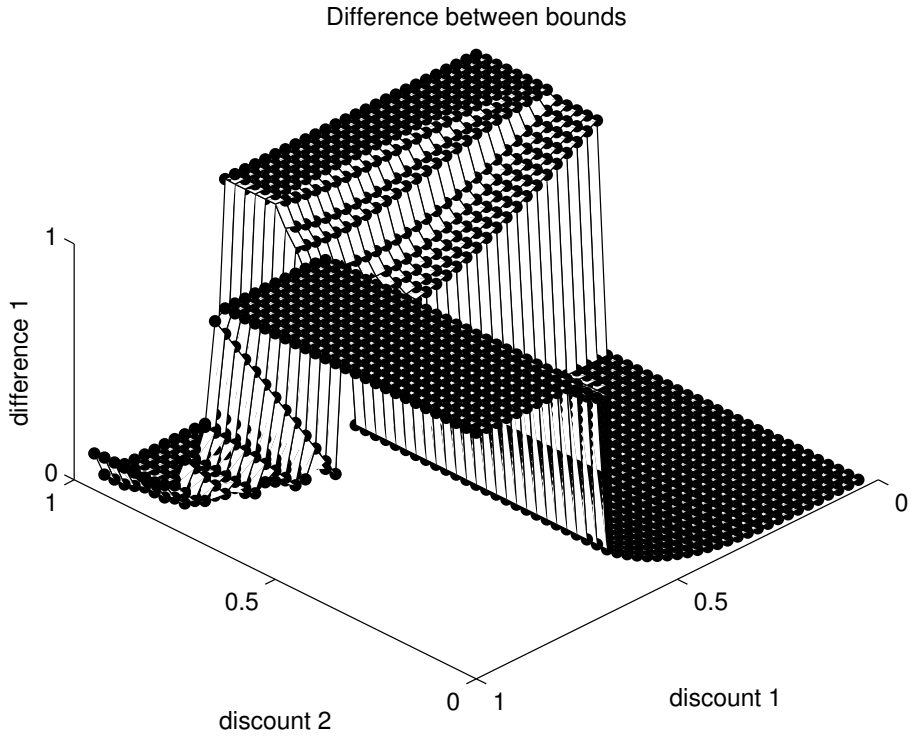

Figure 10: The difference between the bounds in anti-no conflict game.

We see from the figures that it is possible to punish the other player if the discount factors are high enough. The minimum never reaches the minimax value, except in some separate discount factor combinations. The punishment paths are long and complicated for high discount factor values. For example, it is possible to punish with a path $daac(baaaadaa)^\infty$, when $\delta = 0.8$. The payoff is approx. $2.9 \cdot 10^{-7}$ over the minimax value and it may still be suboptimal. It should be noted that it may be possible to reach a low payoff with a simple path. For example, the path dca^∞ gives already a low payoff of 4.08. Moreover, the algorithm has problems with convergence when one discount factor is low and the other one is high.

5.3 Comparison to Abreu&Sannikov-algorithm

Abreu and Sannikov have introduced an algorithm for computing the equilibrium payoff set with correlated strategies. Their work is discussed in Section 2.9. Payoff sets in pure strategies are computationally hard to solve. The

lower bounds of their payoffs are the minimal equilibrium payoffs in the game. In this section we compare lower bounds in anti-no conflict game. Also, the performance of algorithms is reviewed although the output is very different.

5.3.1 Difference between lower bounds

Pure strategies are a subset of correlated strategies. It is obvious that the equilibrium set with correlated strategies includes the set with pure strategies. However, the threat point differs often. As we see in Figure 11 the threat in anti-no conflict game really is lower with correlated strategies. With low discount factors, the cooperation is the only equilibrium of the game with high discount factor it is possible to reach minimax payoff. Between those there is area where the threat point is between those.

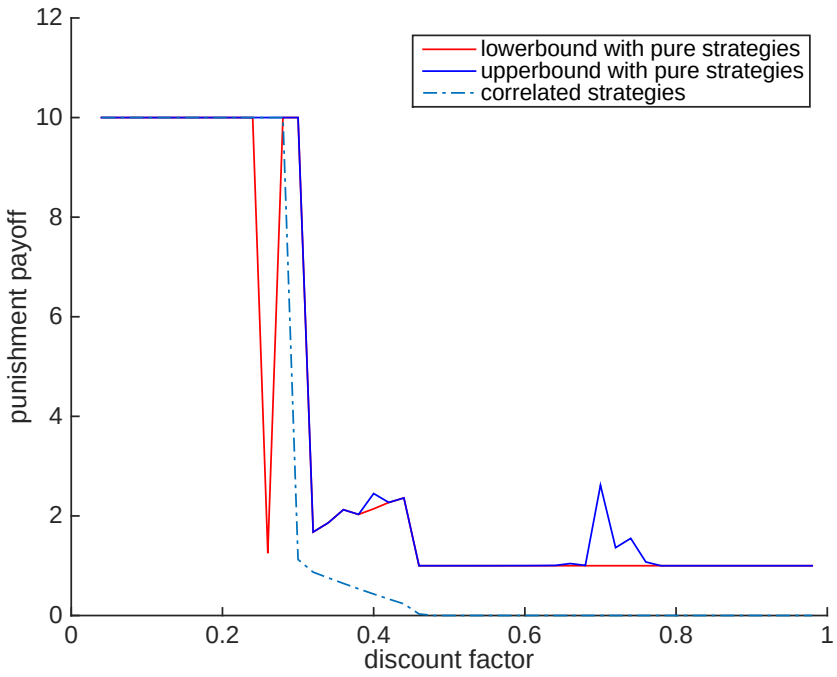

Figure 11: Punishment payoff in anti-no conflict game. Only bounds are computed succesfully with pure-strategies. The results with corralated strategies converged.

Abreu&Sannikov -algorithm is usually much faster than the one presented in this work. Totally 49 games are computed for the figure. Takes 7.6 seconds with Abreu&Sannikov -algorithm and over 1100 seconds with the algorithm for pure strategies. The Abreu&Sannikov-algorithm is reported to be over

1000 times quicker than older JYC algorithm [Abreu and Sannikov, 2013, Judd et al., 2003]. The running time of the pure-strategy algorithm seems to be somewhere between these, although the outcome is directly comparable.

5.4 Worst-case performance analysis

The computing work is proportional to the number of examined branches and to the work needed to compute each branch. It would be impossible, or at least extremely messy, task to derive exact bounds for computing time. Anyway, we can estimate work needed for pure brute-force search in the search tree and compare it work done by the algorithm. Of course, brute force search is also the worst-case bound for the algorithm.

The size of a search tree is computed by formula

$$W = \Sigma_{i=1}^{l} M^i = O(M^l) = O(m^{Nl}), \tag{39}$$

where l is the depth of search tree and M is the number of actions in a game. Alternately number of actions can be expressed as function of the number of players in a game N and the number of strategies available for each player m. The relation between these is $M = m^N$. This upper bound holds for any kind of algorithm or heuristic, which is used for computing the search-tree. The work, which is needed for one branch, increases at least linearly to number of players, number of strategies for each player and the length of the path. The worst-case work is at least

$$O(Nml) \tag{40}$$

N is the number of players in a game. In practical implementation the branch selection process is partly implemented in the work with each branch. That is why, we can expect a bit worse performance in computing branches and a bit better performance in the number of branches. There is an exponential relationship between work and number of players, which makes games with many players exhaustive. Remember that number of branches grows exponentially with number of players. The algorithm stops when upper and lower bounds are close enough to each other. The precision is now denoted by p, where p is the difference between upper and lower bound divided the size of feasible payoff set. In the example if all the payoffs are between 0 and 10. The precision 10^{-6} mean that the algorithm stops when difference between bounds is smaller than 10^{-5}. Next, we compute how many actions from start we need to include to ensure that the payoff is in wanted precision. We do

not assume any information about feasible continuation payoffs. $\overline{U(p)}$ and $\underline{U(p)}$ are maximum and minimum payoff from path p. Continuation payoff is just the largest and the smallest feasible payoff, which are denoted as $\overline{U}$ and $\underline{U}$. After each stage game the feasible payoff set is the scaled by discount factor. To ensure small enough set we need to solve equation

$$\begin{aligned} \overline{U(p)} - \underline{U(p)} &\leq p(\overline{U} - \underline{U}) \\ ((1-\delta)U(p^k) + d^k\overline{U}) - ((1-\delta)U(p^k) + \overline{d^k\underline{U}}) &\leq p(\overline{U} - \underline{U}) \\ \delta^l &\leq p \\ l &\geq \tfrac{\ln(p)}{\ln(\delta)}. \end{aligned} \tag{41}$$

l is now needed path length for wanted precision.

$$O(M^l) = O(M^{\frac{\ln(p)}{\ln(\delta)}}) = O(m^{N\frac{\ln(p)}{\ln(\delta)}}) \tag{42}$$

Note that examining all branches to length l does not guarantee convergence of the bounds. There might be open branches which could offer an equilibrium solution, if they would be further examined. However, we need to examine paths up to length l, if we want to ensure convergence in precision p in case there is only one branch left. Bounding the complexity of strategies or switching to ϵ-equilibria could offer possibilities for more strict conditions.

There really are some cases where the algorithm really converges to one solution. It is the most obvious case if we found an equilibrium path which gives the minimax payoff for every player. Another case is when all the open branches could give the payoff which equal best found punishment payoff in selected precision. This it is common especially if all the open branches are children of the current best solution. In this case, we do not know only the punishment payoff in selected precision, but also the first actions of the optimal punishment path. Although, it is seems likely that we have found the best paths, it is possible that the path would be different if there would not be a limit for complexity.

Figure 12 shows computational complexity with respect to different variables. Figures are computed directly from Formula 42 and with following parameters: $N = 2$, $m = 2$, $p = 10^{-5}$ and $\delta = 0.4$. Only one variable varies in figures. Exponential growth appears linear graph with logarithmic y-axis. We see that the work grows exponentially as function of the number of players, linearly with respect computing precision (note logarithmic scale also in x-axis). Increasing number of strategies for player affects only polynomially. The most dramatic is the growth when the discount factor grows. This

is because we examine paths from start and with high discount factor, the most of value comes from far the future. The paths needed are very long and the computing work increases over exponentially towards infinity when the discount factor grows to one.

5.5 Empirical performance analysis

There is no analytical worst case running time, but we can analyze algorithms performance with a random set of games. Each test set consists of 300 matrix games and the payoffs are chosen randomly between 1-10. The exception is is test set analyzing effect of the number of players, which is extremely hard to compute and consists of only 30 entries. The number of players is two, the number of strategies for each player is also two and the discount factor for each player is 0.4. The algorithm stops when precision 10^{-5} is reached. One of these variables is varied, and the performance is compared with theoretical upper bounds. The algorithm has a limit for the number of examined branches, which is set to 10^4. There is also some other conditions for the premature stop of the algorithm, but they are rare and they do not affect too much to averages. The average number of branches are presented in Figures 12 and the data used for computation is in Figure 14. The effects of different variables are explained in next subsections. A variable may affect the number of branches and the work of the average branch. Those are analyzed separately.

Altogether 930 example games are generated for the data set and general results are presented in Table 3. We see that 76.9% of games are solved in a sense that bounds are converged in selected precision. In 9% the algorithm is stopped, before founding solution and 14% stops because there are no more branches to examine. In some cases there the reason may be a rounding error or another numerical error, but the mostly this happens because there are no pure strategy equilibria in a game. This means that the algorithm solves almost 90% of the games in test sets.

Table 2 presents number of games in each category (Section 3.2). We see that the solution is the stage-game Nash equilibrium in 37% of cases (Category A). No equilibrium solution is found in 14% of games (Category D). In 60% of games repeating minimax-actions is an equilibrium strategy, if the discount factor is sufficiently high (Category B). In 40% of games high discount factors may lead to extremely long computation time (Category C).

Category	N	Freq.
A	344	37.0%
B	561	60.3%
C	369	39.7%
D	131	14.1%

Table 2: The number of games in each category

Reason of stop	N	Freq.
Solution found: bounds are converged and only one branch is left for each player	414	44.5%
Other/Unknown	0	0%
Data structure B full	11	1.2%
Data structure P full	0	0%
Bounds converged, but several branches left	301	32.4%
Max number of branches searched	73	7.9%
No branches left in B	131	14.1%

Table 3: Frequencies of algorithm stop conditions

5.5.1 The effect of discount factor

The algorithm handle even large discount factor much better than one could expect by theoretical analysis of the problem. If the largest of unequal discount factors is below 0.5 the convergence is quick as seen in Figure 12. There is clear step in complexity when discount factor 0.5 is exceed, but the growth is not dramatic. However, Figure 13 tells, that the percentage of solved games drops from 90% to only a bit over 50% when the largest discount factor in game grows from 0.4 to 0.9.

High discount factor forces us to examine paths further to ensure selected precision. On the other hand, the high discount factor allows players really execute more complex temporary structures. We see in Figure 15 that the optimal punishments really tend to be more complex punishment and even punishment strategies with dozens of actions are possible. The complexity is measured by minimum number of states in automata, which execute the punishment strategy for every player.

5.5.2 The effect of computing precision

One of the largest surprises is that selected precision does not affect much the algorithm performance. Figure 12 seems about constant difficulty in contrast to expected exponential growth. The most likely explanation is that the games typically are either easy or very exhaustive as we see in Figure 14. In easy games, the solution is found quickly regardless the precision and in exhaustive games the solution is not found at all and search is cut. Thus, the graph gives more information about the fraction of difficult and easy cases than the real effect of computing precision. The low precision let us quit computation, when we found just one, often simple, solution with a low payoff. That is why the solutions are often less complex. This is seen in Figure 15.

5.5.3 The effect of number of players

The empirical relationship between the number of players and the computation work cannot be studied well. We see in Figure 12 that only four players is enough to increase the average branch number over the stopping condition. With $2-4$ players computing work seems to grow faster than exponentially (convex line in logarithmic a graph). Actually, Figure 13 tells that no games with 5 player are solved successfully.

The algorithm can solve games with arbitrary many players, but curse of dimensionality makes the task very time consuming. The number of players has an exponential relationship to the needed branches and possibly an exponential relationship to time needed for one branch. This would mean total time $O(c^{N^2})$. Furthermore, it is possible that complexity of the punishment with many players tends to be high. This question cannot be studied well with current algorithms and computing capacity.

5.5.4 The effect of number of strategies for each player

The effect of varying number of strategies corresponds well the expectation: the polynomial growth in Figure 12. The fraction of solved games decreases naturally when the average computation task goes harder (Figure 13). Figure 16 shows the computation time instead of the number of branches. Now, the differences in average time for a branch become visible. Median computation time as function of the number of strategies seems to be exponential although the average time and the number of the branches grows only polynomially.

Theoretical and practical performance of the algorithm

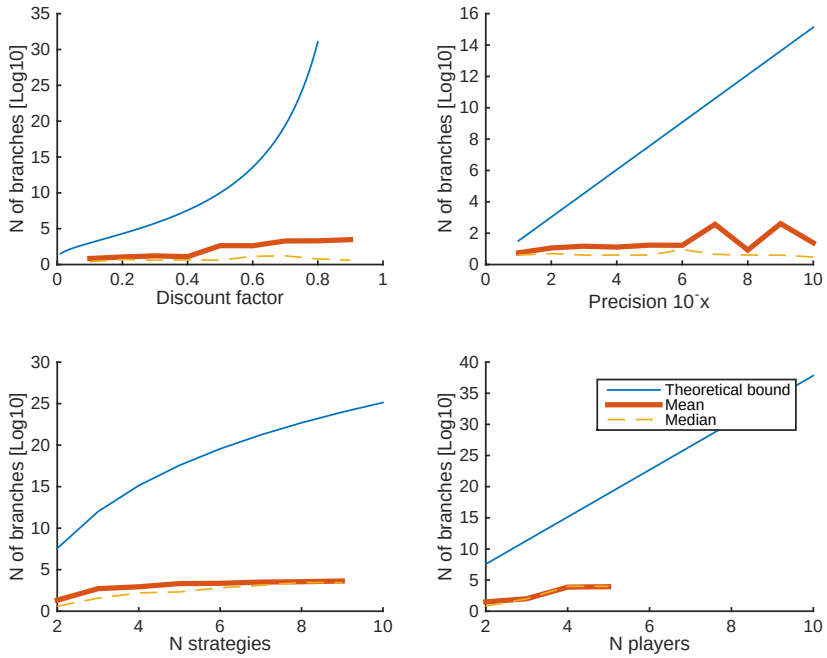

Figure 12: Theoretical bounds for number of branches versus pratical performance of the algorithm

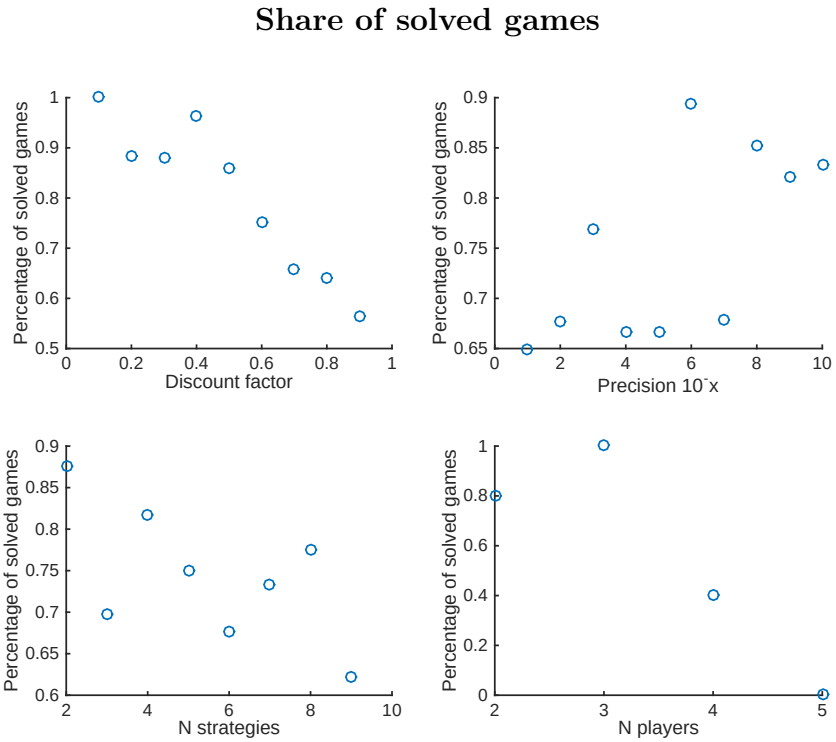

Figure 13: Percentage of succesfully solved games in test set

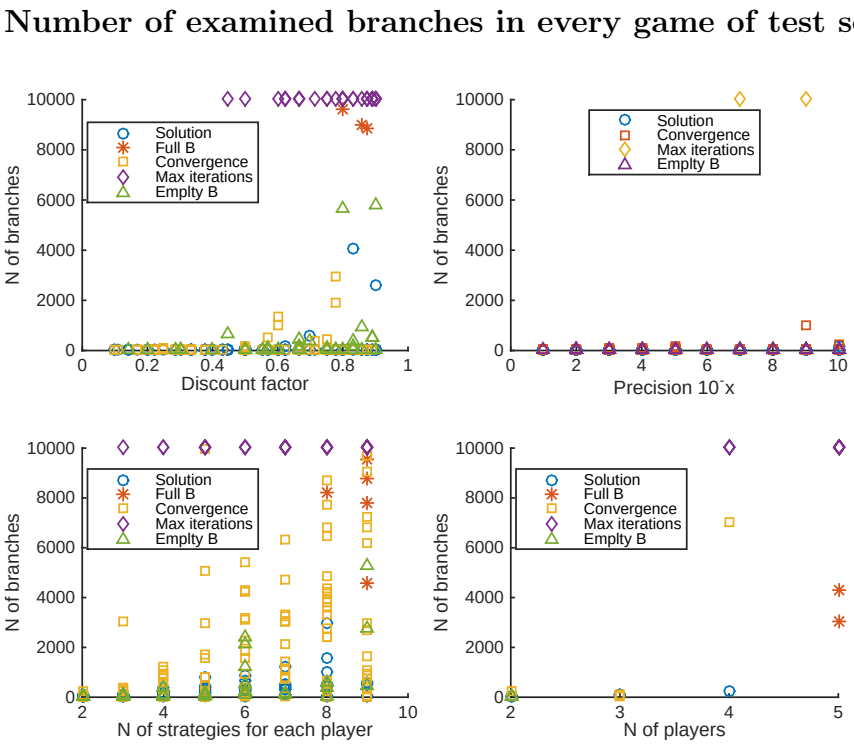

Figure 14: The number of branches and stopping reasons for all 930 games in tests. This data is used for computing means and medians for Figure 12.

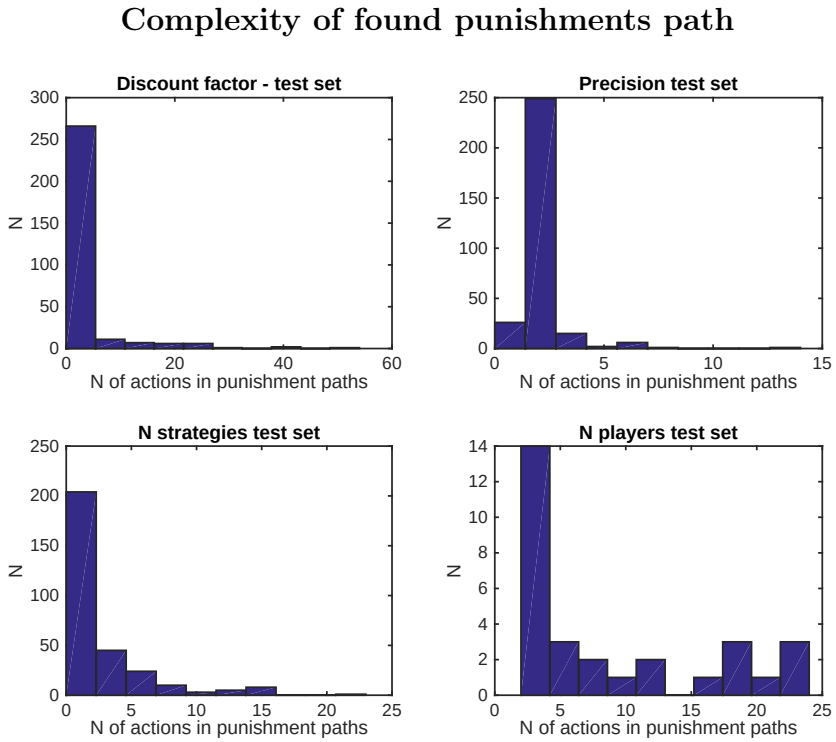

Figure 15: Histrograms presenteing total number of actions in best found punishment paths. Corresponds the minimun size of automata, which is needed to implement all punishments of a game.

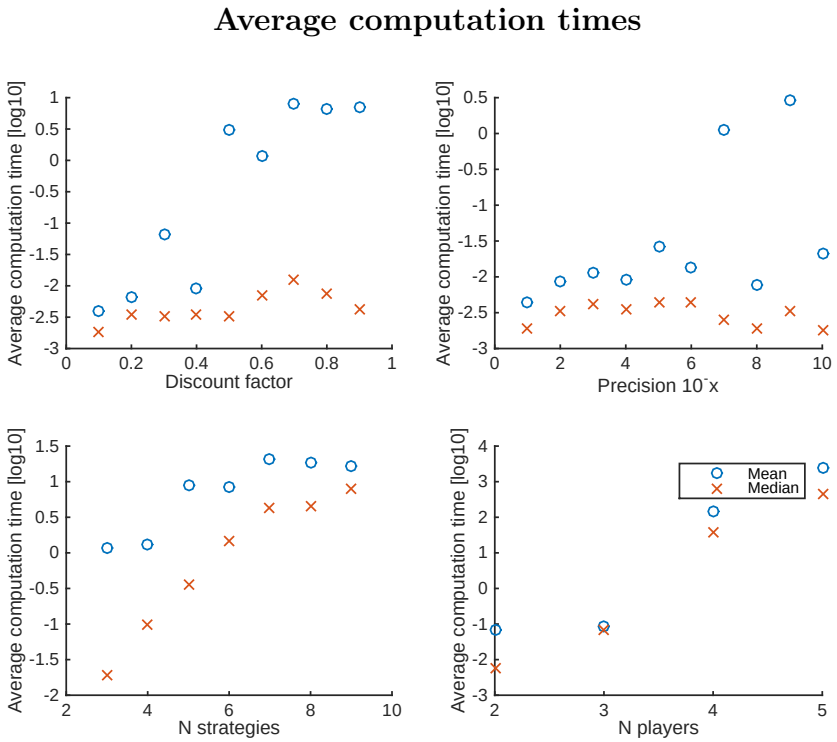

Figure 16: Mean and median computation times in test games

5.5.5 Computation time of a branch

The computation work of a branch is hard to analyze theoretically. Figure 17 shows computation times of all branches ($n = 383056$) in data set, where the discount factor varies and number of players two and their strategies are also two. The x-axis tells the length of a path. We see clearly a few games where computing time grows almost linearly in the figure. Because both axes are logarithmic, a linear line in the figure corresponds polynomial growth in computation times. The lines in the figure are not perfectly straight, but the growth rate is higher than polynomial. The reason for this is unclear. The possible explanation is that the computing just goes slower when there is lack of memory, but it is also possible that some steps in the algorithm take exponential time when the path length is high.

Figures 18 and 19 presents roughly the average computing time with the different number of players and strategies. The problem is that the path length has a significant factor and there a plane fitted to data points. We see that computation time increases linearly with the number of players, which means that empirical computation time (N of branches x Average branch time) increases about exponentially, while both factors increases about exponentially. Figure 19 predicts at least worse than polynomial performance when paths are short and number of strategies increases. Surprisingly, the average computation time seems to decrease when path length increases, but this must be a bias related the fact, that longer paths are examined only in a biased subset of games. Polynomial performance would make sense because the number of actions on a game increases polynomially when the number of strategies grow. Apparently there are some harder steps in the algorithm, which worsens the performance at least with short paths.

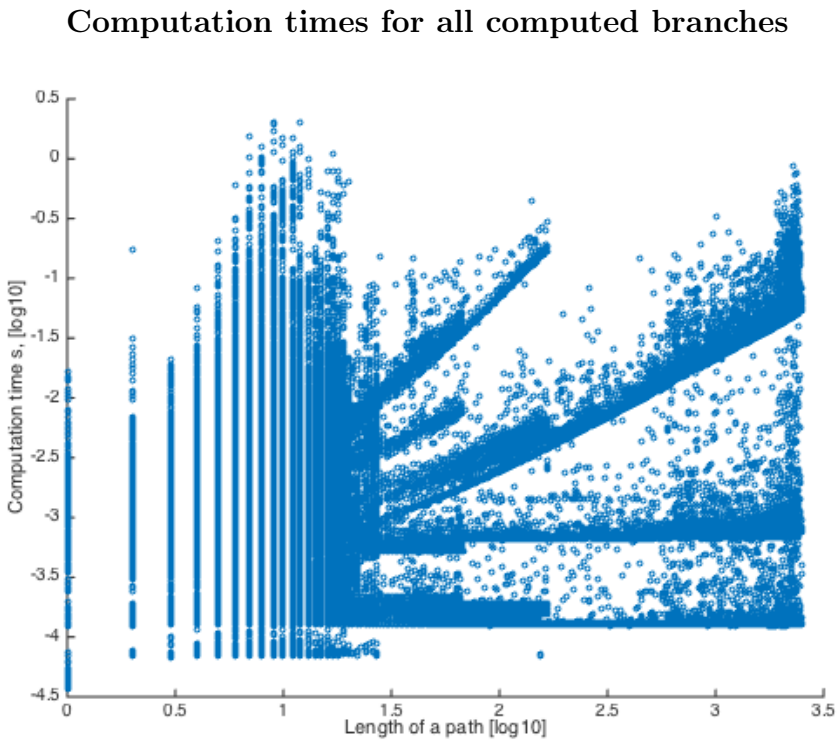

Figure 17: Computation times for branches in a test set with variable dicount factors. N=383 056.

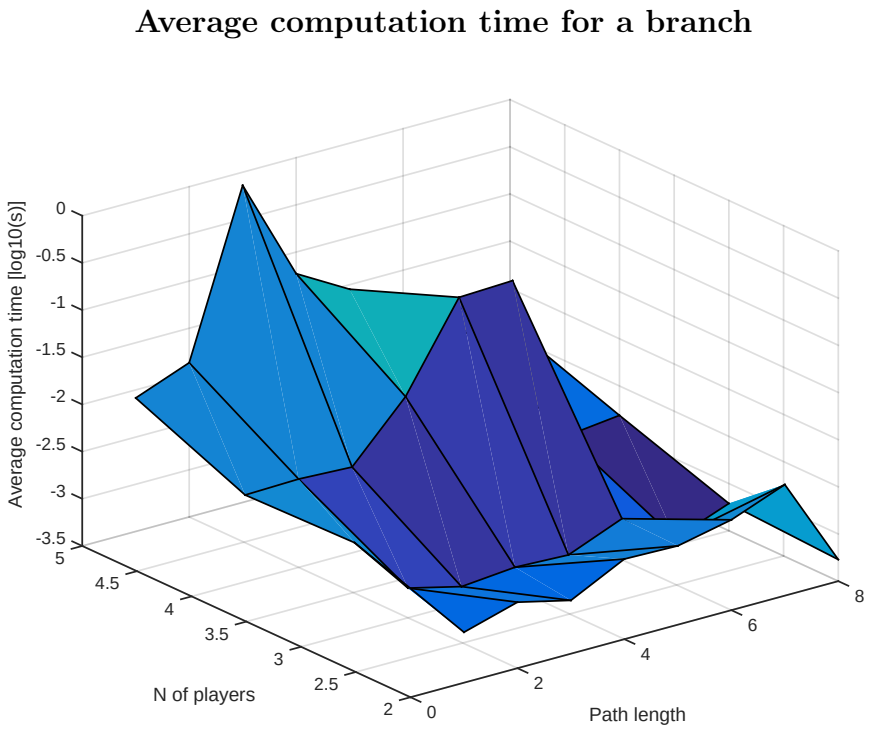

Figure 18: Average computation time for a brach in function of the number of players and the path length.

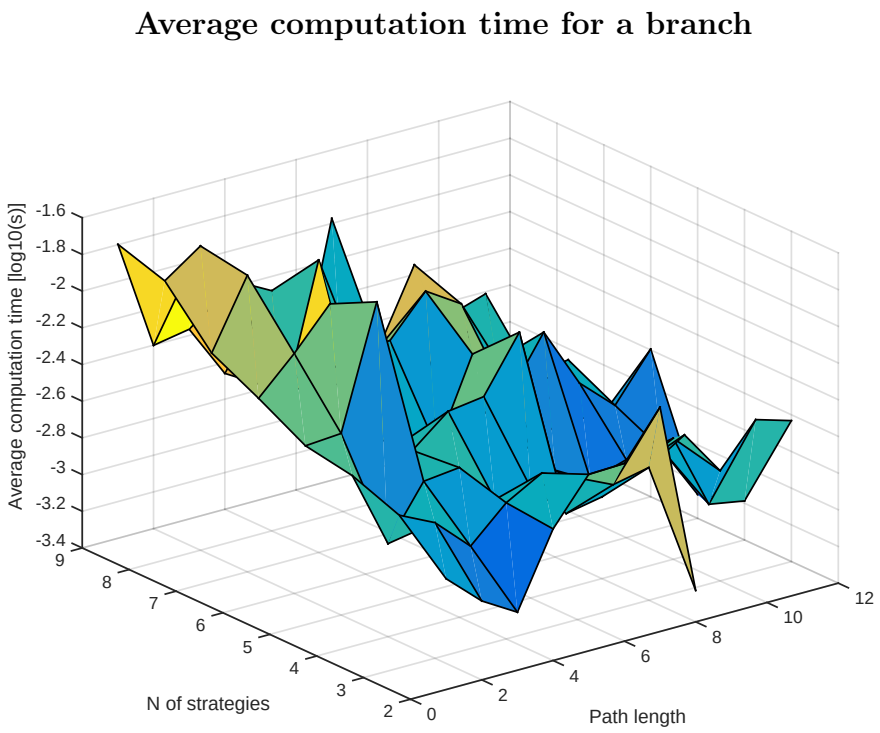

Figure 19: Average computation time for a brach in function of the number of strategies for each player and the path length.

6 Conclusions

This study examines pure-strategy equilibria in repeated games and presents an algorithm for computing the extremal solutions. The matrix games are divided into four categories. In some class the solutions are simple and in others they may be extremely complicated. The problem is formulated as an optimization problem and solved by an algorithm which applies the branch-and-bound idea. The algorithm returns bounds for extremal equilibrium solutions for each player. Also the best-found equilibrium path is returned for each player.

The algorithm restricts on strategies, which are representable as a finite automata. There are equilibrium strategies, which are not, but there is no practical method for implementing strategies of this kind. Therefore, it is a sound restriction to examine just finite strategies. Another common way to bypass problems related to infinitely complex equilibrium solutions is to approximate solutions with the ϵ-equilibria. The set of ϵ-equilibria also contains new solutions, which are not ϵ-approximations of any pure Nash equilibrium.

The numerical experiments show that the extremal equilibrium is very unstable related to the discount factor. The increase in the discount factor can lead either increase or decrease in the extremal equilibrium payoff. This phenomenon has an unintuitive influence: an equilibrium path may cease to be an equilibrium if the discount factor increases a bit [Berg, 2013].

6.1 Further research

There are many theoretical questions, which are without an proper answer. The most fundamental is conditions when the punishments are finitely complex. Also, the convergence conditions of the algorithm are not solved in this study. The algorithm performance could also be improved by a branch-selection heuristic, but this is still a minor question without proper converging conditions and speeds.

In wider perspective, this study tackles the fundamental question about extremal punishments. The theory of the repeated games leans on idea optimal punishments, which can be used as a threat. This study suggests that the extremal equilibria may be uncomputable or unimplementable is some games. On the other hand, this study proves that it is possible to find even complicated punishment strategies, if they are representable as a finite automata.

References

Dilip Abreu. Extremal equilibria of oligopolistic supergames. *Journal of Economic Theory*, 39(1):191–225, 1986.

Dilip Abreu. On the Theory of Infinitely Repeated Games with Discounting. *Econometrica*, 56(2):383, 1988.

Dilip Abreu and Yuliy Sannikov. An algorithm for two player repeated games with perfect monitoring. *Theoretical Economics*, 2013.

Dilip Abreu, David Pearce, and Ennio Stacchetti. Toward a Theory of Discounted Repeated Games with Imperfect Monitoring. *Econometrica*, 58 (5):1041, 1990.

Dilip Abreu, Prajit K Dutta, and Lones Smith. The folk theorem for repeated games: a NEU condition. *Econometrica*, 62(4):939–948, 1994.

Jean-Pierre Benoit and Vijay Krishna. Finitely Repeated Games. *Econometrica*, 53:905–922, 1985.

Kimmo Berg. Extremal Pure Strategies and Monotonicity in Repeated Games. *Working paper*, 2013. URL http://sal.aalto.fi/publications/pdf-files/mber14.pdf.

Kimmo Berg and Markus Karki. How patient players can get all the relevant payoffs in the symmetric 2 x 2 supergames? *Working paper*, 2013. URL http://sal.aalto.fi/publications/pdf-files/mber14b.pdf.

Kimmo Berg and Mitri Kitti. Computing Equilibria in Discounted 2x2 Supergames. *Computational Economics*, pages 1–18, 2011.

Kimmo Berg and Mitri Kitti. Equilibrium Paths in Discounted Supergames. *Working paper*, 2012. URL sal.aalto.fi/publications/pdf-files/mber09b.pdf.

Kimmo Berg and Mitri Kitti. Fractal geometry of equilibrium payoffs in discounted supergames. *Fractals: Complex Geometry, Patterns, and Scaling in Nature and Society*, 2014.

Henrik Björklund. Combinatorial Optimization for Infinite Games on Graphs. 2005. URL http://www.diva-portal.org/smash/get/diva2:165625/COVER01.

Christian Borgs, Jennifer Chayes, Nicole Immorlica, Adam Tauman Kalai, Mirrokni Vahab, and Christos Papadimitriou. The myth of the folk theorem. *Games and Economic Behavior*, 70(1):34–43, 2010.

Andriy Burkov and Brahim Chaib-draa. An Approximate Subgame-Perfect Equilibrium Computation Technique for Repeated Games. *AAAI*, 2010.

Bo Chen. On effective minimax payoffs and unequal discounting. *Economics Letters*, 100(1):105–107, 2008.

Bo Chen and Satoru Takahashi. A folk theorem for repeated games with unequal discounting. *Games and Economic Behavior*, 76(2):571–581, 2012.

X Chen and X Deng. Settling the complexity of 2-player nash-equilibrium. *In Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 261–272, 2006.

X Chen, X Deng, and S Ten g. Computing nash equilibria: Approximation and smoothed complexity. *In Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 603–612, 2006.

Mark B Cronshaw. Algorithms for Finding Repeated Game Equilibria. *Computational Economics*, 10(2):139–168, 1997.

M. Das and S.-M. Ngai. Graph-directed iterated function systems. *Indiana University Mathematics Journal*, 53:109–134, 2004.

Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou. Computing equilibria in anonymous games. *FOCS*, 2007.

Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009.

G. A. Edgar and J. Golds. A fractal dimension estimate for a graph - directed iterated function system of non-similarities. *Indiana University Mathematics Journal*, 48 (2):429–447, 1999.

James W Friedman. A non-cooperative equilibrium for supergames. *The Review of Economic Studies*, 38(1):1–12, 1971.

Drew Fudenberg and Eric Maskin. The folk theorem in repeated games with discounting or with incomplete information. *Econometrica: Journal of the Econometric Society*, pages 533–554, 1986.

Olivier Gossner and Johannes Hörner. When is the lowest equilibrium payoff in a repeated game equal to the minmax payoff? *Journal of economic theory*, 145(1):63–84, 2010.

Kenneth L Judd, Sevin Yeltekin, and James Conklin. Computing Supergame Equilibria. *Econometrica*, 71(4):1239–1254, 2003.

Ehud Kalai and William Stanford. Finite rationality and interpersonal complexity in repeated games. *Econometrica: Journal of the Econometric Society*, pages 397–410, 1988.

A. H. Land and A. G Doig. An Automatic Method of Solving Discrete Programming Problems. *Econometrica*, 28:497–520, 1960.

Omid Madani, Mikkel Thorup, and Uri Zwick. Discounted Deterministic Markov Decision Processes and Discounted All-Pairs Shortest Paths. *ACM Transactions on Algorithms*, 6(2), 2010.

George J Mailath and Larry Samuelson. *Repeated Games and Relationships: Long-Run Relationships*. Oxford University Press, 2006.

Stephen J Majeski. Arms races as iterated prisoner's dilemma games. *Mathematical Social Sciences*, 7(3):253–266, 1984.

Daniel Mauldin and S.C. Williams. Hausdorff dimension in graph directed constructions. *Transactions of the american mathematical society*, 1988.

John H. Nachbar and William R. Zame. Non-computable strategies and discounted repeated games*. *Economic Theory*, 8:103–122, 1996.

John Nash. Non-Cooperative Games. *Annals of Mathematics*, 54:286–295, 1951.

Ariel Rubinstein. Finite automata play the repeated prisoner's dilemma. *Journal of Economic Theory*, 39:83–96, 1986.

Bruno Salcedo and Bruno Sultanum. Computation of subgame-perfect equilibria of repeated games with perfect monitoring and public randomization. 2012. URL http://www.academia.edu/2097084/Computation_of_Subgame-perfect_Equilibria_of_Repeated_Games_with_Perfect_Monitoring_and_Public_Randomization.

Herbert Simon. Theories of Bounded Rationality. *Decision and Organization*, pages 161–176, 1972.

Dale O Stahl. The graph of prisoners' dilemma supergame payoffs as a function of the discount factor. *Games and Economic Behavior*, 3(3):368–384, 1991.

Quan Wen. The" folk theorem" for repeated games with complete information. *Econometrica: Journal of the Econometric Society*, pages 949–954, 1994.

A Summary of the notation

players	i ∈ N={1,..,n}
action set	$a_i \in A_i$
action profile	a ∈ A
opponents action profile	$a_{-i} \in A_{-i} = \times_{j \neq i} A_i, j \in N$
payoff (from action profile)	$u_i(a)$
maximum payoff	$\overline{v}_i$
minimax payoff	$\underline{v}_i$
minimax action profile	$\underline{a}_i$
best deviation payoff	$v_i^*(a) = \max_{a_i \in A_i} u_i(a)$
best deviation action profile	a_i^*
k length history	$A^k = \times_k A$
set of all histories	$\mathcal{A} = \bigcup_{k=0}^{\infty} A^k$, $A^0 = \{\emptyset\}$
path	$p = \times_k A$
path length	|p|
path from j:th action	p_j
j first actions of p	p^j
first action of p	f(p)
pure strategy	$\sigma_i^k : A^k \to A_i$
strategy profile	$\sigma = (\sigma_1, ..., \sigma_n)$
average payoff	$U_i(\sigma)$
discount factor	δ_i
punishment payoff	v_i^-
branch	$b = \times_{i \in N} A_i^k$
parent of branch b	par(b)
bounds for payoff from b	lb(b), ub(b)
continuation payoff	v_i
minimal continuation payoff	con_i