Diploma Thesis

Equilibrium Computation for 2-Player Extensive Games with Simultaneous Moves

Alexander Bisaliev

April 20th, 2011

Advisors:

PD Dr. Dirk Lebiedz
Prof. Dr. Bernhard Nebel

Albert-Ludwigs-Universität Freiburg
Fakultät für Mathematik und Physik

Abstract

The class of extensive games with simultaneous moves is a subset of extensive games with imperfect information. In this thesis, we introduce an efficient representation form for 2-player extensive games with simultaneous moves and provide two algorithms for finding a sample equilibrium: The baseline algorithm and the interval-heuristic (IH) algorithm.

We show that the baseline algorithm computes a subgame perfect equilibrium in an extensive game with simultaneous moves, which is a perfect Bayesian equilibrium, if we convert the game into the corresponding extensive game with imperfect information. We compare the runtime of our implementation of the baseline algorithm with the runtime of the gambit's implementation of Lemke's algorithm, which is the state of the art algorithm for equilibrium computation in extensive games with imperfect information. Our experimental results show that the baseline algorithm is much more efficient than Lemke's algorithm for extensive games with simultaneous moves.

We also prove that the IH algorithm finds a strategy which is part of a subgame perfect equilibrium in an extensive game with simultaneous moves. In addition we test the baseline algorithm and the IH algorithm on the simultaneous Tic-Tac-Toe game from General Game Playing (GGP) and compare their performance.

Acknowledgements

I thank PD Dr. Dirk Lebiedz for supervising this work and providing me such a great work environment in his research group. I thank Prof. Dr. Bernhard Nebel for supervising my thesis and awaking my interest for game theory. I am very grateful to Robert Mattmüller and Jens Witkowski for providing continual assistance, proofreading this thesis many times and answering tons of my questions about game theory and programming in C++. Special thanks are due to Jochen Siehr, Markus Esenwein and Marcel Rehberg for proofreading, making very detailed comments, and helping me out with MATLAB® and LaTeX. I also thank Corinna Seidl, Andrea and Felix Menne for proofreading parts of this thesis. It was a great pleasure to work in the research group with Jonas Unger, Dominik Skanda and Marc Fein. I am grateful to Bernhard Link for providing me informations about debugging my programs.

Contents

1. Introduction

1.1. Motivation

Two-player extensive games with simultaneous moves allow us to model situations in which two decision-makers simultaneously interact with each other. The main fields of application are economic, political and social sciences. Possible situations are: a challenge between a company entering a market which is controlled by a monopoly, two political candidates competing for votes, phenomena of relationships, etc. More examples are given in the the textbook by Osborne [2, pp. 213–226].

Extensive games with simultaneous moves are a subset of extensive games with imperfect information. However, there are easier equilibrium concepts there than in imperfect-information extensive games. We develop a new efficient representation form and solution methods for finding a sample equilibrium in extensive games with simultaneous moves. The main idea of our representation form is to use normal-form games to express the simultaneity and the extensive games tree structure to model the complex multi-move situation. Both algorithms can be applied in General Game Playing.

In this work we will only consider finite 2-player extensive games with perfect recall. Perfect recall means that at every point of the game each player remembers everything he knew before.

1.2. Thesis Outline

In Chapter 2 we overview theoretic and algorithmic concepts of normal-form games. Chapter 3 is about extensive form games. We introduce extensive games with perfect information in Section 3.2 and extensive games with imperfect information in Section 3.3. The main focus of this thesis is about extensive games with simultaneous moves which are discussed in Chapter 4. In Section 4.2 we introduce the theoretical concepts for this class of games. Our first solution method — the baseline algorithm — is presented and analysed in Section 4.3. In Section 4.4 we focus on the interval-heuristic (IH) algorithm. We present some experimental results in Chapter 5 obtained from implementing a simultaneous Tic-Tac-Toe game from general game playing and finding a sample equilibrium there with the baseline and the IH algorithms. At the end of our work we summarize our results and give some ideas on future work in Chapter 6.

2. Equilibrium Computation for 2-Player-Bimatrix Games

In this chapter we briefly introduce the basic game theoretic concepts for finite 'normal-form' games. We give the formal definition of 'game' and consider the concept of Nash equilibrium which is the mostly used term to describe the 'optimal strategies'. The definitions and notations are based on the terms introduced by Osborne and Rubinstein in [1] and [2].

2.1. Preliminaries

Notation 2.1.1. *We assume that the players are enumerated by* 0 *and* 1*. They are referred to as player* 0 *and player* 1*.*

Definition 2.1.2. (Bimatrix Game) *A finite 2-player* bimatrix (also called normal-form) game *is a tuple* $G = \langle A_0, A_1, u_0, u_1 \rangle$*, where for player* $i \in \{0, 1\}$*:*

- A_i *is a finite set of player* i*'s actions (or pure strategies). Each* 2*-dim. vector* $a \in A := A_0 \times A_1$ *is called a strategy profile;*
- $u_i : A \to \mathbb{R}$ *is called player* i*'s utility function.*

Note 2.1.3. *Let* $G = \langle A_0, A_1, u_0, u_1 \rangle$ *be a normal-form game with* $|A_0| = m$ *and* $|A_1| = n$*. The utility function* u_i *can be interpreted as player* i*'s* $(m \times n)$ *payoff-matrix. The game is played by simultaneously choosing a row* i *from* u_0 *by player 0 and column* j *from* u_1 *by player 1. Then they receive payoffs* u_0^{ij} *and* u_1^{ij}*, respectively. We represent the games in the short tabular form as in Figure 2.1.*

Example 2.1.4. (Prisoners' Dilemma) *In the prisoners' dilemma game each player (prisoner) has two actions: 'confess' (C) and 'decline' (D). Their payoffs are given in the following table as two-dimensional vectors, where the first number is payoff of the player* 0*, and the second number is payoff of the player* 1*.*

	D	C
D	3,3	0,4
C	4,0	1,1

Figure 2.1.: The Prisoner's Dilemma.

Definition 2.1.5. (Zero-Sum and General-Sum Games) *A game is called* zero-sum *if the sum of the payoff-matrices is zero:* $u_0 + u_1 = 0$. *Otherwise it is called* general-sum.

Definition 2.1.6. (Mixed Strategy) *A* mixed strategy *of player* $i \in \{0,1\}$ *is a probability distribution vector* α_i *over the set of player* i*'s actions* A_i. *The set of pure strategies with strictly positive probability is called the* support *of mixed strategy* $\text{supp}(\alpha_i) = \{a_i \in A_i \mid \alpha_i(a_i) > 0\}$. *The set of player* i*'s mixed strategies is denoted by* $\Delta(A_i)$. *The set of mixed strategy profiles is denoted by* $\Delta(A) := \Delta(A_0) \times \Delta(A_1)$.

Definition 2.1.7. (Expected Payoff) *Player* i*'s* expected payoff $U_i(\alpha)$ *of the mixed strategy profile* $\alpha = (\alpha_0, \alpha_1) \in A$ *is defined as*

$$U_i(\alpha) := \alpha_0^\top \cdot u_i \cdot \alpha_1.$$

Definition 2.1.8. (Best Response) *A mixed strategy* α_i *of player* $i \in \{0,1\}$ *is* best response *to the mixed strategy* α_{1-i} *of player* $1-i$ *if it maximizes his expected payoff. The set of player* i*'s best responses to the strategy* α_{1-i} *of player* $1-i$ *is defined as*

$$B_i(\alpha_{1-i}) := \{\alpha_i^* \in \Delta(A_i) \mid U_i(\alpha_i^*, \alpha_{1-i}) = \max_{\alpha_i \in \Delta(A_i)} \{U_i(\alpha_i, \alpha_{1-i})\}\}.$$

One of the most important tools in the theory of normal-form games is the concept of *Nash equilibrium*. Nash equilibrium in pure strategies is a special case of Nash equilibrium in mixed strategies. Thus we define it directly for a general case.

Definition 2.1.9. (Nash Equilibrium) *A* Nash equilibrium *is a strategy profile* $\alpha^* \in \Delta(A)$ *such that for every player* $i \in \{0,1\}$*:*

$$U_i(\alpha^*) \geq U_i(\alpha_{1-i}^*, \alpha_i) \; \forall \; \alpha_i \in \Delta(A_i).$$

Example 2.1.10. *The strategy profile* (C, C) *is a Nash equilibrium in the prisoners' dilemma game from Example 2.1.4 since:*

$$U_0(C,C) = 1 > 0 = U_0(D,C) \text{ and}$$
$$U_1(C,C) = 1 > 0 = U_1(C,D).$$

Lemma 2.1.11. (Existence of Nash Equilibrium) *Every finite 2-player bimatrix game has a Nash Equilibrium in mixed strategies.*

Proof. It is a special case of the *Nash Theorem*. See [1, pp. 19–20]. □

Lemma 2.1.12. (Support Lemma) *Let* $G = \langle A_0, A_1, u_0, u_1 \rangle$ *be a finite 2-player-bimatrix game. Then* $\alpha^* \in \Delta(A)$ *is a Nash equilibrium in mixed strategies if and only if for every player* $i \in \{0,1\}$ *each pure strategy from* $supp(\alpha_i^*)$ *is a best response to* α_{1-i}^*.

Proof. See [1, pp. 33–34]. □

2.2. Pure Equilibrium Computation

In this section we introduce algorithms for pure equilibrium computation in the normal-form games.

2.2.1. Computing a Sample Pure Equilibrium

Let $G = \langle A_0, A_1, u_0, u_1 \rangle$ be a 2-player-bimatrix game with $A_0 = \{1, \ldots, m\}$ and $A_1 = \{1, \ldots, n\}$. There is a simple algorithm for finding a sample pure equilibrium (if one exists). First find one of the column-maxima in u_0. If there is a row-maximum at the same column in u_1, then it is a pure equilibrium. If not, repeat the procedure with the next column of u_0. Some games can have multiple column-maxima in u_0 but only one of them is an equilibrium. To avoid skipping some column-maxima, save them all in an array and test each if there is the row-maximum in u_1. The pseudocode is presented in Algorithm 1.

Algorithm 1 Computing a Sample Pure Equilibrium

```
1: function GetPureEquilibrium(G)
2:     for j ← 1, ..., n do
3:         A ← [1]                          ▷ Array with indices of the column maxima
4:         stop ← false
5:         for i ← 2, ..., m do
6:             if u0(A[|A| − 1], j) < u0(i, j) then
7:                 A ← [i]                          ▷ Reset the array A to [i].
8:             else if u0(A[|A| − 1], j) = u0(i, j) then
9:                 A[|A|] ← i
10:            end if
11:        end for
12:        for k ← A[0], ..., A[|A| − 1] do
13:            for l(← 2, ..., n) ∧ (¬stop) do
14:                if u1(k, j) < u1(k, l) then
15:                    stop ← true
16:                end if
17:            end for
18:            if stop = false then
19:                return (k, j)
20:            end if
21:        end for
22:    end for
23: end function
```

2.2.2. Computing All Pure Equilibria

We find all pure equilibria in a 2-player-bimatrix game if we continue the computation after finding the first equilibrium. The algorithm for finding all pure equilibria is implemented in the GetPureEquilibria function.

2.2.3. Computing a Sample Pure Equilibrium with Certain Properties

Sometimes it is useful to find a pure equilibrium with certain properties. For example an equilibrium with the maximal payoff for player p. The algorithm is implemented by overloading the function GetPureEquilibrium with three input values. The first input parameter is a game G, the second is a player $p \in \{0, 1\}$, and the third input value is a Boolean variable max, which indicates maximization of player p's payoff, if it is $true$, or to minimize it otherwise. The algorithm finds a pure equilibrium (if it exists) with player p's maximal (or minimal) utility. First it finds all equilibria in G with the function GetPureEquilibria and saves them into a vector v. Then the algorithm searches in v for an equilibrium with the maximal (or minimal) payoff for player p. We denote the player p's expected utility at the i-th equilibrium v_i with $U_p(v_i)$ for $i \in \{1, \ldots, |v|\}$. The pseudocode is presented in Algorithm 2.

Algorithm 2 Computing a Sample Pure Equilibrium with Certain Properties

```
1: function GetPureEquilibrium(G, p, max)
2:     v ←GetPureEquilibria(G)              ▷ v contains all pure equilibria of G
3:     j ← 1
4:     for i ← 1, ..., |v| do
5:         if (max ∧ U_p(v_j) < U_p(v_i)) ∨ (¬max ∧ U_p(v_j) > U_p(v_i)) then
6:             j ← i
7:         end if
8:     end for
9:     return v_j
10: end function
```

2.3. Mixed Equilibrium Computation

Some 2-person-bimatrix games do not have a pure equilibrium. For example, the zero-sum games. In this section we present different ways for computing equilibria in the mixed strategies. First we take a look at the zero-sum games which can be solved by linear programming. Afterwords we observe approaches for solving the general-sum games using linear complementary and mixed integer programming.

2.3.1. Linear Programming

Linear programming (LP) is a technique for computing a solution for a system of linear inequality constraints optimizing (maximizing or minimizing) a linear objective function. Linear Programs can be expressed in the canonical form given in equations 2.1–2.3.

$$\text{Maximize } c^\top x \text{ subject to} \tag{2.1}$$
$$Ax \leq b, \tag{2.2}$$
$$x \geq 0. \tag{2.3}$$

Such that $x \in \mathbb{R}^n$, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^m \times \mathbb{R}^n$. An *objective function* is given by $c^\top x$ and is to be maximized subject to the linear constraints $Ax \leq b$ and $x \geq 0$. The linear constraints $Ax \leq b$ define a *convex polytope*. A vector $v \in \mathbb{R}^n$ is called a (feasible) *solution* to the LP if v satisfies every constraint in 2.2 and 2.3. Let S denote a set of all solutions to the LP. A linear program is called *feasible* if S is not empty. A solution v^* is called an *optimal solution*, if $c^\top v^* = \max_{x \in S} c^\top x$.

2.3.2. Computing a Mixed Equilibrium in Zero-Sum Games

The zero-sum games are easy to solve, since the sum of players' payoffs is zero and a Nash equilibrium is a pair of *maximinimizers*[1]. Let $A_0 = \{a_1, \ldots, a_m\}$, $A_1 = \{b_1, \ldots, b_n\}$ and u_i^* be player i's expected utility in equilibrium. Player i searches for a mixed strategy α_i, such that his expected utility is maximized, and the expected utility of player $1 - i$ is minimized. This yields the following linear program for player i:

$$\text{Maximize } u_i^* \text{ s.t.:} \tag{2.4}$$
$$\alpha_i(a_j) \geq 0 \; \forall \; j \in \{1, \ldots, m\}, \tag{2.5}$$
$$\sum_{j=1}^{m} \alpha_i(a_j) = 1, \tag{2.6}$$
$$U_i(\alpha_i, b_k) = \sum_{j=0}^{m} \alpha_i(a_j) \cdot u_i(a_j, b_k) \geq u_i^* \text{ for all } k \in \{1, \ldots, n\}. \tag{2.7}$$

The solution of the LP 2.4–2.7 is a maximinimizer for player i. It can be solved by the *simplex algorithm*. SolveLP function (Algorithm 3) obtains two linear programs, one for each player, from an input game G and solves them with LP_SOLVE[2], which is a free linear (integer) programming solver based on the revised simplex method and the Branch-and-bound method for the integers.

[1] See [5, pp. 89–90]

[2] For further informations see the website http://lpsolve.sourceforge.net.

Algorithm 3 Computing a Sample Equilibrium in Zero-Sum Games

```
1: function SolveLP(G)
2:     lp_0 ← "Generate LP from G for player 0."
3:     lp_1 ← "Generate LP from G for player 1."
4:     LP_SOLVE(lp_0)
5:     LP_SOLVE(lp_1)
6:     return Solutions of lp_0 and lp_1
7: end function
```

2.3.3. Computing a Mixed Equilibrium in General-Sum Games

The problem of finding an equilibrium in general-sum games cannot be represented as a linear program since the players' payoffs are not zero-sum. A player cannot maximize his expected payoff by minimizing the expected payoff of the opposite player. We formulate the problem as *linear complementary program* (LCP). Let $G = \langle A_0, A_1, u_0, u_1 \rangle$ be a general-sum 2-player bimatrix game with $A_0 = \{a_1, \ldots, a_m\}$, $A_1 = \{b_1, \ldots, b_n\}$ and a Nash equilibrium (α, β) with the expected payoff (u, v). The LCP formulation for finding a sample Nash equilibrium in G is given in equations 2.8–2.13.

$$u - U_0(a_i, \beta) \geq 0 \; \forall \; i \in \{1, \ldots, m\}, \tag{2.8}$$

$$v - U_1(\alpha, b_j) \geq 0 \; \forall \; j \in \{1, \ldots, n\}, \tag{2.9}$$

$$\alpha(a_i) \cdot (u - U_0(a_i, \beta)) = 0 \; \forall \; i \in \{1, \ldots, m\}, \tag{2.10}$$

$$\beta(b_j) \cdot (v - U_1(\alpha, b_j)) = 0 \; \forall \; j \in \{1, \ldots, n\}, \tag{2.11}$$

$$\alpha(a_i) \geq 0 \; \forall \; i \in \{1, \ldots, m\}, \; \sum_{i=0}^{m} \alpha(a_i) = 1, \tag{2.12}$$

$$\beta(b_j) \geq 0 \; \forall \; j \in \{1, \ldots, n\}, \; \sum_{j=0}^{n} \beta(b_j) = 1. \tag{2.13}$$

The equations 2.10 and 2.11 reflect the Nash equilibrium condition from Lemma 2.1.12. For example, the product in 2.10 is zero if and only if either $\alpha(a_i)$ is zero or $u - U_0(a_i, \beta)$ is zero. If $\alpha(a_i) = 0$, then $a_i \notin \text{supp}(\alpha)$ and therefore a_i is not optimal. If $u - U_0(a_i, \beta) = 0$, then player 0's expected payoff $U_0(a_i, \beta)$ from playing a_i is equal to the expected payoff at the Nash equilibrium (α, β). Equation 2.11means the same for player 1. The LCP from above can be solved with *Lemke-Howson* algorithm which resembles the simplex algorithm and consists of iterated pivoting. The algorithm originally appeared in in the paper by Lemke and Howson [9], in 1964. It can take an exponential number of iterations for comupting a solution. There is no polynomial technique known by now for finding a sample Nash equilibrium in general-sum games.

Another approach for finding a Nash equilibrium in 2-persons-bimatrix games was presented by Sandholm, Gilpin and Conitzer in their paper [7]. They introduced an algorithm (*MIP Nash* algorithm) based on mixed integer programming (MIP). MIP is

a linear program with some variables constrained to be integers. The MIP formulation for the problem of finding a Nash equilibrium is based on the idea of *regret* r_{a_i} of player i's pure strategy $a_i \in A_i$, given the mixed strategy of the player $1-i$. The 'regret' r_{a_i} of some action a_i is the difference between player i's utility from playing pure strategy a_i and the optimal strategy α_i^*, given the mixed strategy of player $1-i$. The criterium for Nash equilibrium can be reformulated using regret: *The mixed strategy profile* (α_0^*, α_1^*) *is a Nash equilibrium if and only if for each player* $i \in \{0,1\}$ *every pure strategy from* α_i^* *has either regret* $= 0$ *or does not belong to support of* α_i^*.

Sandholm, Gilpin and Conitzer present four MIP formulations. In their first formulation Nash equilibria are the only feasible solutions of the MIP. Thus we can optimize over the solutions and find equilibria with some properties. For example, an equilibrium maximizing the utility for one player or maximizing the social welfare (the sum of the players' payoffs). Let $G = \langle A_0, A_1, u_0, u_1 \rangle$ be a 2-player-bimatrix game and (α_0, α_1) a Nash equilibrium with expected payoff (U_0, U_1). We define following variables:

- Player i's expected utility u_{a_i} from playing a pure strategy $a_i \in A_i$ given the mixed strategy α_{1-i} of the opposite player is defined as:

$$u_{a_i} = \sum_{a_j \in A_{1-i}} \alpha_{1-i}(a_j) \cdot u_i(a_i, a_j);$$

- The regret $r(a_i)$ of player i's pure strategy $a_i \in A_i$ is the difference between his expected payoff U_i at the equilibrium (α_0, α_1) and the expected payoff u_{a_i} from playing a_i:

$$r(a_i) := U_i - u_{a_i};$$

- For each pure strategy $a_i \in A_i$ we define a boolean variable $f_{a_i} \in \{0,1\}$. If $f_{a_i} = 0$, then $a_i \notin supp(\alpha_i)$. If $f_{a_i} = 1$, then a_i can be in the $supp(\alpha_i)$ but the regret r_{a_i} of a_i must be zero.

- The constant Δ_i is the maximal difference between two utilities of player i:

$$\Delta_i := \max_{a,a' \in A} (u_i(a) - u_i(a')).$$

The MIP formulation is given in the equations 2.14–2.20.

$$\sum_{a_i \in A_i} \alpha_i(a_i) = 1 \qquad \text{for all } i, a_i \in A_i, \tag{2.14}$$

$$u_{a_i} = \sum_{a_j \in A_{1-i}} \alpha_{1-i}(a_j) \cdot u_i(a_i, a_j) \qquad \text{for all } i, a_i \in A_i, \tag{2.15}$$

$$U_i \geq u_{a_i} \qquad \text{for all } i, a_i \in A_i, \tag{2.16}$$

$$r_{a_i} = U_i - u_{a_i} \qquad \text{for all } i, a_i \in A_i, \tag{2.17}$$

$$\alpha_i(a_i) \leq 1 - f_{a_i} \qquad \text{for all } i, a_i \in A_i, \tag{2.18}$$

$$r_{a_i} \leq \Delta_i \cdot f_{a_i} \qquad \text{for all } i, a_i \in A_i, \tag{2.19}$$

$$\alpha_i(a_i) \geq 0,\ r_{a_i} \geq 0,\ f_{a_i} \in \{0, 1\} \qquad \text{for all } i, a_i \in A_i. \tag{2.20}$$

2.3.4. Mixed Equilibrium with Certain Properties

If we solve the MIP with an objective '*maximize* U_i *subject to* equations 2.14–2.20', then the solution is an equilibrium with maximal expected utility for player $p \in \{0, 1\}$. The pseudocode of our implementation is given in Algorithm 4.

Algorithm 4 Computing a Mixed Equilibrium with Certain Properties

```
1: function SolveMIP(G, p, max)
2:     mip ← Generate MIP from G
3:     if p ∈ {0,1} ∧ max = true then
4:         Set the objective function: maximize U_p
5:     else if p ∈ {0,1} ∧ max = false then
6:         Set the objective function: minimize U_p
7:     end if
8:     return lp_solve(mip)
9: end function
```

3. Extensive Games

Extensive games allow us to model situations where each player may take more than just one action. In this chapter we introduce extensive games with perfect information (Section 3.1) and different equilibrium concepts: a Nash equilibrium, which ignores a sequential structure of such games, and a subgame perfect equilibrium. Then we consider imperfect-information extensive games (Section 3.2) and introduce the concept of perfect Bayesian equilibrium. Although we only consider extensive games without chance moves, we can easily extend our models and algorithms to extensive games with chance moves. A definition of perfect-information extensive game with chance moves is given in the textbook by Osborne and Rubinstein [1]. This chapter is a summary of the notions presented by Shoham and Leyton-Brown (see [5]) as well as by Peters (see [3]).

3.1. Extensive Games with Perfect Information

In perfect-information extensive games each player is perfectly informed about all the events that happened previously, while choosing an action at his decision node. Informally, an extensive game with perfect information is a tree, in which each non-terminal node corresponds to a choice of one player, each leaf corresponds to a final payoff where each player gets his utility, and the edges correspond to the possible actions.

Definition 3.1.1. (Perfect-Information Extensive Game) *A 2-player finite* extensive game with perfect information *is a tuple* $G = \langle A, H, Z, \chi, \rho, \sigma, u \rangle$ *where:*

- A *is a finite set of* actions*;*
- H *is a finite set of* non-terminal nodes*;*
- Z *is a set of* terminal nodes*, s.t.* $H \cap Z = \emptyset$*;*
- $\chi : H \to 2^A$ *is called* action function*;*
- $\rho : H \to \{0, 1\}$ *is called* player function*;*
- $\sigma : H \times A \to H \cup Z$ *is a bijective* successor function*;*
- $u = (u_0, u_1)$ *where* $u_i : Z \to \mathbb{R}$ *is called player* i*'s* utility function*.*

Definition 3.1.2. (History) *A* history *of node* $h \in H \cup Z$ *is a sequence of actions* $\langle a^0, \ldots, a^n \rangle$ *leading from the root node to* h*.*

Notation 3.1.3. *We denote the set of non-terminal histories with Σ_H, the set of terminal histories with Σ_Z and the set of all histories with $\Sigma := \Sigma_H \cup \Sigma_Z$.*

Definition 3.1.4. (Pure Strategy) *A* pure strategy *s_i of player $i \in \{0,1\}$ in $G = \langle A, H, Z, \chi, \rho, \sigma, u\rangle$ is a map assigning an action $a \in \chi(h)$ to every decision node $\{h \in H \mid \rho(h) = i\}$ of player i. The set of pure strategies of player i is defined in the following way:*

$$S_i := \prod\nolimits_{h \in H, \rho(h)=i} \chi(h).$$

The above definition of pure strategy requires the definition of player i's action in *each* node even if the node is not reached.

Definition 3.1.5. (Outcome of Pure Strategy) *An* outcome *$O : S_0 \times S_1 \to Z$ assigns a terminal node to a pure strategy profile $s \in S_0 \times S_1$.*

Definition 3.1.6. (Nash Equilibrium) *Let $G = \langle A, H, Z, \chi, \rho, \sigma, u\rangle$ be an extensive game with perfect information. A pure strategy profile $s^* \in S$ is a* Nash equilibrium *if for each player $i \in \{0,1\}$:*

$$u_i(O(s_i^*, s_{1-i}^*)) \geq u_i(O(s_i, s_{1-i}^*)) \; \forall s_i \in S_i.$$

The concept of Nash equilibrium does not capture the sequential structure of an extensive game. Some Nash equilibria of an extensive game may lack rationality. Such non-plausible equilibria are called *'non-credible threats'*. Examples are given in the textbook by Osborne and Rubinstein [1, pp. 95–97]. This motivates the concept of subgame perfect equilibrium.

Definition 3.1.7. (Subgame) *Given an extensive game $G = \langle A, H, Z, \chi, \rho, \sigma, u\rangle$ with perfect information, the* subgame rooted at node *$h \in H$ is the restriction $G|_h$ to the node h and all successor nodes of h.*

Definition 3.1.8. (Subgame Perfect Equilibrium) *A* subgame perfect equilibrium (SPE) *of an extensive game with perfect information G is a strategy profile s^* such that:*

$$\forall\, i \in \{0,1\},\ \forall\, h \in H :\ \ s^*|_h \text{ is a NE in } G|_h.$$

The next lemma asserts the existence of a SPE in finite extensive games with perfect information.

Lemma 3.1.9. (Existence of Subgame Perfect Equilibrium) *Every 2-player finite extensive game with perfect information has a SPE.*

Proof. It is a special case for 2 players of *Kuhn's theorem*. For a proof see [1, p. 99]. □

A sample SPE can be computed with the *backward induction*. The description and examples of the backward induction are given in the textbook by Shoham and Leyton-Brown [5, p. 124].

3.2. Extensive Games with Imperfect Information

Definition 3.2.1. (Imperfect-Information Extensive Game) *A 2-player finite* extensive game with imperfect information *is a tuple* $G = \langle A, H, Z, \chi, \rho, \sigma, u, I \rangle$*, where:*

- $\langle A, H, Z, \chi, \rho, \sigma, u \rangle$ *is a 2-player finite extensive game with perfect information;*
- $I = (I_0, I_1)$*, where for player* $i \in \{0, 1\}$*,* $I_i := (I_{i,1}, \ldots, I_{i,k})$ *is an equivalence relation (*information partition*) on the action nodes* $\{h \in H \mid \rho(h) = i\}$ *of player* i*, such that if* $h, h' \in I_{i,j}$*, then* $\chi(h) = \chi(h')$*.*

The nodes in an information partition $I_{i,j}$ are indistinguishable for player i. Therefore we use $\chi(I_{i,j})$ to denote the set of actions available to player i at any node in an information partition $I_{i,j}$.

Example 3.2.2. *Figure 3.1 shows an imperfect-information extensive game corresponding to the prisoners' dilemma bimatrix game from Example 2.1.4. The actions of player* 0 *are denoted with capital letters.*

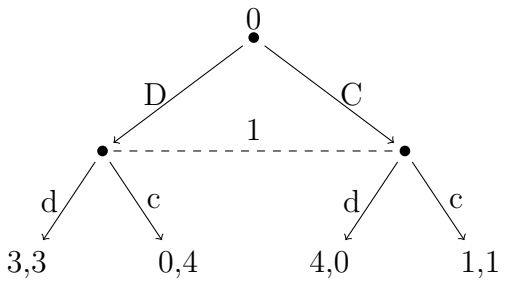


Figure 3.1.: The Prisoners' Dilemma as an Extensive Game with Imperfect Information.

Player 1 *has a non-trivial information set* $\{\langle D \rangle, \langle C \rangle\}$ *with two nodes coming after histories* $\langle D \rangle$ *and* $\langle C \rangle$*. He cannot distinguish between them and has the same set of actions available on each of them:* $\chi(\{\langle D \rangle, \langle C \rangle\}) = \{d, c\}$*.*

Note 3.2.3. *An extensive game is a perfect information game if and only if for each player* $i \in \{0, 1\}$ *every information partition contains only a single decision node of player* i*.*

Definition 3.2.4. (Perfect Recall) *An extensive game satisfies* perfect recall *if for each* $i \in \{0, 1\}$*, every information set* $I_{i,k} \in I_i$ *and each pair of nodes* $x, y \in I_{i,k}$*, player* i*'s actions from the root to* x *are exactly the same as his actions from the root to* y*.*

We only consider extensive games with perfect recall. In games with perfect recall each node has the unique history.

Definition 3.2.5. (Pure Strategy) *A* pure strategy s_i *of player* $i \in \{0,1\}$ *in* $G = \langle A, H, Z, \chi, \rho, \sigma, u, I\rangle$ *is a map assigning an action* $a \in \chi(I_{i,k})$ *to every information set* $I_{i,k} \in I_i$ *of player* i*. We denote player* i*'s pure strategy as a sequence of actions* $\langle a_K\rangle_{K\in I_i}$*. The actions are ordered by the breadth-first search. The sequential notion of pure strategy is similar to the notion of history but it should be clear from the context which one is meant. The set of player* i*'s pure strategies is denoted with* S_i*.*

Definition 3.2.6. (Mixed Strategy) *Let* $G = \langle A, H, Z, \chi, \rho, \sigma, u, I\rangle$ *be an imperfect-information extensive game. Player* i*'s* mixed strategy *is a probability distribution* α_i *over* S_i*. The set of player* i*'s mixed strategies is denoted with* $\Delta(S_i)$*.*

We consider so-called *behavioral strategies* in extensive games with imperfect information instead of mixed strategies. While a mixed strategy assigns a probability distribution over each pure strategy, a behavioral strategy assigns a probability distribution over the actions each information set in the game.

Definition 3.2.7. (Behavioral Strategy) *Let* $G = \langle A, H, Z, \chi, \rho, \sigma, u, I\rangle$ *be an extensive game with imperfect information. A* behavioral strategy *of player* $i \in \{0,1\}$ *is a map* b_i *which assigns to every information set* $I_{i,k}$ *a probability distribution* $b_i(I_{i,k})$ *over the set of actions* $\chi(I_{i,k})$*. We denote the set of player* i*'s behavioral strategies with* B_i*.*

Definition 3.2.8. (Outcome Equivalence) *Two strategies (behavioral or mixed) of player* $i \in \{0,1\}$ *are* outcome equivalent *if the probability distributions over the terminal nodes are equal, given the pure strategy of the opposite player* $1-i$*.*

Theorem 3.2.9. (Kuhn's Theorem) *In an extensive game with perfect recall, any mixed strategy has a corresponding behavioral strategy that is outcome equivalent; and vice versa.*

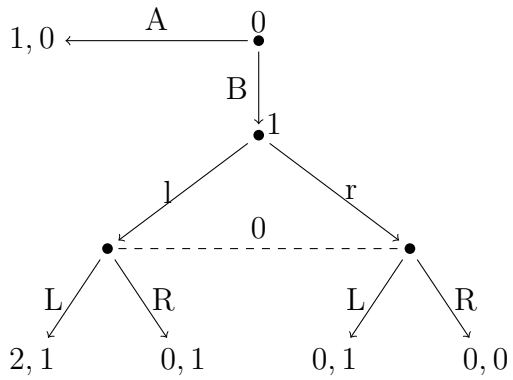
Proof. See [1], p.215. □

The next example demonstrates how to transform a behavioral strategy into a mixed strategy.

Example 3.2.10. (Behavioral Strategy → Mixed Strategy) *Given player* i*'s behavioral strategy* b_i*, we compute an outcome equivalent mixed strategy* α_i *by multiplying all probabilities assigned by* b_i *to the actions appearing in each of his information sets. Let* $s_i = \langle a_1, \ldots, a_k\rangle$ *be player* i*'s pure strategy such that* $a_l \in I_{i,l}$*. Then*

$$\alpha_i(s_i) := \prod_{j=1}^{k} b_i(a_j)$$

Consider the extensive game with simultaneous moves from the next figure.



Let b_0 and b_1 be the behavioral strategies such that

$$b_0(\{\langle\rangle\})(A) = \frac{1}{2}, \qquad b_0(\{\langle\rangle\})(B) = \frac{1}{2},$$
$$b_0(\{\langle B,l\rangle, \langle B,r\rangle\})(L) = \frac{2}{3}, \qquad b_0(\{\langle B,l\rangle, \langle B,r\rangle\})(R) = \frac{1}{3},$$
$$b_1(\{\langle B\rangle\})(l) = \frac{1}{3}, \qquad b_1(\{\langle B\rangle\})(r) = \frac{2}{3}.$$

The outcome equivalent mixed strategy of player 0 is α_0 with

$$\alpha_0(\langle A,L\rangle) = b_0(\{\langle\rangle\})(A) \cdot b_0(\{\langle B,l\rangle, \langle B,r\rangle\})(L) = \frac{1}{2} \cdot \frac{2}{3} = \frac{2}{6},$$
$$\alpha_0(\langle A,R\rangle) = b_0(\{\langle\rangle\})(A) \cdot b_0(\{\langle B,l\rangle, \langle B,r\rangle\})(R) = \frac{1}{2} \cdot \frac{1}{3} = \frac{1}{6},$$
$$\alpha_0(\langle B,L\rangle) = b_0(\{\langle\rangle\})(B) \cdot b_0(\{\langle B,l\rangle, \langle B,r\rangle\})(L) = \frac{1}{2} \cdot \frac{2}{3} = \frac{2}{6},$$
$$\alpha_0(\langle B,R\rangle) = b_0(\{\langle\rangle\})(B) \cdot b_0(\{\langle B,l\rangle, \langle B,r\rangle\})(R) = \frac{1}{2} \cdot \frac{1}{3} = \frac{1}{6}.$$

The outcome equivalent mixed strategy of player 1 is α_1 with

$$\alpha_1(\langle l\rangle) = b_1(\{\langle B\rangle\})(l) = \frac{1}{3},$$
$$\alpha_1(\langle r\rangle) = b_1(\{\langle B\rangle\})(r) = \frac{2}{3}.$$

Definition 3.2.11. (Outcome of Behavioral Strategy) *An* outcome *of a behavioral strategy profile b is a probability distribution $O(b)$ over the terminal histories. Let $\sigma = \langle a^1, \ldots, a^K\rangle \in \Sigma_Z$, then*

$$O(b)(\sigma) := \prod_{k=0}^{K-1} b_{\rho(\langle a^1,\ldots,a^k\rangle)}(\langle a^1, \ldots, a^k\rangle)(a^{k+1}),$$

where $\langle a^1, a^k\rangle = \langle\rangle$ is the initial history for $k = 0$.

Now we define expected payoff of a behavioral strategy profile which leads us to the concept of Nash equilibrium in extensive games with imperfect information.

Definition 3.2.12. (Expected Payoff) *Let $b = (b_0, b_1)$ be a behavioral strategy profile in an extensive game with imperfect information. Player i's* expected payoff $U_i(b)$ *is defined in the following way:*

$$U_i(b) := \sum_{\sigma \in \Sigma_Z} O(b)(\sigma) \cdot u_i(\sigma).$$

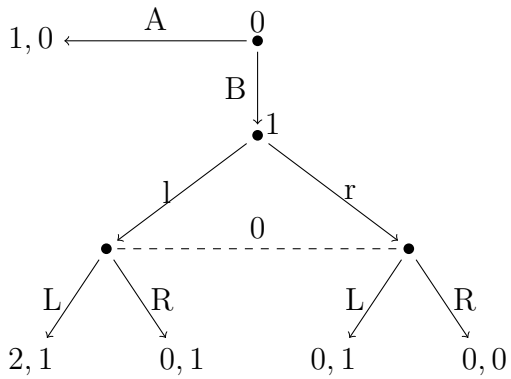
Definition 3.2.13. (Nash Equilibrium) *Let $G = \langle A, H, Z, \chi, \rho, \sigma, u, I \rangle$ be an extensive game with imperfect information. A behavioral strategy profile $b^* = (b_0^*, b_1^*)$ is called a* Nash equilibrium *if for every player $i \in \{0, 1\}$:*

$$U_i(b_i^*, b_{1-i}^*) \geq U_i(b_i, b_{1-i}^*) \; \forall \; b_i \in B_i.$$

Definition 3.2.14. (Corresponding Bimatrix Game) *There are several ways to transform an extensive game with imperfect information into a bimatrix game. The most natural way is to tabulate pure strategies of both players and record the resulting expected payoffs. We call this transformation the* corresponding normal-form (or bimatrix) game.

Example 3.2.15 illustrates the transformation from an extensive game with imperfect information into the corresponding bimatrix game.

Example 3.2.15. *The next figure shows the 2-player imperfect-information extensive game from Example 3.2.10 on the left and the corresponding bimatrix game on the right. Player* 0 *has 4 pure strategies: $\langle A, L \rangle$, $\langle A, R \rangle$, $\langle B, L \rangle$ and $\langle B, R \rangle$. Player* 1 *has two pure strategies: $\langle l \rangle$ and $\langle r \rangle$.*



	$\langle l \rangle$	$\langle r \rangle$
$\langle A, L \rangle$	$1, 0$	$1, 0$
$\langle A, R \rangle$	$1, 0$	$1, 0$
$\langle B, L \rangle$	$2, 1$	$0, 1$
$\langle B, R \rangle$	$0, 1$	$0, 0$

Figure 3.2.: An Imperfect-Information Extensive Game and the Corresponding Bimatrix Form.

Lemma 3.2.16. *Let $G = \langle A, H, Z, \chi, \rho, \sigma, u, I\rangle$ be an extensive game with imperfect information and $G' = \langle A'_0, A'_1, u'_0, u'_1\rangle$ the corresponding normal-form game. If $\alpha'^* = (\alpha'^*_0, \alpha'^*_1)$ is a Nash equilibrium in G', then there is a behavioral strategy profile $b^* = (b^*_0, b^*_1)$ which is a Nash equilibrium in G with expected payoff $U_i(b^*) = U'_i(\alpha'^*)$ for every player $i \in \{0, 1\}$.*

Proof. The game G' is constructed from G such that $A'_i = S_i$. Thus the mixed strategy α'^*_i is a probability distribution over the player i's pure strategies S_i and is also a mixed strategy in G by Definition 3.2.6. Player i's utility function u'_i is determined by u_i in the following way: Let $(a, b) \in A'_0 \times A'_1 = S_0 \times S_1$, then the pure strategy profile induces the unique terminal history $\sigma \in \Sigma_Z$, such that $u'_i(a, b) := u_i(\sigma)$. Thus α'^* is also a Nash equilibrium in G: $U_i(\alpha'^*_i, \alpha'^*_{1-i}) = U'_i(\alpha'^*_i, \alpha'^*_{1-i}) \geq U'_i(\alpha'_i, \alpha'^*_{1-i}) = U_i(\alpha'_i, \alpha'^*_{1-i})$ for all $\alpha'_i \in S_i$ and $i \in \{0, 1\}$. Now the statement follows directly from Kuhn's theorem (see Theorem 3.2.9): There is a behavioral strategy profile (b^*_0, b^*_1) in G which is outcome equivalent to the mixed strategy $(\alpha'^*_0, \alpha'^*_1)$. Therefore (b^*_0, b^*_1) is a Nash equilibrium in G with expected payoff $U_i(b^*) = U'_i(\alpha'^*)$ for every player $i \in \{0, 1\}$. □

Lemma 3.2.17. (Existence of Nash Equilibrium) *Every finite extensive game has a Nash equilibrium.*

Proof. Transform an extensive game G into the corresponding normal-form game G'. Lemma 2.1.11 provides the existence of a Nash equilibrium in G'. Thus there is also a Nash equilibrium in G (Lemma 3.2.16). □

Definition 3.2.18. (Subgame) *A* subgame *of an imperfect-information extensive game $G = \langle A, H, Z, \chi, \rho, \sigma, u, I\rangle$ rooted at a non-terminal node $h \in H$ is a subset $G|_h$ of G such that:*

1. *$\{h\} \in I_{\rho(h)}$ which means that $G|_h$ begins with an information set that contains only the node h.*
2. *$G|_h$ contains only h and all successor nodes of h.*
3. *$G|_h$ contains only complete information sets.*

For example, the game G from Example 3.2.15 has two subgames: the subgame which starts after the action B of player 0 and the whole game itself which starts at the root node $\emptyset$.

Although we can transform each extensive game with imperfect information into a corresponding normal-form game and then find a Nash equilibrium, it would be very inefficient, since the number of actions in the corresponding normal-form game is exponential in the size of the extensive game. The second reason not to do this is that the concept of Nash equilibrium does not capture the sequential structure of extensive games. We generalize the idea of subgame perfect equilibrium to the concept of *perfect Bayesian equilibrium* where each player's strategy is 'optimal' in each of his information sets. The optimality in an information set of a player depends on his belief about the

history which may have occurred. Therefore perfect Bayesian equilibrium does not only depend on the players' strategies but also on their beliefs about what has happened before.

Definition 3.2.19. (Belief System) *Let $G = \langle A, H, Z, \chi, \rho, \sigma, u, I\rangle$ be an extensive game with imperfect information. A* belief system *$\beta : I_0 \cup I_1 \to [0,1]$ is a map which assigns to every information set $I_{i,k} \in I_0 \cup I_1$ a probability distribution $\beta_{i,k} := \beta(I_{i,k})$ over the nodes in $I_{i,k}$.*

Definition 3.2.20. (Assessment) *Let $b = (b_0, b_1)$ be a behavioral strategy profile and β a belief system. An* assessment *is a pair (b, β).*

Notation 3.2.21. *Let $\mathbb{P}_b(I_{i,k})$ denote a probability that an information set $I_{i,k}$ of player $i \in \{0,1\}$ is reached, given a behavioral strategy profile b and $\mathbb{P}_b(h)$ denote a probability that node $h \in H \cup Z$ is reached, given b.*

Definition 3.2.22. (Bayesian Consistency)
An assessment (b, β) in an extensive game with imperfect information is Bayesian consistent *if for all $I_{i,k} \in I_0 \cup I_1$ with $\mathbb{P}_b(I_{i,k}) > 0$ and $h \in I_{i,k}$:*

$$\beta_{i,k}(h) = \frac{\mathbb{P}_b(h)}{\mathbb{P}_b(I_{i,k})}$$

We adjust the term of outcome to the concept of assessment since it also depends on players' beliefs.

Definition 3.2.23. (Conditional Outcome of Behavioral Strategy) *Let (b, β) be an assessment. We define the* outcome *$O(b, \beta \,|\, I_{i,k})$* of *$(b, \beta)$* conditional on *$I_{i,k} \in I_i$ as a probability distribution $O(b, \beta \,|I_{i,k}) : \Sigma_Z \to [0,1]$ over the terminal histories determined by b* and *β, conditional on $I_{i,k}$ being reached, in the following way: let $\sigma^* = \langle a^1, \ldots, a^K\rangle \in \Sigma_Z$ be a terminal history. Then there are two cases:*

1. *If there is no subhistory of σ^* which leads to $I_{i,k}$, then*

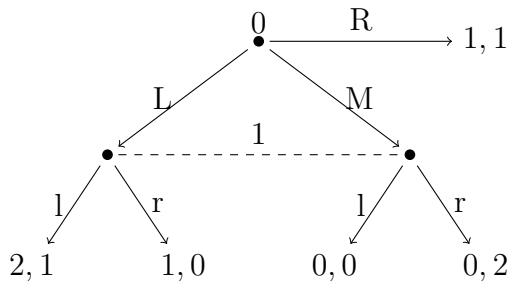
   $$O(b, \beta \,|\, I_{i,k})(\sigma^*) = 0.$$

2. *If there is a subhistory $\sigma = \langle a^1, \ldots, a^L\rangle \in I$ of σ^* which leads to $I_{i,k}$ with $L < K$, then*

   $$O(b, \beta \,|\, I_{i,k})(\sigma^*) = \beta(I_{i,k})(\sigma) \prod_{k=L}^{K-1} b_{\rho(\langle a^1, \ldots, a^k\rangle)}(\langle a^1, \ldots, a^k\rangle)(a^{k+1}).$$

Since we only consider games with perfect recall, there is only one unique subhistory of σ^* which lies in the information set $I_{i,k}$.

Example 3.2.24. *We illustrate the concept of conditional outcome on the following extensive game with imperfect information.*



Let us determine the behavioral strategies in the following way:

$$b_0(\{\langle\rangle\})(L) = \frac{2}{3}, \quad b_1(\{\langle L\rangle, \langle M\rangle\})(l) = \frac{1}{2},$$
$$b_0(\{\langle\rangle\})(M) = \frac{1}{6}, \quad b_1(\{\langle L\rangle, \langle M\rangle\})(r) = \frac{1}{2}.$$
$$b_0(\{\langle\rangle\})(R) = \frac{1}{6}.$$

We compute the Bayesian consistent beliefs with Bayes rule:

$$\beta(\{\langle L\rangle, \langle M\rangle\})(\langle L\rangle) = \frac{b_0(\{\langle\rangle\})(L)}{b_0(\{\langle\rangle\})(L) + b_0(\{\langle\rangle\})(M)} = \frac{\frac{2}{3}}{\frac{2}{3} + \frac{1}{6}} = \frac{4}{5},$$
$$\beta(\{\langle L\rangle, \langle M\rangle\})(\langle M\rangle) = \frac{b_0(\{\langle\rangle\})(M)}{b_0(\{\langle\rangle\})(L) + b_0(\{\langle\rangle\})(M)} = \frac{\frac{1}{6}}{\frac{2}{3} + \frac{1}{6}} = \frac{1}{5}.$$

Now we compute the outcome $O(b, \beta \,|\, \{\langle L\rangle, \langle M\rangle\})$ for the terminal histories $\langle L, l\rangle$, $\langle L, r\rangle$, $\langle M, l\rangle$, $\langle M, r\rangle$ and $\langle R\rangle$, conditional being in the information set $\{\langle L\rangle, \langle M\rangle\}$:

$$O(b, \beta \,|\, \{\langle L\rangle, \langle M\rangle\})(\langle R\rangle) = 0,$$

$$O(b, \beta \,|\, \{\langle L\rangle, \langle M\rangle\})(\langle L, l\rangle) = \beta(\{\langle L\rangle, \langle M\rangle\})(\langle L\rangle) \cdot b_1(\{\langle L\rangle, \langle M\rangle\})(l) = \frac{4}{5} \cdot \frac{1}{2} = \frac{2}{5},$$

$$O(b, \beta \,|\, \{\langle L\rangle, \langle M\rangle\})(\langle L, r\rangle) = \beta(\{\langle L\rangle, \langle M\rangle\})(\langle L\rangle) \cdot b_1(\{\langle L\rangle, \langle M\rangle\})(r) = \frac{4}{5} \cdot \frac{1}{2} = \frac{2}{5},$$

$$O(b, \beta \,|\, \{\langle L\rangle, \langle M\rangle\})(\langle M, l\rangle) = \beta(\{\langle L\rangle, \langle M\rangle\})(\langle M\rangle) \cdot b_1(\{\langle L\rangle, \langle M\rangle\})(l) = \frac{1}{5} \cdot \frac{1}{2} = \frac{1}{10},$$

$$O(b, \beta | \{\langle L\rangle, \langle M\rangle\})(\langle M, r\rangle) = \beta(\{\langle L\rangle, \langle M\rangle\})(\langle M\rangle) \cdot b_1(\{\langle L\rangle, \langle M\rangle\})(r) = \frac{1}{5} \cdot \frac{1}{2} = \frac{1}{10}.$$

Definition 3.2.25. (Conditional Expected Payoff of Behavioral Strategy)
Let $G = \langle A, H, Z, \chi, \rho, \sigma, u, I\rangle$ be an imperfect-information extensive game and (b, β) an assessment. Player i's expected payoff $U_i(b, \beta \mid I_{i,k})$ conditional being in an information set $I_{i,k}$ *is defined as follows:*

$$U_i(b, \beta \mid I_{i,k}) := \sum_{\sigma \in \Sigma_Z} O(b, \beta \mid I_{i,k})(\sigma) \cdot u_i(\sigma).$$

Example 3.2.26. *We continue Example 3.2.24 and compute the expected payoffs for both players conditional being in the information set $I_1 := \{\langle L\rangle, \langle M\rangle\}$.*

$$U_0(b, \beta \mid I_1) = \sum_{\sigma \in \Sigma_Z} O(b, \beta \mid I_1)(\sigma) \cdot u_0(\sigma) = \frac{2}{5} \cdot 2 + \frac{2}{5} \cdot 1 + \frac{1}{10} \cdot 0 + \frac{1}{10} \cdot 0 + 0 \cdot 1 = \frac{6}{5}.$$

$$U_1(b, \beta \mid I_1) = \sum_{\sigma \in \Sigma_Z} O(b, \beta \mid I_1)(\sigma) \cdot u_1(\sigma) = \frac{2}{5} \cdot 1 + \frac{2}{5} \cdot 0 + \frac{1}{10} \cdot 0 + \frac{1}{10} \cdot 2 + 0 \cdot 1 = \frac{3}{5}.$$

Definition 3.2.27. (Sequential Rationality)
An assessment (b^*, β^*) is sequentially rational *if for each player $i \in \{0, 1\}$ and every information set $I_{i,k} \in I_i$:*

$$U_i((b_i^*, b_{1-i}^*), \beta^* \mid I_{i,k}) \geq U_i((b_i, b_{1-i}^*), \beta^* \mid I_{i,k}) \; \forall \; b_i \in B_i.$$

Example 3.2.28. *In Example 3.2.24 and Example 3.2.26 the behavioral strategy profile b is not sequentially rational since players' expected payoffs $U_i(b, \beta \mid I)$ is not maximized in every information set $I \in I_0 \cup I_1$ for $i \in \{0, 1\}$. Since player 0's action M is strictly dominated by the other actions, he would never play it with positive probability. Given the player 1's behavioral strategy b_1 and his belief system β, player 0 achieves maximal payoff in the initial information set $\{\langle\rangle\}$ if he plays action L with the probability α, action M with the probability 0 and action R with the probability $(1 - \alpha)$ and maximizes over his expected payoff $U_0(b, \beta \mid \{\langle\rangle\})$ conditional being in the initial information set $\{\langle\rangle\}$ and given (b_1, β):*

$$b_0(\{\langle\rangle\})(L) = \alpha = \arg\max_{\alpha \in [0,1]} \{\alpha \cdot \frac{1}{2} \cdot 2 + \alpha \cdot \frac{1}{2} + (1 - \alpha) \cdot 1\} = 1.$$

Thus the behavioral strategy of player 0 is determined in the following way:

$$b_0(\{\langle\rangle\})(L) = 1, \; b_0(\{\langle\rangle\})(M) = 0 \text{ and } b_0(\{\langle\rangle\})(R) = 0.$$

Player 1 would have to 'update' his beliefs in order to keep them Bayesian consistent with the new behavioral strategy b_0 of player 0. This is called 'Bayesian updating'.

$$\beta(\{\langle L\rangle, \langle M\rangle\})(\langle L\rangle) = 1 \text{ and } \beta(\{\langle L\rangle, \langle M\rangle\})(\langle M\rangle) = 0.$$

Given his new beliefs and player 0*'s strategy* b_0*, player* 1 *would achieve the maximal expected utility if he plays action* l *with probability* $b_1(\{\langle L\rangle, \langle M\rangle\})(l) = 1$ *and* r *with probability* 0*. After this 'sequential computation' we would get a sequentially rational and Bayesian consistent assessment with expected payoff* $(2, 1)$*. Such an assessment is called* 'perfect Bayesian equilibrium'.

The concept of perfect Bayesian equilibrium is a natural generalization of subgame perfect equilibrium.

Definition 3.2.29. (Perfect Bayesian Equilibrium) *An assessment* (b, β) *in an extensive game with imperfect information is a* perfect Bayesian equilibrium (PBE) *if*

1. (b, β) *is sequentially rational.*
2. *The beliefs* β *are Bayesian consistent.*
3. β *is always updated according to Bayes rule 'when applicable'.*

The third requirement 'when applicable' means that the beliefs are determined in order by equilibrium strategies.

A. Perea defines in the textbook [11] a more practical concept for sequential rationality: Only those actions should be played with positive probability which are a local best response to the behavioral strategy of the opposite player. He calls it *'local sequential rationality'*.

Definition 3.2.30. (Local Expected Payoff) *Player* i*'s* local expected payoff $U_i(a \,|\, b, h)$ *from playing action* a*,* conditional on being in a non-terminal node $h \in H$ *and given behavioral strategy profile* $b = (b_0, b_1)$ *is defined in the following way: Let* $\langle a^1, \dots a^L\rangle \in \Sigma_H$ *be a non-terminal history which leads from the root node to the node* h*, then*

$$U_i(a \,|\, b, h) :=$$

$$\sum_{\langle a^1,\dots,a^L,a,c^1,\dots,c^K\rangle \in \Sigma_Z} u_i(\langle a^1, \dots, a^L, a, c^1, \dots, c^K\rangle) \cdot \prod_{j=1}^{K-1} b_{\rho(\langle a^1,\dots,a^L,a,c^1,\dots,c^j\rangle)}(c^{j+1}).$$

Player i*'s (local) expected payoff from action* a*,* conditional on being at information set $I_{i,k} \in I_i$ *and given behavioral strategy profile* $b = (b_0, b_1)$ *is then defined with:*

$$U_i(a \,|\, b, I_{i,k}) := \sum_{h \in I_{i,k}} \beta_{i,k}(a) \cdot U_i(a \,|\, b, h).$$

Definition 3.2.31. (Local Best Response) *Let* (b, β) *be an assessment with* $b = (b_0, b_1)$*. An action* $a \in \chi(I_{i,k})$ *is called* a local best response *of player* i *to the behavioral strategy* b_{1-i} *of the opposite player* $1 - i$ *in the information set* $I_{i,k}$ *if*

$$U_i(a \,|\, b, I_{i,k}) \geq U_i(a' \,|\, b, I_{i,k}) \;\forall\; a' \in \chi(I_{i,k}).$$

Definition 3.2.32. (Local Sequential Rationality) *An assessment (b, β) with $b = (b_0, b_1)$ is called* locally sequentially rational *if for every player $i \in \{0, 1\}$, every information set $I_{i,k} \in I_i$ and each action $a \in \chi(I_{i,k})$ we have*

$$b_i(h)(a) > 0 \Rightarrow a \text{ is a local best response to } b_{1-i} \text{ in } I_{i,k}.$$

The difference between local sequential rationality (Definition 3.2.32) and sequential rationality (Definition 3.2.27) is that local sequential rationality requires players to believe, that the opponent player chooses the optimal action instead of the optimal strategy in every information set. A. Perea shows in the textbook [11] that local sequential rationality and sequential rationality are equivalent in extensive games with perfect information. In extensive games with imperfect information sequential rationality implies local sequential rationality but not the other way around.

Example 3.2.33. *In this example we compute a locally sequentially rational assessment in the game from Example 3.2.24. Let $b = (b_0, b_1)$ be a behavioral strategy profile with:*

$$\begin{aligned} b_0(\{\langle\rangle\})(L) = \alpha_1, \quad & b_1(\{\langle L\rangle, \langle M\rangle\})(l) = \gamma, \\ b_0(\{\langle\rangle\})(M) = \alpha_2, \quad & b_1(\{\langle L\rangle, \langle M\rangle\})(r) = 1 - \gamma. \\ b_0(\{\langle\rangle\})(R) = 1 - \alpha_1 - \alpha_2. & \end{aligned}$$

We compute Bayesian consistent beliefs with Bayes rule:

$$\begin{aligned} \beta(\{\langle L\rangle, \langle M\rangle\})(\langle L\rangle) &= \frac{b_0(\{\langle\rangle\})(L)}{b_0(\{\langle\rangle\})(L) + b_0(\{\langle\rangle\})(M)} = \frac{\alpha_1}{\alpha_1 + \alpha_2}, \\ \beta(\{\langle L\rangle, \langle M\rangle\})(\langle M\rangle) &= \frac{b_0(\{\langle\rangle\})(M)}{b_0(\{\langle\rangle\})(L) + b_0(\{\langle\rangle\})(M)} = \frac{\alpha_2}{\alpha_1 + \alpha_2}. \end{aligned}$$

Now let us compute the local expected payoffs. The expected payoffs of player 0 from playing actions L, M and R conditional being in the initial information set $\{\langle\rangle\}$ and given the assessment (b, β) are:

$$\begin{aligned} U_0(L \,|\, b, \{\langle\rangle\}) &= \gamma \cdot 2 + (1 - \gamma) \cdot 1 = \gamma + 1, \\ U_0(M \,|\, b, \{\langle\rangle\}) &= \gamma \cdot 0 + (1 - \gamma) \cdot 0 = 0, \\ U_0(R \,|\, b, \{\langle\rangle\}) &= 1. \end{aligned}$$

The expected payoffs of player 1 from playing actions l and r conditional being in the information set $\{\langle L\rangle, \langle M\rangle\}$ and given the assessment (b, β) are:

$$\begin{aligned} U_1(l \,|\, b, \{\langle L\rangle, \langle M\rangle\}) &= \beta(\{\langle L\rangle, \langle M\rangle\})(\langle L\rangle) \cdot 1 + \beta(\{\langle L\rangle, \langle M\rangle\})(\langle M\rangle) \cdot 0 = \frac{\alpha_1}{\alpha_1 + \alpha_2}, \\ U_1(r \,|\, b, \{\langle L\rangle, \langle M\rangle\}) &= \beta(\{\langle L\rangle, \langle M\rangle\})(\langle L\rangle) \cdot 0 + \beta(\{\langle L\rangle, \langle M\rangle\})(\langle M\rangle) \cdot 2 = \frac{2\alpha_2}{\alpha_1 + \alpha_2}. \end{aligned}$$

Local sequential rationality requires that $\alpha_2 = 0$ *since*

$$0 = U_0(M|b, \{\langle\rangle\}) < U_0(L|b, \{\langle\rangle\}) = \gamma + 1,$$
$$0 = U_0(M|b, \{\langle\rangle\}) < U_0(R|b, \{\langle\rangle\}) = 1.$$

Thus the beliefs are determined by $\beta(\{\langle L\rangle, \langle M\rangle\})(\langle L\rangle) = 1$ *and* $\beta(\{\langle L\rangle, \langle M\rangle\})(\langle M\rangle) = 0$. *Now we know player* 1*'s local expected payoffs:*

$$U_1(l|b, \{\langle L\rangle, \langle M\rangle\}) = 1,$$
$$U_1(r|b, \{\langle L\rangle, \langle M\rangle\}) = 0.$$

From the local conditional rationality follows that $\gamma = 1$ *and thus* $\alpha_1 = 1$.

Definition 3.2.34. (Consistency)
An assessment (b, β) *in an extensive game with imperfect information is* consistent *if there exist a sequence* $(b^m, \beta^m)_{m\in\mathbb{N}}$ *of Bayesian consistent assessments with each* b^m *completely mixed and* $\lim_{m\to\infty}(b^m, \beta^m) = (b, \beta)$.

Definition 3.2.35. (Sequential Equilibrium) *An assessment* (b, β) *in an extensive game with imperfect information is a* sequential equilibrium *if it is sequentially rational and consistent.*

Since player i's local expected payoff from playing an action (Definition 3.2.30) is not linear, we cannot transfer the MIP formulation 2.14–2.20 from normal-form games to extensive games with imperfect information. D. Koller, N. Megiddo and B. Stengel introduced the *sequence form* for extensive 2-player games in their book [12]. They also derived a linear complementary program from the sequence form and showed that its solution is a sequential equilibrium. A solution of the LCP can be found by Lemke's complementary pivoting algorithm (see the brief description in [12, pp. 251-254]).

4. Extensive Games with Simultaneous Moves

A special class of extensive games with imperfect information consists of extensive games with simultaneous moves. In this section we introduce an efficient data structure for representing extensive games with simultaneous moves and algorithms for finding a sample equilibrium. The main idea is to use the bimatrix-game form to simulate the simultaneity and the extensive-game tree structure to express the multi-move situations. Our solution methods for finding a sample equilibrium in extensive games with simultaneous moves unify the theory of imperfect-information extensive games with the algorithmic results of normal-form games.

4.1. Preliminaries

Definition 4.1.1. (Extensive Games with Simultaneous Moves) *A 2-player finite* extensive game with simultaneous moves *is a tuple* $\Gamma = \langle H, (A_h^0)_{h\in H}, (A_h^1)_{h\in H}, (E_h)_{h\in H}, (T_h)_{h\in H}, \sigma, u\rangle$ *where for player* $i \in \{0,1\}$*:*

- $H \neq \emptyset$ *is a finite set of* nodes*;*
- A_h^i *is a finite set of* actions *available for player* i *in node* $h \in H$*;*
- $E_h \subseteq A_h^0 \times A_h^1$ *is the set of* non-terminal entries *in node* $h \in H$*;*
- $T_h := (A_h^0 \times A_h^1)\backslash E_h$ *is the set of* terminal entries *in node* $h \in H$*;*
- $\sigma : \bigcup_{h\in H} E_h \to H$ *is the* successor function *which* bijectively *maps each non-terminal entry* $(a,b) \in E_h$ *to a child node* $\sigma(a,b)$*. Node* h *is called the* parent node*, and* (a,b) *the* parent entry *of node* $\sigma(a,b)$*. A node* h *is called the* root node*, if it does not have a parent node. There is only one root node in* H*. A node is called* non-terminal*, if it has at least one non-terminal entry. Otherwise it is called* terminal*. The set of terminal nodes is denoted with* H_T*.*
- $u := (u_0, u_1)$ *where* $u_i : \bigcup_{h\in H} T_h \to \mathbb{R}$ *is called the* utility function *for player* i*.*

Note 4.1.2. *The requirements that* σ *is bijectiv and that the root node does not have a parent imply that an extensive game with simultaneous moves is an acyclic tree.*

Each terminal node represents a 2-player bimatrix game as defined in Definition 2.1.2. Example 4.1.3 illustrates *'Bach or Stravinsky Game* with a Book option'.

Example 4.1.3. *Player* 0 *chooses first if he wants to attend a concert (C) or if he prefers to stay home and read a book (R). If he chooses the concert option, then player* 0 *and player* 1 *simultaneously choose between Bach (B) and Stravinsky (S) concerts. Player* 0 *prefers Bach to Stravinsky and player* 1 *the other way around. But they both prefer to stay together. The game can be represented as an extensive game with simultaneous moves:*

$\emptyset$	0
R	1, 1
C	$C0$

$\rightarrow C0$	B	S
B	2, 1	0, 0
S	0, 0	1, 2

Figure 4.1.: Bach or Stravinsky with a Book Option

Definition 4.1.4. (Pure Strategy)
Let $\Gamma = \langle H, (A_h^0)_{h\in H}, (A_h^1)_{h\in H}, (E_h)_{h\in H}, (T_h)_{h\in H}, \sigma, u\rangle$ *be a 2-player extensive game with simultaneous moves. Player i's* pure strategy $s_i : H \to \bigcup_{h\in H} A_h^i$ *is a map assigning an action* $a \in A_h^i$ *to every node* $h \in H$*. We denote the set of player i's pure strategies with* S_i*.*

Definition 4.1.5. (Behavioral Strategy)
Let $\Gamma = \langle H, (A_h^0)_{h\in H}, (A_h^1)_{h\in H}, (E_h)_{h\in H}, (T_h)_{h\in H}, \sigma, u\rangle$ *be a 2-player extensive game with simultaneous moves. A* behavioral strategy *of player* $i \in \{0, 1\}$ *is a map* b_i *which assigns to every node* $h \in H$ *a probability distribution* $b_i(h)$ *over the set of actions* A_h^i*. We denote the set of player i's behavioral strategies with* B_i*.*

Definition 4.1.6. (History) *A* history $\gamma = \langle (a^0, b^0), (a^1, b^1), \ldots, (a^K, b^K)\rangle$ *is a sequence of entries, such that* $(a^k, b^k) \in E_{\sigma(a^{k-1},b^{k-1})}$ *for all* $k \in \{1, \ldots, K-1\}$ *and* $(a^K, b^K) \in E_{\sigma(a^{K-1},b^{K-1})} \cup T_{\sigma(a^{K-1},b^{K-1})}$*. If* $(a^K, b^K) \in E_{\sigma(a^{K-1},b^{K-1})}$*, then* γ *is called* non-terminal history*. If* $(a^K, b^K) \in T_{\sigma(a^{K-1},b^{K-1})}$*, then* γ *is called* terminal history*. We denote the set of all terminal histories with* Σ_T*. We use the notion* $u_i(\gamma) := u_i(a^K, b^K)$ *to denote the player i's payoff in the terminal history* $\gamma = \langle (a^0, b^0), (a^1, b^1), \ldots, (a^K, b^K)\rangle \in \Sigma_T$*.*

Definition 4.1.7. (Outcome) *An* outcome $O(b) : \Sigma_T \to [0, 1]$ *of a behavioral strategy b is a probability distribution over the terminal histories.*
Let $\gamma = \langle (a^0, b^0), (a^1, b^1), \ldots, (a^K, b^K)\rangle \in \Sigma_T$*, then:*

$$O(b)(\gamma) = \prod_{k=0}^{K-1} b_0(\langle (a^1, b^1), \ldots, (a^k, b^k)\rangle)(a^{k+1}) \cdot b_1(\langle (a^1, b^1), \ldots, (a^k, b^k)\rangle)(b^{k+1}).$$

Where $\langle (a^1, b^1), (a^k, b^k)\rangle = \langle\rangle$ *is the initial history for* $k = 0$*.*

Definition 4.1.8. (Expected Payoff) *Assume* $b \in B_0 \times B_1$*. Player i's* expected payoff *$U_i(b)$ is defined analogously to Definition 3.2.12:*

$$U_i(b) := \sum_{\gamma\in\Sigma_T} O(b)(\gamma) \cdot u_i(\gamma).$$

The class of extensive games with simultaneous moves is easier to handle than general extensive games with imperfect information since it is not necessary to define a belief system. We show this in Lemma 4.1.15.

Definition 4.1.9. (Corresponding Imperfect-Information Form) *There is a natural way to transform an extensive game with simultaneous moves into an extensive game with imperfect information. We call this transformation* 'corresponding imperfect-information form'. *Let* $\Gamma = \langle H, (A_h^0)_{h\in H}, (A_h^1)_{h\in H}, (E_h)_{h\in H}, (T_h)_{h\in H}, \sigma, u\rangle$ *be a 2-player extensive game with simultaneous moves. We construct the corresponding extensive game with imperfect information* $G' = \langle A', H', Z', \chi', \rho', \sigma', u', I'\rangle$ *as follows.*

- $A'^i := \bigcup_{h\in H} A_h^i$.
- *Since* G' *is an ordered game tree where each node is either a non-terminal node of player* i *or a terminal node, we need to determine the order in which the actions are taken. Without loss of generality let player* 0*'s actions* A_h^0 *be the first ones in* G'*. To carry the simultaneity of each node* $h \in H$ *into* G'*, the actions* A_h^0 *must lay in the same information set of player* 1*. Thus the non-terminal nodes of* G' *are* $H' := (\bigcup_{h\in H} A_h^0) \cup H$*. Figure 4.2 shows the transformation of node* $h \in H$ *with* $A_h^0 = \{a^1, \ldots, a^m\}$ *and* $A_h^1 = \{b^1, \ldots, b^n\}$ *into a part of the game tree* G'*.*

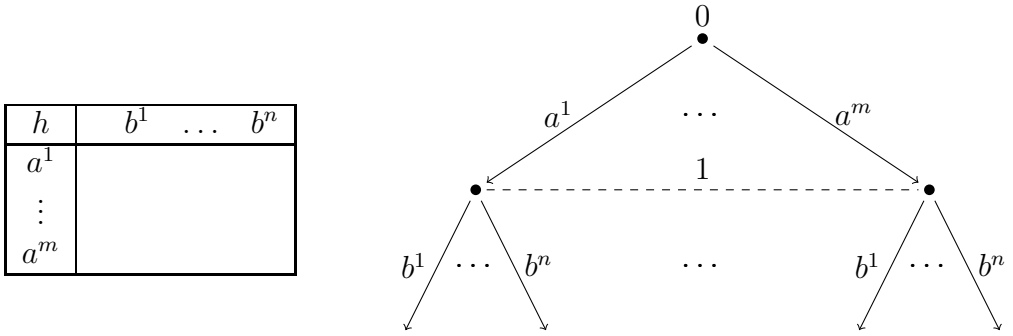


Figure 4.2.: Transformation from $h \in H$ into a part of G'

- *The terminal nodes of* G' *are equivalent to the terminal entries of* Γ*:* $Z' := \bigcup_{h\in H} T_h$.
- *Define the action function* $\chi' : H' \to 2^{A'}$ *with* $\chi'(h') := \begin{cases} A_{h'}^0, & \text{if } h' \in H \\ A_h^1, & \text{otherwise.} \end{cases}$
- *The player function* $\rho' : H' \to \{0, 1\}$ *is defined by:* $\rho'(h') := \begin{cases} 0, & \text{if } h' \in H \\ 1, & \text{otherwise.} \end{cases}$

- *Use σ to define the successor function $\sigma' : H' \times A' \to H' \cup Z'$ in G':*
  $$\sigma'(h', a') := \begin{cases} a', & \text{if } h' \in H. \\ \sigma(h', a'), & \text{otherwise.} \end{cases}$$
- $u'_i := u_i$.
- *The information sets of player 0 are always singletons $I'_0 := \{\{h\} | h \in H\}$ and the information sets of player 1 are exactly the action sets of player 0 available at nodes $h \in H$: $I'_1 := \{A^0_h | h \in H\}$.*

Example 4.1.10. *Consider the BoS game with a Book option as given in Example 4.1.3. Our recipe from Definition 4.1.9 would yield the following game with imperfect information:*

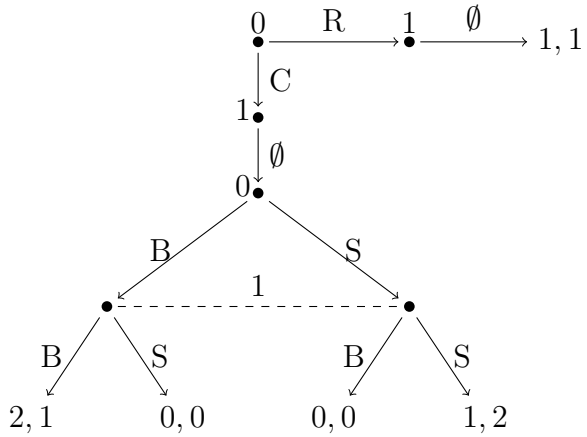


Figure 4.3.: BoS With a Book Option as an Imperfect-Information Extensive Game.

Lemma 4.1.11. *Let $\Gamma = \langle H, (A^0_h)_{h\in H}, (A^1_h)_{h\in H}, (E_h)_{h\in H}, (T_h)_{h\in H}, \sigma, u\rangle$ be a 2-player extensive game with simultaneous moves and $G' = \langle A', H', Z', \chi', \rho', \sigma', u', I'\rangle$ the corresponding imperfect-information extensive game. Then the number of nodes of G' is polynomially greater than the number of nodes of Γ.*

Proof. We consider the worst case assuming that every node $h \in H$ has a constant number of actions for both players:

$$|A^i_h| = a > 1 \; \forall \, h \in H, \, i \in \{0, 1\}.$$

We also assume that the depth of Γ is constantly $d > 1$, which means that all terminal histories must have the length d. Then Γ has $n := |H| = \frac{(a^2)^{d+1}-1}{a^2-1}$ nodes since the number of nodes $|H|$ is the sum of the first d terms of the following geometric series:

$$n = \sum_{i=0}^{d} (a^2)^i = \frac{(a^2)^{d+1} - 1}{a^2 - 1}.$$

The number of non-terminal nodes of G' is

$$|H'| = |H| + |\bigcup_{h \in H} A_h^0| = n + a \cdot n = n \cdot (a+1).$$

The number of terminal nodes of G' is equal the number of the terminal entries in Γ:

$$|Z'| = |\bigcup_{h \in H} T_h| = \underbrace{(a^2)^d}_{|H_T|} \cdot a^2 = (a^2)^{d+1}.$$

Therefore the number of all nodes of G' is then:

$$\begin{aligned} |H' \cup Z'| &= n \cdot (a+1) + (a^2)^{d+1} \\ &= \frac{(a^2)^{d+1} - 1}{a^2 - 1} \cdot (a+1) + (a^2)^{d+1} \\ &= \frac{(a^2)^{d+1} - 1}{a - 1} + (a^2)^{d+1} \\ &= \frac{a^{2d+3} - 1}{a - 1} \end{aligned}$$

Thus G' has approximately $(a^2 + a)$ times more nodes than Γ, since:

$$\begin{aligned} \frac{|H' \cup Z'|}{|H|} &= \frac{a^{2d+3} - 1}{a - 1} \cdot \frac{a^2 - 1}{a^{2d+2} - 1} \\ &= \frac{(a^{2d+3} - 1) \cdot (a+1)}{(a^{2d+2} - 1)} \\ &\approx a \cdot (a+1) = a^2 + a. \end{aligned}$$

□

Definition 4.1.12. (Corresponding Histories)
Let $\Gamma = \langle H, (A_h^0)_{h \in H}, (A_h^1)_{h \in H}, (E_h)_{h \in H}, (T_h)_{h \in H}, \sigma, u \rangle$ be a 2-player extensive game with simultaneous moves and $G' = \langle A', H', Z', \chi', \rho', \sigma', u', I' \rangle$ the corresponding imperfect-information extensive game. If $\gamma = \langle (a^0, b^0), (a^1, b^1), \ldots, (a^K, b^K) \rangle$ is a history in Γ, then $\gamma' := \langle a^0, b^0, a^1, b^1, \ldots, a^K, b^K \rangle$ is called the corresponding history *in G'.*

Definition 4.1.13. (Outcome Equivalence)
Let $\Gamma = \langle H, (A_h^0)_{h \in H}, (A_h^1)_{h \in H}, (E_h)_{h \in H}, (T_h)_{h \in H}, \sigma, u \rangle$ be a 2-player extensive game with simultaneous moves and $G' = \langle A', H', Z', \chi', \rho', \sigma', u', I' \rangle$ the corresponding imperfect-information extensive game. Let B_i denote the set of player i's behavioral strategies in Γ and B_i' the set of player i's behavioral strategies in G' for $i \in \{0, 1\}$. Two behavioral strategies $b \in B_0 \times B_1$ and $b' \in B_0' \times B_1'$ are called outcome equivalent *if the outcomes of b and b' are equal for every terminal history γ and its corresponding terminal history γ': $O(b)(\gamma) = O(b')(\gamma')$.*

Now we take a closer look at the behavioral strategies and beliefs in the corresponding imperfect-information form.

Lemma 4.1.14. *Let $\Gamma = \langle H, (A_h^0)_{h\in H}, (A_h^1)_{h\in H}, (E_h)_{h\in H}, (T_h)_{h\in H}, \sigma, u\rangle$ be a 2-player extensive game with simultaneous moves and $G' = \langle A', H', Z', \chi', \rho', \sigma', u', I'\rangle$ the corresponding imperfect-information extensive game. Let b_i be player i's behavioral strategy in Γ. Then there is an outcome equivalent behavioral strategy b_i' in G'. We call b_i' the* corresponding behavioral strategy *of b_i.*

Proof. Since $I_0' = \{\{h\} \mid h \in H\}$, we define the corresponding player 0's behavioral strategy b_0' of G in an information set $I_{0,k}' = \{h\}$ with $b_0'(I_{0,k}')(a) := b_0(h)(a)$ for an action $a \in \chi'(I_{0,k}') = A_h^0$. Since $I_1' = \{A_h^0 \mid h \in H\}$, there is a node h^* for every information set $I_{1,k}'$ such that $I_{1,k}' = A_{h^*}^0$. The corresponding player 1's behavioral strategy b_1' of G is then defined with $b_1'(I_{1,k}')(a) := b_1(h^*)(a)$ for an action $a \in \chi'(I_{1,k}') = A_{h^*}^1$. Let $\gamma = \langle (c^0, d^0), \ldots, (c^K, d^K)\rangle \in \Sigma_T$ be a terminal history in Γ which leads from the root node to the terminal entry $(c^K, d^K) \in T_{\sigma(c^{K-1}, d^{K-1})}$ with payoff $u(c^K, d^K)$. Then $\gamma' = \langle c^0, d^0, \ldots, c^K, d^K\rangle \in \Sigma_Z$ is the corresponding terminal history in G'. Thus we have the following equality:

$$\begin{aligned} O(b')(\gamma') &= \prod_{k=0}^{K-1} b_0'(\langle c^1, d^1, \ldots, c^k, d^k\rangle)(c^{k+1}) \cdot b_1'(\langle c^1, d^1, \ldots, c^k, d^k\rangle)(d^{k+1}) \\ &= \prod_{k=0}^{K-1} b_0(\langle (c^1, d^1), \ldots, (c^k, d^k)\rangle)(c^{k+1}) \cdot b_1(\langle (c^1, d^1), \ldots, (c^k, d^k)\rangle)(d^{k+1}) \\ &= O(b)(\gamma). \end{aligned}$$

□

Lemma 4.1.15. *In the corresponding imperfect-information extensive game $G' = \langle A', H', Z', \chi', \rho', \sigma', u', I'\rangle$ of an extensive game with simultaneous moves $\Gamma = \langle H, (A_h^0)_{h\in H}, (A_h^1)_{h\in H}, (E_h)_{h\in H}, (T_h)_{h\in H}, \sigma, u\rangle$ the Bayesian consistent beliefs are always either trivial or equal to the behavioral strategy of the opposite player.*

Proof. Since $I_0' = \{\{h\} \mid h \in H\}$, the information sets of player 0 are singletons and his beliefs are always 1. The belief system of player 1 is equal to the player 0's behavioral strategy since $I_1' = \{A_h^0 \mid h \in H\} \Rightarrow \beta(A_h^0)(a) = \frac{b_0(a)}{1} = b_0(a)$. □

Definition 4.1.16. (Conditional Outcome of Behavioral Strategy) *Let b be a behavioral strategy in an extensive game with simultaneous moves. We define the* outcome $O(b \mid h)$ of b conditional on h being reached *as a probability distribution over the terminal histories. Let $\gamma^* = \langle (c^1, d^1), \ldots, (c^K, d^K)\rangle \in \Sigma_T$ be a terminal history. Then there are two cases:*

1. *If there is no subhistory of γ^* which leads to h, then*

   $$O(b \mid h)(\gamma^*) = 0.$$

2. *If there is a subhistory* $\gamma = \langle (c^1, d^1), \ldots, (c^L, d^L) \rangle$ *of* γ^* *which leads from the root node to the node* $h = \sigma(c^{L-1}, d^{L-1})$ *with* $L < K$*, then*

$$O(b \,|\, h)(\gamma^*) = \prod_{k=L}^{K-1} b_0(\langle (c^1, d^1), \ldots, (c^k, d^k) \rangle)(c^{k+1}) \cdot b_1(\langle (c^1, d^1), \ldots, (c^k, d^k) \rangle)(d^{k+1}).$$

Lemma 4.1.17. *Let* $\Gamma = \langle H, (A^0_h)_{h \in H}, (A^1_h)_{h \in H}, (E_h)_{h \in H}, (T_h)_{h \in H}, \sigma, u \rangle$ *be an extensive game with simultaneous moves and* G' *the corresponding imperfect-information extensive game. If* b *is a behavioral strategy in* Γ *and* b' *the corresponding behavioral strategy in* G'*, then the conditional outcome of* b *in* h *is equal to the conditional outcome of* b' *in the corresponding information set* $I_{i,k}$*.*

Proof. Let $\gamma'^* = \langle c^1, d^1, \ldots, c^K, d^K \rangle \in \Sigma_Z$ be a terminal history in G'. Then $\gamma^* = \langle (c^1, d^1), \ldots, (c^K, d^K) \rangle \in \Sigma_T$ is a terminal history in Γ. There are two cases:

1. If $I_{i,k} = \{h\}$, let $\gamma' = \langle c^1, d^1, \ldots, c^L, d^L \rangle$ be a subhistory of γ'^* which leads to $I_{i,k}$. Then:

$$\begin{aligned} &O(b', \beta \,|\, I_{i,k})(\gamma'^*) = \\ &= \underbrace{\beta(I_{i,k})(\gamma')}_{=1} \cdot \prod_{k=L}^{K-1} b'_0(\langle c^1, d^1, \ldots, c^k, d^k \rangle)(c^{k+1}) \cdot b'_1(\langle c^1, d^1, \ldots, c^k, d^k, c^{k+1} \rangle)(d^{k+1}) = \\ &= \prod_{k=L}^{K-1} b_0(\langle (c^1, d^1), \ldots, (c^k, d^k) \rangle)(c^{k+1}) \cdot b_1(\langle (c^1, d^1), \ldots, (c^k, d^k) \rangle)(d^{k+1}) = \\ &= O(b \,|\, h)(\gamma^*). \end{aligned}$$

2. If $I_{i,k} = A^0_h$ for some $h \in H$, let $\gamma' = \langle c^1, d^1, \ldots, c^L \rangle$ be a subhistory of γ'^* which leads to $I_{i,k}$. Then:

$$\begin{aligned} &O(b', \beta \,|\, I_{i,k})(\gamma'^*) = \\ &= \underbrace{\beta(I_{i,k})(\gamma')}_{= b_0(\langle (c^1, d^1), \ldots, (c^{L-1}, d^{L-1}) \rangle)(c^L)} \cdot b'_1(\langle c^1, d^1, \ldots, c^{L-1}, d^{L-1}, c^L \rangle)(d^L) \cdot \\ &\cdot \prod_{k=L+1}^{K-1} b'_0(\langle c^1, d^1, \ldots, c^k, d^k \rangle)(c^{k+1}) \cdot b'_1(\langle c^1, d^1, \ldots, c^k, d^k, c^{k+1} \rangle)(d^{k+1}) = \\ &= \prod_{k=L}^{K-1} b_0(\langle (c^1, d^1), \ldots, (c^k, d^k) \rangle)(c^{k+1}) \cdot b_1(\langle (c^1, d^1), \ldots, (c^k, d^k) \rangle)(d^{k+1}) = \\ &= O(b \,|\, h)(\gamma^*). \end{aligned}$$

If there is no subhistory which leads to $I_{i,k}$, then $O(b', \beta \,|\, I_{i,k})(\gamma'^*) = O(b|\, h)(\gamma^*) = 0$. □

Definition 4.1.18. (Expected Conditional Payoff)
Let $\Gamma = \langle H, (A_h^0)_{h\in H}, (A_h^1)_{h\in H}, (E_h)_{h\in H}, (T_h)_{h\in H}, \sigma, u\rangle$ *be an extensive game with simultaneous moves and* b *a behavioral strategy profile. We define player* i*'s* expected payoff $U_i(b \mid h)$*, conditional being in node* $h \in H$ *with:*

$$U_i(b \mid h) := \underbrace{\sum_{(c,d)\in E_h} b_i(h)(c) \cdot b_{1-i}(h)(d) \cdot U_i(b \mid \sigma(c,d))}_{\text{sum over the non-terminal entries}} + \underbrace{\sum_{(e,f)\in T_h} b_i(h)(e) \cdot b_{1-i}(h)(f) \cdot u_i(e,f).}_{\text{sum over the terminal entries}}$$

We can also use the notion of conditional outcome of behavioral strategy b *to define the conditional expected payoff:*

$$U_i(b \mid h) = \sum_{\gamma\in\Sigma_T} O(b \mid h)(\gamma) \cdot u_i(\gamma).$$

Definition 4.1.19. (Sequential Rationality) *A behavioral strategy profile* $b^*=(b_i^*, b_{1-i}^*)$ *in an extensive game with simultaneous moves is* sequentially rational *if for each player* $i \in \{0,1\}$ *and each node* $h \in H$ *we have:*

$$U_i((b_i^*, b_{1-i}^*) \mid h) \geq U_i((b_i, b_{1-i}^*) \mid h) \; \forall \; b_i \in B_i.$$

Definition 4.1.20. (Subgame) *Let* $\Gamma = \langle H, (A_h^0)_{h\in H}, (A_h^1)_{h\in H}, (E_h)_{h\in H}, (T_h)_{h\in H}, \sigma, u\rangle$ *be an extensive game with simultaneous moves. The* subgame rooted at node $h \in H$ *is the restriction* $\Gamma|_h$ *of* Γ *to node* h *and all successor nodes of* h*.*

Definition 4.1.21. (Subgame Perfect Equilibrium) *A behavioral strategy profile* b^* *in an extensive game with simultaneous moves is a* subgame perfect equilibrium (SPE) *if it is sequentially rational*[1]*.*

Example 4.1.22. *We compute all subgame perfect equilibria in Bach or Stravinsky Game with a book option.*

$\emptyset$	0
R	1, 1
C	$C0$

$C0$	B	S
B	**2, 1**	0, 0
S	0, 0	**1, 2**

Figure 4.4.: Bach or Stravinsky with a Book Option.

[1]It is not necessary to require sequential rationality in every subgame of Γ since the definition of sequential rationality (Definition 4.1.19) already implies it.

We compute the behavioral strategy b_0 of player 0 and the behavioral strategy b_1 of player 1, such that (b_0, b_1) is sequentially rational. Let us determine b_0 and b_1 in the following way:

$$\begin{aligned} b_0(\emptyset)(R) &= \alpha, & \\ b_0(\emptyset)(C) &= 1-\alpha, & b_1(\emptyset)(0) &= 1, \\ b_0(C0)(B) &= \beta, & b_1(C0)(B) &= \gamma, \\ b_0(C0)(S) &= 1-\beta, & b_1(C0)(S) &= 1-\gamma. \end{aligned}$$

Then the expected payoffs in the root node are:

$$\begin{aligned} U_0(b \,|\, \emptyset) &= \alpha + (1-\alpha)\cdot(2\beta\gamma + (1-\beta)\cdot(1-\gamma)), \\ U_1(b \,|\, \emptyset) &= \alpha + (1-\alpha)\cdot(\beta\gamma + 2\cdot(1-\beta)\cdot(1-\gamma)). \end{aligned}$$

The expected payoffs in the node $C0$ are:

$$\begin{aligned} U_0(b \,|\, C0) &= 2\beta\gamma + (1-\beta)\cdot(1-\gamma), \\ U_1(b \,|\, C0) &= \beta\gamma + 2\cdot(1-\beta)\cdot(1-\gamma). \end{aligned}$$

There are three Nash equilibria in node $C0$:

1. *$\beta = 1$, $\gamma = 1$ with expected payoff $(2, 1)$,*
2. *$\beta = 0$, $\gamma = 0$ with expected payoff $(1, 2)$,*
3. *$\beta = \frac{2}{3}$, $\gamma = \frac{1}{3}$ with expected payoff $(\frac{2}{3}, \frac{2}{3})$.*

The behavioral strategy b is sequentially rational if one of the following cases is true:

1. *If $\beta = 1$, $\gamma = 1$, then $U_0(b \,|\, \emptyset) = 2 - \alpha \;\Rightarrow\; \alpha = 0$;*
2. *If $\beta = 0$, $\gamma = 0$, then $U_0(b \,|\, \emptyset) = 1 \;\Rightarrow\; \alpha \in [0, 1]$ is arbitrary;*
3. *If $\beta = \frac{2}{3}$, $\gamma = \frac{1}{3}$, then $U_0(b \,|\, \emptyset) = \frac{1}{3}\alpha + \frac{2}{3} \;\Rightarrow\; \alpha = 1$.*

Thus we have following cases for subgame perfect equilibria in our game from Figure 4.4:

1. *$\alpha = 0$, $\beta = 1$, $\gamma = 1$ with expected payoff $(2, 1)$,*
2. *$\alpha \in [0, 1]$, $\beta = 0$, $\gamma = 0$ with expected payoff $(1, 2 - \alpha)$,*
3. *$\alpha = 1$, $\beta = \frac{2}{3}$, $\gamma = \frac{1}{3}$ with expected payoff $(1, 1)$.*

Lemma 4.1.23. *A behavioral strategy profile b^* in an extensive game with simultaneous moves $\Gamma = \langle H, (A_h^0)_{h\in H}, (A_h^1)_{h\in H}, (E_h)_{h\in H}, (T_h)_{h\in H}, \sigma, u\rangle$ is a subgame perfect equilibrium if and only if the corresponding behavioral strategy b'^* together with the belief system β' is a perfect Bayesian equilibrium in the corresponding imperfect-information extensive game $G' = \langle A', H', Z', \chi', \rho', \sigma', u', I'\rangle$, such that β' is trivial for player 0 and equal to b_0 for player 1.*

Proof. Let b be a behavioral strategy in Γ. Since the corresponding behavioral strategy b' is conditional outcome equivalent to b according to Lemma 4.1.17 and the Bayesian consistent belief system β' in the corresponding imperfect-information extensive game G' is trivial for player 0 and equal to b_0 for player 1 according to Lemma 4.1.15, we have the following equality for player i: Let $h \in H$ such that $I_{0,k} = \{h\}$ and $I_{1,l} = A_h^1$, then

$$U_i((b_0, b_1) \,|\, h) = \sum_{\gamma \in \Sigma_T} O((b_0, b_1) \,|\, h)(\gamma) \cdot u_i(\gamma) =$$
$$= \sum_{\gamma' \in \Sigma'_Z} O((b'_0, b'_1), \beta' \,|\, \{h\})(\gamma') \cdot u'_i(\gamma') = U'_i((b'_0, b'_1), \beta' \,|\, \{h\}).$$

Then we have:

$$b^* \in B \text{ is a SPE in } \Gamma$$
$$\Leftrightarrow$$
$$U_i((b_0^*, b_1^*) \,|\, h) \geq U_i((b_0, b_1^*) \,|\, h) \; \forall\, b_0 \in B_0, \; \forall\, b_1 \in B_1$$
$$\Leftrightarrow$$
$$U'_i((b_0'^*, b_1'^*), \beta' \,|\, \{h\}) \geq U'_i((b'_0, b_1'^*), \beta' \,|\, \{h\}) \; \forall\, b'_0 \in B'_0, \; \forall\, b'_1 \in B'_1$$
$$\Leftrightarrow$$
$$b'^* \in B' \text{ is a PBE in } G'.$$

□

It is useful to introduce the concept of 'local' sequential rationality.

Definition 4.1.24. (Local Expected Payoff)
Let $\Gamma = \langle H, (A_h^0)_{h \in H}, (A_h^1)_{h \in H}, (E_h)_{h \in H}, (T_h)_{h \in H}, \sigma, u \rangle$ be an extensive game with simultaneous moves and b a behavioral strategy profile. We define player i's local expected payoff *$U_i(a^i \,|\, b, h)$ from playing an action $a^i \in A_h^i$, conditional being in node $h \in H$ and given the behavioral strategy profile b as follows.*

$$U_i(a^i \,|\, b, h) := \underbrace{\sum_{\substack{a^{1-i} \in A_h^{1-i}, \\ (a^i, a^{1-i}) \in E_h}} b_{1-i}(h)(a^{1-i}) \cdot \underbrace{U_i(b \,|\, \sigma(a^i, a^{1-i}))}_{\substack{\textit{conditional expected payoff} \\ \textit{in the child-node}}}}_{\textit{sum over the non-terminal entries}}$$
$$+ \underbrace{\sum_{\substack{a^{1-i} \in A_h^{1-i}, \\ (a^i, a^{1-i}) \in T_h}} b_{1-i}(h)(a^{1-i}) \cdot u_i(a^i, a^{1-i})}_{\textit{sum over the terminal entries}}.$$

Definition 4.1.25. (Local Best Response) *Let (b_0, b_1) be a behavioral strategy profile in $\Gamma = \langle H, (A_h^0)_{h \in H}, (A_h^1)_{h \in H}, (E_h)_{h \in H}, (T_h)_{h \in H}, \sigma, u \rangle$. An action $a_i \in A_h^i$ is called* player i's local best response *to b_{1-i} in node $h \in H$ if*

$$U_i(a_i \,|\, b, h) \geq U_i(a'_i \,|\, b, h) \; \forall\, a'_i \in A_h^i.$$

Definition 4.1.26. (Local Sequential Rationality) *A behavioral strategy profile* (b_0, b_1) *is called* locally sequentially rational *if for every player* $i \in \{0, 1\}$*, every node* $h \in H$ *and each action* $a \in A_h^i$ *we have*

$$b_i(h)(a) > 0 \Rightarrow a \text{ is a local best response to } b_{1-i} \text{ in } h.$$

4.2. Baseline Algorithm

In this section we provide our first algorithm for finding a sample equilibrium in extensive games with simultaneous moves. We show in 4.2.2 that the resulting equilibrium is a subgame perfect equilibrium. Finally we compare the runtimes of the Baseline algorithm and Lemke's algorithm in 4.2.3.

The baseline algorithm SolveGame finds a sample equilibrium of an input extensive game with simultaneous moves $\Gamma = \langle H, (A_h^0)_{h \in H}, (A_h^1)_{h \in H}, (E_h)_{h \in H}, (T_h)_{h \in H}, \sigma, u \rangle$ by recursively computing equilibria in *all* subgames of Γ 'from bottom to top' by calling the function SolveNode. There are two cases for the input node h:

- **(h is not terminal)** If h is not terminal, then assume that it has n children $\{c_1, \ldots, c_n\}$. The function SolveNode calls itself recursively n-times with c_k as the input parameter for $k \in \{1, \ldots, n\}$.

- **(h is terminal)** If h is terminal, then the algorithm computes an equilibrium in a corresponding bimatrix game G'_h of h and the payoffs are propagated to the corresponding entry of the parent node of h (if h has no parent and is terminal, then Γ is a normal-form game). An equilibrium in G'_h is computed by the function GetEquilibrium in the following way: If a pure equilibrium exists in G'_h, then it is returned by the function GetPureEquilibrium (Algorithm 1). If there is no pure equilibrium and G'_h is zero-sum, then the solutions of the linear programs are returned by the function SolveLP (Algorithm 3). If no pure equilibrium exists and G'_h is *general-sum*, then the return value is the solution of the mixed integer program and is computed by the function SolveMIP (Algorithm 4).

Let $h \in H$ with $A_h^0 = \{1, \ldots, m\}$ and $A_h^1 = \{1, \ldots, n\}$. The pseudocode of the baseline algorithm is given in Algorithm 5.

Algorithm 5 The Baseline Algorithm

```
1: function SolveGame(Γ)
2:     return SolveNode('the root node of Γ')
3: end function

4: function SolveNode(h)
5:     for i ← 1, ..., m do
6:         for j ← 1, ..., n do
7:             if (i, j) ∈ E_h then                ▷ Does entry (i, j) have a child?
8:                 eq ←SolveNode(σ(i, j))          ▷ Solve the child-node.
9:                 u(i, j) ← U(eq)                 ▷ Propagate the expected utility.
10:            end if
11:        end for
12:    end for
13:    return GetEquilibrium(h)
14: end function

15: function GetEquilibrium(h)
16:    eq ← GetPureEquilibrium(h)
17:    if eq = ∅ ∧ 'h is zero-sum' then
18:        eq ← SolveLP(h)
19:    else if eq = ∅ ∧ 'h is not zero-sum' then
20:        eq ← SolveMIP(h)
21:    end if
22:    return eq
23: end function
```

4.2.1. Example Computation with the Baseline Algorithm

We illustrate an equilibrium computation with the baseline algorithm on the game from Figure 4.5.

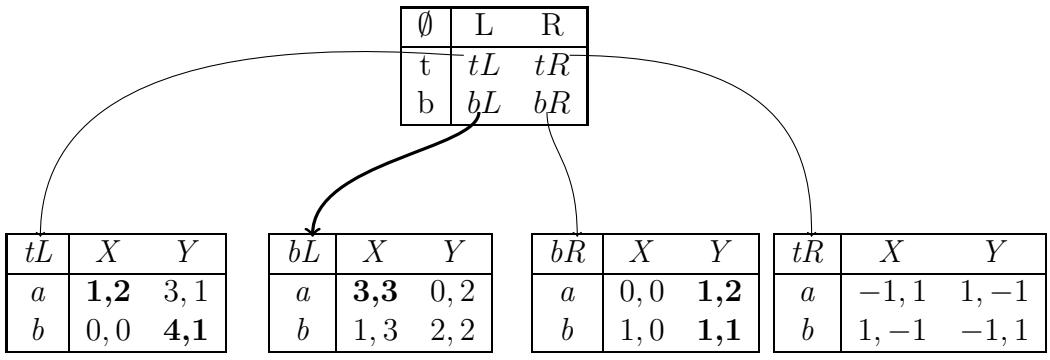


Figure 4.5.: Example Game for the Baseline Algorithm.

First the baseline algorithm computes an equilibrium in each terminal node tL, tR, bL and bR. It finds a pure equilibrium (a, X) with payoff $(1, 2)$ in the node tL since it is the first one. The second pure equilibrium in tL would be (b, Y) with payoff $(4, 1)$ and the third one is a mixed Equilibrium $((a \to 0.5, b \to 0.5), (X \to 0.5, Y \to 0.5))$ with expected payoff $(2, 2)$. The algorithm finds a pure equilibrium (a, X) with payoff $(3, 3)$ in node bL; a pure equilibrium (a, Y) with payoff $(1, 2)$ in node bR; and a mixed equilibrium $(b_0(tR)(a) = \frac{1}{2}; b_1(tR)(X) = \frac{1}{2})$ with expected payoff $(0, 0)$ in the zero-sum node tR. After all child-nodes of the root node $\emptyset$ are 'solved', payoffs at their equilibria are propagated to the corresponding entries in the root node: $(t, L) \leftarrow (1, 2)$, $(t, R) \leftarrow (0, 0)$, $(b, L) \leftarrow (3, 3)$ and $(b, R) \leftarrow (1, 2)$. The modified payoff-matrix at the root node is shown in the next figure. There the baseline algorithm finds a pure equilibrium (b, L) with payoff $(3, 3)$.

$\emptyset$	L	R
t	$1, 2$	$0, 0$
b	**3,3**	$1, 2$

Figure 4.6.: Propagated Payoffs at the Root Node.

Now the baseline algorithm is finished and returns the following equilibrium in behavioral strategies with expected payoff $(3, 3)$:

$$
\begin{aligned}
b_0(\emptyset)(t) = 0,\ b_0(\emptyset)(b) = 1, &\quad b_1(\emptyset)(L) = 1,\ b_1(\emptyset)(R) = 0;\\
b_0(tL)(a) = 1,\ b_0(tL)(b) = 0, &\quad b_1(tL)(X) = 1,\ b_1(tL)(Y) = 0;\\
b_0(tR)(a) = \frac{1}{2},\ b_0(tR)(b) = \frac{1}{2}, &\quad b_1(tR)(X) = \frac{1}{2},\ b_1(tR)(Y) = \frac{1}{2};\\
b_0(bL)(a) = 1,\ b_0(bL)(b) = 0, &\quad b_1(bL)(X) = 1,\ b_1(bL)(Y) = 0;\\
b_0(bR)(a) = 1,\ b_0(bR)(b) = 0, &\quad b_1(bR)(X) = 0,\ b_1(bR)(Y) = 1.
\end{aligned}
$$

This is a subgame perfect equilibrium, since b is sequentially rational.

4.2.2. Analysis of the Baseline Algorithm

In this section we show that the baseline algorithm is sound and complete. An algorithm is sound if any proposed 'solution' that the algorithm returns is a solution. And an algorithm is called complete if, whenever one solution exists, then the algorithm finds one.

First we need to specify what does a 'solution' exactly mean in our case. The input of the baseline algorithm is an extensive game with simultaneous moves as defined in 4.1.1. The algorithm computes an equilibrium in behavioral strategies in each subgame. We claim that the resulting equilibrium is a subgame perfect equilibrium.

Theorem 4.2.1. (Subgame Perfect Equilibrium) *The output of the baseline algorithm (Algorithm 5) is a subgame perfect equilibrium in the input game.*

Proof. Let $\Gamma = \langle H, (A_h^0)_{h\in H}, (A_h^1)_{h\in H}, (E_h)_{h\in H}, (T_h)_{h\in H}, \sigma, u\rangle$ be a 2-player extensive game with simultaneous moves. Since the baseline algorithm is recursive, it is natural to use induction over the depth of the subgame trees to prove the claim. Suppose that $d(h)$ is the depth of the subtree rooted at the node $h \in H$.

Step 1 ($d(h) = 0$): Let $h^0 \in H$ such that $d(h) = 0$. Therefore h^0 must be terminal. The baseline algorithm finds a Nash equilibrium (α_0^*, α_1^*) with expected payoff (u_0, u_1) in the input extensive game with simultaneous moves which consists only of the terminal node h^0. It is easy to see that this equilibrium is also a subgame perfect equilibrium since for each player $i \in \{0, 1\}$ the condition on a Nash equilibrium in mixed strategies (Definition 2.1.9) implies the sequential rationality (Definition 4.1.19):

$$U_i(\alpha^* \,|\, h^0) = U_i(\alpha_i^*, \alpha_{1-i}^*) = \max_{\alpha_i \in \Delta(A^i_{h^0})} U_i(\alpha_i, \alpha_{1-i}^*) = (u_0, u_1).$$

Step 2 ($n \to n+1$): Now assume that the baseline algorithm finds a subgame perfect equilibrium in all input nodes h^n with $d(h^n) \leq n$. Let $b = (b_0, b_1)$ be a behavioral strategy returned by the baseline algorithm for the input node h^{n+1} with $d(h^{n+1}) = n + 1$. We want to show that b is also a subgame perfect equilibrium. Since b is sequentially rational in all nodes h^n with $d(h^n) \leq n$, we know that for all $b' \in B_0 \times B_1$ the expected payoff is $U_i(b' \,|\, \sigma(c, d)) \leq U_i(b \,|\, \sigma(c, d))$ for all child-nodes $\sigma(c, d)$ of h^{n+1} with $(c, d) \in E_{h^{n+1}}$. Since the expected payoffs of the child-nodes $U_i(b \,|\, \sigma(c, d))$ are propagated to the corresponding non-terminal entries $(c, d) \in E_{h^{n+1}}$, we can handle all entries as terminal with the payoffs $u_i(c, d)$ for player $i \in \{0, 1\}$. Let us assume that the sequential rationality of b does not hold in node h^{n+1}, which means that there is a behavioral strategy $b' \in B_0 \times B_1$ such that $b' \neq b$ and $U_i(b' \,|\, h^{n+1}) > U_i(b \,|\, h^{n+1})$ for at least one player $i \in \{0, 1\}$. Then we would have:

$$\begin{aligned} U_i(b' \,|\, h^{n+1}) &= \sum_{(c,d)\in E_h \cup T_h} b_i'(h^{n+1})(c) \cdot b_{1-i}'(h^{n+1})(d) \cdot u_i(c, d) \\ > \sum_{(c,d)\in E_h \cup T_h} & b_i(h^{n+1})(c) \cdot b_{1-i}(h^{n+1})(d) \cdot u_i(c, d) = U_i(b \,|\, h^{n+1}). \end{aligned}$$

This would be a contradiction to the fact that the expected payoff of $b_i(h^{n+1})$ must be maximal in h^{n+1} for $i \in \{0, 1\}$:

$$U_i(b \,|\, h^{n+1}) = U_i(b_i(h^{n+1}), b_{1-i}(h^{n+1})) = \max_{b_i'(h^{n+1} \in \Delta(A_h^i))} U_i(b_i'(h^{n+1}), b_{1-i}(h^{n+1})).$$

Thus (b_0, b_1) is sequentially rational in all nodes h with $d(h) \leq n + 1$ and therefore b is a subgame perfect equilibrium in Γ. □

Corollary 4.2.2. (Soundness and Completeness of the Baseline Algorithm) *The baseline algorithm (Algorithm 5) is sound and complete.*

Proof. The soundness follows directly from Theorem 4.2.1. The baseline algorithm is complete since a Nash equilibrium exists in each node (Lemma 2.1.11) and it is computed

with the complete algorithms for finding a pure equilibrium (Algorithm 1) or mixed equilibrium using the mixed integer programming (Algorithm 4) for general-sum games; or linear programming (Algorithm 3) for zero-sum games. Thus the baseline algorithm always finds a sample SPE in the input 2-player extensive game with simultaneous moves. □

The baseline algorithm finds a solution in exponential time in the worst case, since it uses exponential-time algorithms for finding a mixed equilibrium in each node[2]. With a little modification of the baseline algorithm we can also find a sample equilibrium with certain properties. For example an equilibrium with maximal payoff for player $i \in \{0, 1\}$. We have already introduced the required algorithms for finding a pure and mixed equilibrium with such properties in Chapter 2.

4.2.3. Baseline Algorithm in Practice

In this section we compare the performance of the baseline algorithm (Algorithm 5) with Lemke's algorithm which is briefly introduced at the end of Section 3. We use the Gambit[3] implementation of Lemke's algorithm. The command line tool GAMBIT-LCP can be used to find a sample equilibrium in a two-player extensive game with imperfect information via linear complementary. The extensive games with imperfect information are represented in so-called *efg-format* (see the detailed description on the Gambit website [13]). Every extensive game with simultaneous moves needs to be transformed into the corresponding extensive game with imperfect information (Definition 4.1.9) before it can be represented in the efg-format and solved with GAMBIT-LCP.

We implemented the baseline algorithm in *C++*. The tests were contacted on a Macbook Pro with 2.16 GHz Intel Core 2 Duo processor and 4 GB 667 Mhz DDR2 SDRAM. We tested both algorithms on randomly generated extensive games with simultaneous moves or respectively their corresponding imperfect information forms.

The random games are generated with two variables: a and d. The variable a determines the maximal number of actions available for every player $i \in \{0, 1\}$ in each node. The variable d determines the maximal depth of the game tree. The payoffs at the terminal entries are random numbers uniformly distributed between -1 and 1. Each player $i \in \{0, 1\}$ has a random number of actions $2 \leq |A_h^i| \leq a$ in $h \in H$. Therefore each randomly generated extensive game with simultaneous moves has a different number of nodes between $\frac{4^{d+1}-1}{3}$ and $\frac{(a^2)^{d+1}-1}{a^2-1}$ (see Lemma 4.1.11). Table 4.1 compares the average ($\varnothing$ SIM / $\varnothing$ EFG), minimal (min SIM / min EFG) and maximal (max SIM / max EFG) number of nodes of 100 random extensive games with simultaneous moves (SIM) and of their corresponding imperfect-information extensive games in efg format (EFG) for $a = 2$ and $1 \leq d \leq 4$. The values of a and d are represented in the vector (a, d) in the first column.

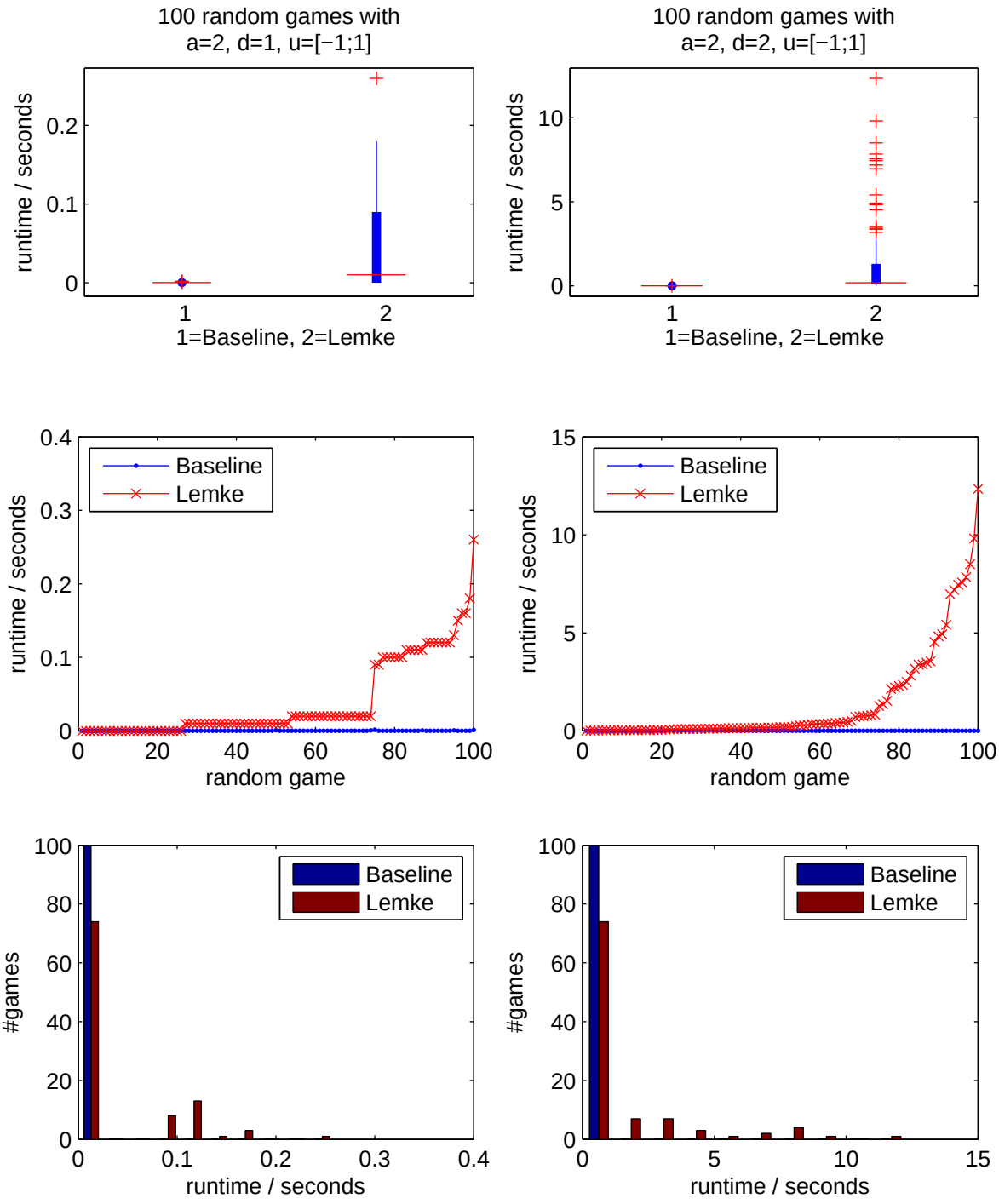
[2]Exponential in number of actions in every node.

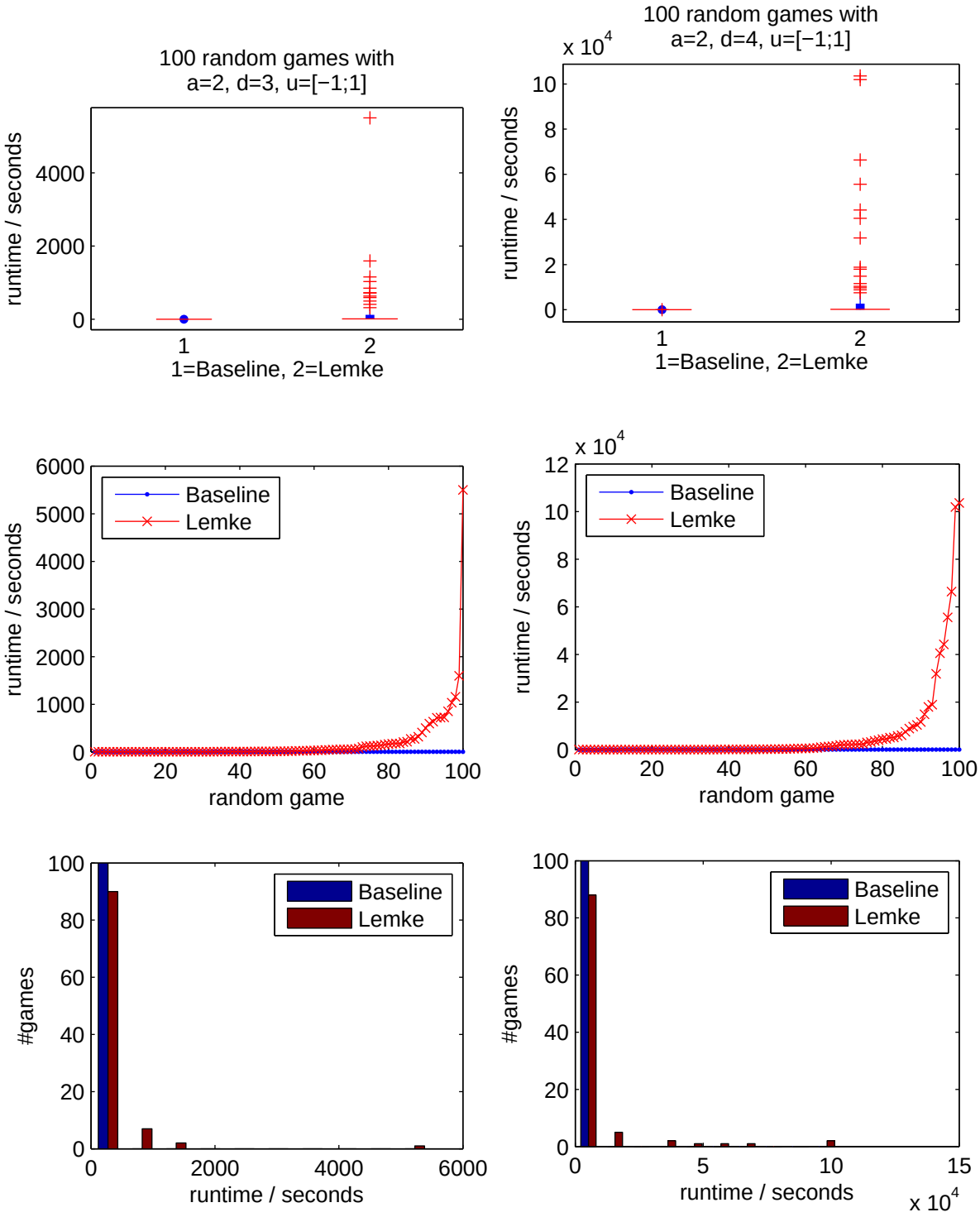
[3]Gambit is a free source software bundle for game theory. See http://www.gambit-project.org.

	min **SIM**	min **EFG**	∅ **SIM**	∅ **EFG**	max **SIM**	max **EFG**
$(2,1)$	2	5	4	14	5	29
$(2,2)$	3	8	9	33	19	78
$(2,3)$	4	9	23	87	51	203
$(2,4)$	5	12	44	166	117	464

Table 4.1.: Number of Nodes in 100 Randomly Generated Games.

The tests are visualized in three graphs for each pair of a and d. The first graph is a box plot. The second one shows the runtime of both algorithms sorted in ascending order of the runtime of Lemke's algorithm. The third graph is a distribution plot which shows several runtime frequencies.


100 random games with
a=2, d=1, u=[−1;1]
runtime / seconds
0
0.1
0.2
1
2
1=Baseline, 2=Lemke
100 random games with
a=2, d=2, u=[−1;1]
runtime / seconds
0
5
10
1
2
1=Baseline, 2=Lemke
runtime / seconds
random game
Baseline
Lemke
0
0.1
0.2
0.3
0.4
0
20
40
60
80
100
runtime / seconds
random game
Baseline
Lemke
0
5
10
15
0
20
40
60
80
100
#games
runtime / seconds
Baseline
Lemke
0
20
40
60
80
100
0
0.1
0.2
0.3
0.4
#games
runtime / seconds
Baseline
Lemke
0
20
40
60
80
100
0
5
10
15



The next table shows the summary of the statistical results of the runtime tests in 100 randomly generated games for $a = 2$ and $1 \leq d \leq 4$. The a and d values are represented in the the vector (a, d) in the first row.

	$(2,1)$	$(2,2)$	$(2,3)$	$(2,4)$
Mean BL	0.0001626	0.00043396	0.00105	0.002
Mean LEM	0.039	1.3767	182.5423	6180.6619
Standard Deviation BL	0.00022354	0.00037812	0.00089188	0.0014702
Standard Deviation LEM	0.053852	2.4755	602.3805	17700.3622
25th Percentile BL	6.95e-05	0.000182	5e-04	0.001
25th Percentile LEM	1e-04	0.09	2.095	19.69
50th Percentile BL	9.95e-05	0.0002815	0.001	0.002
50th Percentile LEM	0.01	0.19	11.93	215.99
75th Percentile BL	0.0001605	0.000562	0.002	0.003
75th Percentile LEM	0.1	1.36	122.115	2687.79
Number of outliers BL	6	3	0	1
Number of outliers LEM	0	12	12	14

Table 4.2.: Runtime Comparison between the Baseline and Lemke's Algorithms.

If we compare the medians of the runtimes of the baseline and Lemke's algorithm, then we see that the baseline algorithm clearly outperforms Lemke's algorithm: Even in the small games generated for $d = 1$ the baseline algorithm is 100 faster than Lemke's algorithm in median values. With the increasing number of nodes in the randomly generated games, the gap between the runtimes becomes exponentially higher. In general the trend is: the more nodes in a game the higher the standard deviation of Lemke's algorithms. The standard deviation of the baseline algorithm slowly increases, too, but remains less than $1.5 \cdot 10^{-3}$ seconds. Lemke's runtimes have 14 outliers in 100 randomly generated games with $d = 4$ and maximal 464 nodes in the corresponding imperfect-information extensive games. These outliers have up to 10.000.000 higher values than the maximal value of the baseline runtime in those games. An extensive game with simultaneous moves from our experiments has up to 6 times less nodes than its corresponding imperfect-information extensive game represented in the efg format (see Lemma 4.1.11). The baseline algorithm locally computes a Nash equilibrium in every node of the game. It first tries to find a pure equilibrium, which is much faster than solving a MIP or a LCP. By contrast, Lemke's algorithm works globally and solves a LCP generated from the sequence form of the corresponding imperfect-information game. Therefore it is natural that the baseline algorithm is faster than Lemke's algorithm.

4.3. Interval-Heuristic Algorithm

In the worst case the baseline algorithm solves a MIP in each node. We use a heuristic to skip parts of the game tree. The motivation is given in Example 4.3.1.

Example 4.3.1. *The pure Nash equilibria are bold faced in the terminal nodes of the extensive game with simultaneous moves from the next figure.*

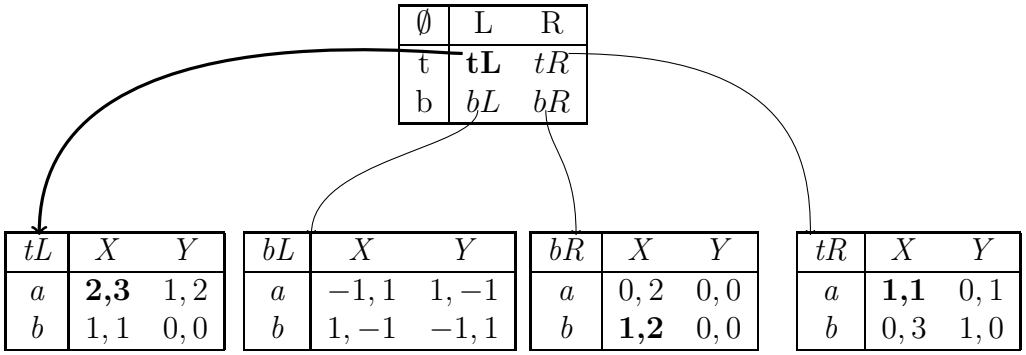


Figure 4.7.: Motivating Example for Heuristic.

Entry (a, X) *is the unique Nash equilibrium with payoff* $(2, 3)$ *in node* tL*. If we propagate the expected payoff to the corresponding entry* (t, L) *of the root node, then* (t, L) *is a Nash equilibrium there if player* 0 *has no incentive to play the action* b *instead of* t *and player* 1 *has no incentive to play the action* R *instead of* L*. Player* 0*'s maximal payoff is* 1 *in the node* bL *which is less than his expected payoff* 2 *at* (t, L)*. Player* 1*'s maximal payoff is* 3 *in the node* tR *which is equal to his expected payoff at* (t, L)*. Thus no player has an incentive to deviate from* (t, L) *independent of the behavioral strategies* $b_i(bL)$ *and* $b_i(tR)$ *for* $i \in \{0, 1\}$*. However, it is not a subgame perfect equilibrium since we do not have a sequentially rational behavioral strategy which is defined in* every *node of* Γ*. Instead of that we have the strategy* $b' = (b'_0, b'_1)$ *which is defined in the root node* $\emptyset$ *and in the node* tL*:*

$$b'_0(\emptyset)(t) = 1,\ b'_0(\emptyset)(b) = 0,\ b'_0(tL)(a) = 1,\ b'_0(tL)(b) = 0;$$
$$b'_1(\emptyset)(L) = 1,\ b'_1(\emptyset)(R) = 0,\ b'_1(tL)(X) = 1,\ b'_1(tL)(Y) = 0.$$

The 'expected payoff' of b' *is* $(2, 3)$*. We call such a partial description of a behavioral strategy* $b' = (b'_0, b'_1)$ *a 'partial behavioral strategy'. If we extend* b' *to the nodes* tR*,* bL *and* bR *such that* $b'(tR)$*,* $b'(bL)$ *and* $b'(bR)$ *are Nash equilibrium strategies in the accordant nodes, then* b' *would be a subgame perfect equilibrium strategy.*

Definition 4.3.2. (Partial Behavioral Strategy)
Let $\Gamma = \langle H, (A^0_h)_{h\in H}, (A^1_h)_{h\in H}, (E_h)_{h\in H}, (T_h)_{h\in H}, \sigma, u\rangle$ *be an extensive game with simultaneous moves and* $\emptyset \neq H^* \subseteq H$*. A* partial behavioral strategy (pbs) *of player* $i \in \{0, 1\}$ *is a map* b'_i *which assigns to every node* $h \in H^*$ *a probability distribution* $b'_i(h)$ *over the set of the player* i*'s actions* A^i_h*. We use the following notations:*

- *We denote the subset H^* where b' is defined with* $\text{supp}(b') := H^*$.
- *The set of player i's actions in node h which are played with positive probability is denoted with* $\text{supp}(b'_i(h)) := \{a^i \in A^i_h \mid b'_i(h)(a) > 0\}$.
- $\text{supp}(b'(h)) := \text{supp}(b'_0(h)) \times \text{supp}(b'_1(h))$.

The difference between partial behavioral strategy and behavioral strategy (Definition 4.1.5) in an extensive game with simultaneous moves Γ is that a pbs does not have to be defined on all nodes of Γ. The expected conditional payoff $U_i(b' \mid h)$ of a pbs b' is defined analogous to Definition 4.1.18. Every pbs b' induces a so-called *partial game tree* Γ'.

Definition 4.3.3. (Partial Game)
Let $\Gamma = \langle H, (A^0_h)_{h\in H}, (A^1_h)_{h\in H}, (E_h)_{h\in H}, (T_h)_{h\in H}, \sigma, u\rangle$ *be an extensive game with simultaneous moves and* b' *a pbs.*
$\Gamma' = \langle H', (A'^0_h)_{h\in H'}, (A'^1_h)_{h\in H'}, (E'_h)_{h\in H'}, (T'_h)_{h\in H'}, \sigma', u'\rangle$ *is called a* partial game *of* Γ *induced by* b' *such that:*

- $H' := \text{supp}(b')$;
- $A'^i_h := \text{supp}(b'_i(h))$ *for player i in node* $h \in H'$;
- $E'_h := E_h \cap (A'^0_h \times A'^1_h)$ *for node* $h \in H'$;
- $T'_h := T_h \cap (A'^0_h \times A'^1_h)$ *for node* $h \in H'$;
- $\sigma'(a, b) := \sigma(a, b)$ *for* $(a, b) \in E'_h$;
- $u'(a, b) := u(a, b)$ *for* $(a, b) \in T'_h$.

Example 4.3.4. *The next figure shows a partial game induced by a pbs* b' *from Example 4.3.1.*
b' *is defined on* $\text{supp}(b') = \{\emptyset, tL\}$ *with*

$$b'_0(\emptyset)(t) = 1,\ b'_0(\emptyset)(b) = 0,\ b'_0(tL)(a) = 1,\ b'_0(tL)(b) = 0;$$
$$b'_1(\emptyset)(L) = 1,\ b'_1(\emptyset)(R) = 0,\ b'_1(tL)(X) = 1,\ b'_1(tL)(Y) = 0.$$

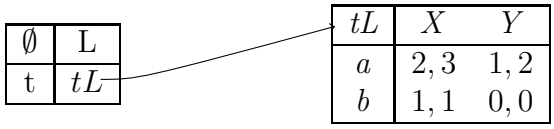


Figure 4.8.: Partial Game from Example 4.3.1.

4.3.1. Interval Heuristic

The interval heuristic (IH) computes a partial behavioral strategy in an extensive game with simultaneous moves. We call it 'interval heuristic' since we compare so-called 'utility intervals'. The main idea of the IH is to propagate the minimal and maximal outcomes in the terminal nodes up to the root node and compare them and their mean values according to the utility comparison for the Nash equilibrium computation in the normal-form games.

Definition 4.3.5. (Utility Interval)
Let $\Gamma = \langle H, (A_h^0)_{h\in H}, (A_h^1)_{h\in H}, (E_h)_{h\in H}, (T_h)_{h\in H}, \sigma, u\rangle$ *be an extensive game with simultaneous moves. Player* i*'s* utility interval *in a terminal node* $h \in H_T$ *is an interval* $\Omega_h^i := [a, b]$*, where* a *is player* i*'s minimal utility in node* h*, and* b *his maximal utility. We define* $\Omega^i_{(k,l)} := \Omega^i_{\sigma(k,l)}$ *in a non-terminal entry* $(k, l) \in E_h$ *as the utility interval in the child-node* $\sigma(k, l)$*. We define* $\Omega^i_{(u,v)} := [u_i(u, v), u_i(u, v)]$ *in a terminal entry* $(u, v) \in T_h$*. Then we recursively compute the utility interval* Ω_h^i *in an arbitrary node* $h \in H$ *in the following way:*

$$\Omega_h^i := [\min_{(x,y)\in E_h\cup T_h}(\min \Omega^i_{(x,y)}), \max_{(x,y)\in E_h\cup T_h}(\max \Omega^i_{(x,y)})].$$

Example 4.3.6. *We illustrate this definition on the game from Example 4.3.1. The utility intervals for both players in the game nodes are given below:*

$$\begin{aligned} \Omega^0_{tL} &= [0,2] & \Omega^1_{tL} &= [0,3] \\ \Omega^0_{tR} &= [0,1] & \Omega^1_{tR} &= [0,3] \\ \Omega^0_{bL} &= [-1,1] & \Omega^1_{bL} &= [-1,1] \\ \Omega^0_{bR} &= [0,1] & \Omega^1_{bR} &= [0,2] \\ \Omega^0_{\emptyset} &= [-1,2] & \Omega^1_{\emptyset} &= [-1,3] \end{aligned}$$

The process of the recursive computation of the utility intervals in the root node $\emptyset$ *is illustrated in the next figure:*

tL	X	Y
u	$[2,2],[3,3]$	$[1,1],[2,2]$
v	$[1,1],[1,1]$	$[0,0],[0,0]$

tR	X	Y
u	$[1,1],[1,1]$	$[0,0],[1,1]$
v	$[0,0],[3,3]$	$[1,1],[0,0]$

bL	X	Y
u	$[-1,-1],[1,1]$	$[1,1],[-1,-1]$
v	$[1,1],[-1,-1]$	$[-1,-1],[1,1]$

bR	X	Y
u	$[0,0],[2,2]$	$[0,0],[0,0]$
v	$[1,1],[2,2]$	$[0,0],[0,0]$

$\emptyset$	L	R
t	$[0,2],[0,3]$	$[0,1],[0,3]$
b	$[-1,1],[-1,1]$	$[0,1],[0,2]$

Figure 4.9.: Example Computation of the Utility Intervals.

The pseudocode for the computation of the utility intervals is given in Algorithm 6. Let $\Gamma = \langle H, (A_h^0)_{h\in H}, (A_h^1)_{h\in H}, (E_h)_{h\in H}, (T_h)_{h\in H}, \sigma, u\rangle$ be an extensive game with simultaneous moves. The input of the algorithm is a node $h \in H$ with $A_h^0 := \{1, \ldots, m\}$ and $A_h^1 := \{1, \ldots, n\}$. The output of the algorithm is a pair of utility intervals (Ω_h^0, Ω_h^1).

Algorithm 6 Utility-Intervals Computation

function GetUtilityIntervals(h)
  for $i \leftarrow 1, \ldots, |A_h^0|$ **do**
    for $j \leftarrow 1, \ldots, |A_h^1|$ **do**
      if $(i,j) \in E_h$ **then**
        $(\Omega^0_{(i,j)}, \Omega^1_{(i,j)}) \leftarrow$GetUtilityIntervals$(\sigma(i,j))$
      else
        $(\Omega^0_{(i,j)}, \Omega^1_{(i,j)}) \leftarrow ([u_0(i,j), u_0(i,j)], [u_1(i,j), u_1(i,j)])$
      end if
    end for
  end for
  $\Omega_h^0 \leftarrow [\min_{(x,y)\in E_h\cup T_h}(\min \Omega^0_{(x,y)}), \max_{(x,y)\in E_h\cup T_h}(\max \Omega^0_{(x,y)})]$
  $\Omega_h^1 \leftarrow [\min_{(x,y)\in E_h\cup T_h}(\min \Omega^1_{(x,y)}), \max_{(x,y)\in E_h\cup T_h}(\max \Omega^1_{(x,y)})]$
  return (Ω_h^0, Ω_h^1)
end function

Now we develop tools to evaluate the utility intervals. With the help of this evaluation we choose which actions are to play with a positive probability. We compare the utility intervals starting in the root node and compute the so-called '*in*' and '*out*' values in each entry for both players. The idea of *in* and *out* values is motivated in Example 4.3.7.

Example 4.3.7. (Motivating the *in* and *out* values)
Let us take a look at the node tL from the game from Example 4.3.1. The next figure shows the procedure of Nash equilibrium computation with the incoming and outgoing arrows. An entry has an outgoing arrow if player $i \in \{0, 1\}$ has an incentive to deviate from it to another entry to get there a higher payoff.

tL	X	Y
a	**2,3**	$1, 2$
b	$1, 1$	$0, 0$

Figure 4.10.: Node tL from the game from Example 4.3.1.

Let us denote the number of incoming arrows of entry (x, y) for player i with in^i_{xy} and the number of the outgoing arrows with out^i_{xy}. The entry (a, X) is the only pure

Nash equilibrium in node tL. (a, X) *is also the only entry with* $out^0_{aX} = out^1_{aX} = 0$ *and* $in^0_{aX} + in^1_{aX} = \max_{(x,y)\in T_{tL}}\{in^0_{xy} + in^1_{xy}\} = 1$. *In general an entry* $(x^*, y^*) \in T_h$ *is a pure Nash equilibrium in a terminal node* h *if the out values are zero for both players.*

Now we generalize the idea of *in* and *out* values from Example 4.3.7 to the non-terminal nodes. To compute the *in* and *out* values in a non-terminal node let us compare the utility intervals there instead of comparing the payoffs in the terminal nodes. Let $\Gamma = \langle H, (A^0_h)_{h\in H}, (A^1_h)_{h\in H}, (E_h)_{h\in H}, (T_h)_{h\in H}, \sigma, u\rangle$ be an extensive game with simultaneous moves, $(a, b) \in E_h \cup T_h$ and $(c, d) \in E_h \cup T_h$ two entries in a node $h \in H$ with the utility intervals $\Omega^i_{ab} = [min^i_{ab}, max^i_{ab}]$ and $\Omega^i_{cd} = [min^i_{cd}, max^i_{cd}]$. We compute the *in* and *out* values by comparing Ω^i_{ab} with Ω^i_{cd} for each player in the following way:

Case 1: If $\Omega^i_{ab} > \Omega^i_{cd}$ $(min^i_{ab} > max^i_{cd})$, then increase the player i's *in* value of the entry (a, b) and *out* value of the entry (c, d): $in^i_{ab} \leftarrow in^i_{ab} + 1$, $out^i_{cd} \leftarrow out^i_{cd} + 1$.

Case 2: If $\Omega^i_{ab} < \Omega^i_{cd}$ $(max^i_{ab} < min^i_{cd})$, then increase the player i's *in* value of the entry (c, d) and *out* value of the entry (a, b): $in^i_{cd} \leftarrow in^i_{cd} + 1$, $out^i_{ab} \leftarrow out^i_{ab} + 1$.

Case 3: If $\Omega^i_{ab} \cap \Omega^i_{cd} \neq \emptyset$, then assume that the equilibria-payoffs are continuously uniform-distributed in the utility intervals with the mean values $m^i_{ab} = \frac{1}{2}(min^i_{ab} + max^i_{ab})$ and $m^i_{cd} = \frac{1}{2}(min^i_{cd} + max^i_{cd})$. If we compare these mean values, then we have the following three cases:

Case 3.1: If $m^i_{ab} > m^i_{cd}$, then increase the player i's *in* value of the entry (a, b): $in^i_{ab} \leftarrow in^i_{ab} + 1$. Note that we do not change the *out* values.

Case 3.2: If $m^i_{cd} > m^i_{ab}$, then increase the player i's *in* value of the entry (c, d): $in^i_{cd} \leftarrow in^i_{cd} + 1$.

Case 3.3: If $m^i_{cd} = m^i_{ab}$, then increase the player i's *in* value of the both entries (a, b) and (c, d): $in^i_{cd} \leftarrow in^i_{cd} + 1$, $in^i_{ab} \leftarrow in^i_{ab} + 1$.

Now after the *in* and *out* values have been computed in every entry for each player, test if there is an entry $(k, l) \in E_h \cup T_h$ with the maximal sum of the *in* values $in^0 + in^1$ and zero *out* values for both players. We call this entry the *pivot entry* or just the *pivot*. We implement the computation of the *in* and *out* values in the function GetPivot. The pseudocode is given in Algorithm 7. The input is a node h with $A^0_h = \{1, \ldots, m\}$ and $A^1_h = \{1, \ldots, n\}$. The output of the function is the pivot entry if it exists or 0 otherwise.

Algorithm 7 Pivot Computation

function GetPivot(h)
  $\forall (i,j) \in E_h : (\Omega^0_{ij}, \Omega^1_{ij}) \leftarrow$ GetUtilityIntervals($\sigma(i,j)$)
  $\forall (i,j) \in T_h, \forall p \in \{0,1\} : \Omega^p_{ij} \leftarrow [u_p(i,j), u_p(i,j)]$
  for $i \leftarrow 1, \ldots, |A^0_h|$ **do**
    for $j \leftarrow 1, \ldots, |A^1_h|$ **do**
      $x \leftarrow (i,j)$, $max^p_x \leftarrow \max\{\Omega^p_x\}$, $min^p_x \leftarrow \min\{\Omega^p_x\}$
      for $k \leftarrow i+1, \ldots, |A^0_h|$ **do** ▷ Comparison for player 0.
        $y \leftarrow (k,j)$, $max^p_y \leftarrow \max\{\Omega^p_y\}$, $min^p_y \leftarrow \min\{\Omega^p_y\}$
        if $max^0_x < min^0_y$ **then**
          $in^0_y \leftarrow in^0_y + 1$, $out^0_x \leftarrow out^0_x + 1$
        else if $max^0_y < min^0_x$ **then**
          $in^0_x \leftarrow in^0_x + 1$, $out^0_y \leftarrow out^0_y + 1$
        else
          $m_x \leftarrow 1/2 \cdot (min^0_x + max^0_x)$, $m_y \leftarrow 1/2 \cdot (min^0_y + max^0_y)$
          if $m_x < m_y$ **then**
            $in^0_y \leftarrow in^0_y + 1$
          else if $m_y < m_x$ **then**
            $in^0_x \leftarrow in^0_x + 1$
          else
            $in^0_x \leftarrow in^0_x + 1$, $in^0_y \leftarrow in^0_y + 1$
          end if
        end if
      end for
      for $k \leftarrow j+1, \ldots, |A^1_h|$ **do** ▷ Comparison for player 1.
        $y \leftarrow (i,k)$, $max^p_y \leftarrow \max\{\Omega^p_y\}$, $min^p_y \leftarrow \min\{\Omega^p_y\}$
        if $max^1_x < min^1_y$ **then**
          $in^1_y \leftarrow in^1_y + 1$, $out^1_x \leftarrow out^1_x + 1$
        else if $max^1_y < min^1_x$ **then**
          $in^1_x \leftarrow in^1_x + 1$, $out^1_y \leftarrow out^1_y + 1$
        else
          $m_x \leftarrow 1/2 \cdot (min^1_x + max^1_x)$, $m_y \leftarrow 1/2 \cdot (min^1_y + max^1_y)$
          if $m_x < m_y$ **then**
            $in^1_y \leftarrow in^1_y + 1$
          else if $m_y < m_x$ **then**
            $in^1_x \leftarrow in^1_x + 1$
          else
            $in^1_x \leftarrow in^1_x + 1$, $in^1_y \leftarrow in^1_y + 1$
          end if
        end if
      end for
    end for
  end for
  $pivot \leftarrow \arg\max_{(u,v) \in E_h \cup T_h} \{in^0_{uv} + in^1_{uv} \mid out_{uv} = 0\}$
  return *pivot*
end function

Example 4.3.8. *We compute the pivot entry in the root node from Example 4.3.1 with Algorithm 7. We have already computed the utility intervals in Example 4.3.6. Let us compare them pair-wisely and compute the in and out values for every entry.*

$$\begin{aligned}
\Omega^0_{tL} \text{ vs. } \Omega^0_{bL} &: [0,2] \cap [-1,1] \neq \emptyset \Rightarrow \text{ Case 3: } m^0_{tL} > m^0_{bL} \Rightarrow in^0_{tL} = 1 \\
\Omega^1_{bL} \text{ vs. } \Omega^1_{bR} &: [-1,1] \cap [0,2] \neq \emptyset \Rightarrow \text{ Case 3: } m^1_{bR} > m^1_{bL} \Rightarrow in^1_{bR} = 1 \\
\Omega^0_{tR} \text{ vs. } \Omega^0_{bR} &: [0,1] = [0,1] \Rightarrow \text{ Case 3.3: } \Rightarrow in^0_{tR} = 1, in^0_{bR} = 1 \\
\Omega^1_{tL} \text{ vs. } \Omega^1_{tR} &: [0,3] = [0,3] \Rightarrow \text{ Case 3.3: } \Rightarrow in^1_{tL} = 1, in^1_{tR} = 1.
\end{aligned}$$

The players' in_{ab} and out_{ab} values of some entry (a,b) are stored in a (2×2) dimensional matrix $\begin{pmatrix} in^0_{ab} & out^0_{ab} \\ in^1_{ab} & out^1_{ab} \end{pmatrix}$. The next figure shows the in and out values in the root node:

$\emptyset$	L	R
t	$\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$
b	$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$

Figure 4.11.: *in* and *out* values in the root node.

Since (t,L) is the first entry with $out^0_{tL} = out^1_{tL} = 0$ and $in^0_{tL} + in^1_{tL} = 2$, GetPivot *returns (t,L) as the pivot entry of the root node.*

The next lemma shows that the procedure of the pivot computation GetPivot is equivalent to the pure Nash equilibrium computation in the terminal nodes.

Lemma 4.3.9. *In a terminal input node $h \neq 0$ the return value of function* GetPivot(h) *is either a pure Nash equilibrium (NE) or 0.* GetPivot(h) *returns 0 if and only if no pure Nash equilibrium exists in h.*

Proof. 1) GetPivot(h) *returns $(x,y) \Rightarrow (x,y)$ is a NE in h.*

Let us assume that GetPivot(h) returns the pivot (x,y) and (x,y) is *not* a Nash equilibrium in h. Then there is an entry $(x',y') \in T_h$ such that

$$\begin{aligned}
u_0(x,y) < u_0(x',y') \text{ with } x \neq x',\ y = y' \\
\text{or } u_1(x,y) < u_1(x',y') \text{ with } x = x',\ y \neq y'.
\end{aligned}$$

If $u_i(x,y) < u_i(x',y')$, then player i's *out* value is $out^i_{xy} > 0$ which is a contradiction to the definition of the pivot entry (x,y) since the *out* values of the pivot entry must be zero for both players.

2) GetPivot(h) *returns $0 \Leftrightarrow$ There is no pure NE in h.*

If GetPivot(h) returns 0, then for every entry $(x, y) \in T_h$ exists $j \in \{0, 1\}$ such that $out^j_{xy} > 0$ and there is an entry $(x', y') \in T_h$ such that

$$u_0(x, y) < u_0(x', y') \text{ with } x \neq x',\ y = y'$$
$$\text{or } u_1(x, y) < u_1(x', y') \text{ with } x = x',\ y \neq y'.$$

Therefore there is no pure equilibrium in node h. The '$\Leftarrow$' direction of *2)* is analogous. □

If we compute the pivot entry in a non-terminal node, then we want to test if the pivot is already a pure Nash equilibrium. This test is implemented in the function TestPivot which has two input parameters. The first one is a node $h \in H$ and the second one is a pivot (x, y). Assume that $A^0_h = \{1, \ldots, m\}$ and $A^1_h = \{1, \ldots, n\}$. The function tests if the pivot entry (x, y) is a Nash equilibrium in the input node h. The pivot (x, y) is a pure Nash equilibrium in h with expected payoff $U_0(x, y)$ if player 0 has no incentive to deviate from (x, y) to (x', y) with $x' \in A^0_h \backslash \{x\}$ and player 1 has no incentive to deviate to (x, y') with $y' \in A^1_h \backslash \{y\}$. To test the optimality in (x, y) for player 0, compare for all $1 \leq x' \leq m$ the expected payoff $U_0(x, y)$ with the maximum $\max \Omega^0_{(x',y)}$ of the utility interval $\Omega^0_{(x',y)}$:

If $U_0(x, y) < \max \Omega^0_{(x',y)}$ and entry (x', y) has a child, then compute a subgame perfect equilibrium b with the minimal utility $U_0(b \,|\, \sigma(x', y))$ for player 0 in the subgame rooted at entry (x', y). If $U_0(x, y) < U_0(b \,|\, \sigma(x', y))$, then replace the pivot (x, y) with the entry (x', y). The comparison procedure for player 1 is analogous for all $1 \leq y' \leq n$.

If TestPivot changes the pivot from (x, y) to (x', y') after the comparison procedure for player 1, then recursively repeat the procedure with the pivot (x', y'). To avoid the infinite loops, stop the procedure and return 0 if it has been run more than $\min\{|A^0_h|, |A^1_h|\}$ times. After this step one of the following cases is true:

- TestPivot(h,(x, y)) returns $(x', y') \in E_h \cup T_h \ \Rightarrow\ (x', y')$ is a NE in h;
- TestPivot(h,(x, y)) returns $0 \ \Rightarrow$ there is no pure NE in node h.

The pseudocode of the TestPivot function is given Algorithm 8.

Algorithm 8 Pivot Test

```
1: function TestPivot(h, (x, y))
2:     counter ← counter + 1                          ▷ Prevents endless loops.
3:     if counter > min{|A_h^0|, |A_h^1|} ∨ (x, y) = 0 then
4:         return 0
5:     end if
6:     if h ∈ H_T then                    ▷ Do not test the pivot in terminal nodes.
7:         return (x, y)
8:     end if
9:     for x' ← 1, ..., |A_h^0| do                    ▷ Pivot test for player 0.
10:        if x' ≠ x ∧ U_0(x, y) < max Ω^0_{(x',y)} then
11:            if (x', y) ∈ E_h then
12:                b ←GetEquilibrium(σ(x', y), 0, false)
13:                if U_0(x, y) < U_0(b | σ(x', y)) ∨ (x', y) ∈ T_h then
14:                    (x, y) ← (x', y)
15:                end if
16:            end if
17:        end if
18:    end for
19:    for y' ← 1, ..., |A_h^1| do                    ▷ Pivot test for player 1.
20:        if y' ≠ y ∧ U_1(x, y) < max Ω^1_{(x,y')} then
21:            if (x, y') ∈ E_h then
22:                b ←GetEquilibrium(σ(x, y'), 1, false)
23:                if U_1(x, y) < U_1(b | σ(x, y')) ∨ (x, y') ∈ T_h then
24:                    (x, y) ← (x, y')
25:                end if
26:            end if
27:        end if
28:    end for
29:    return (x, y)
30: end function
```

Now we can put all pieces together and present the IH algorithm. We implement it in the function GetPBS. It consists of two main procedures GetPivot (Algorithm 7) and TestPivot (Algorithm 8). They are applied recursively on the nodes starting with the input node h and then to its successors which are determined by the support of the computed partial behavioral strategy. The input of GetPBS is a node h and the output is a partial behavioral strategy of the subgame rooted at the node h. The pseudocode is given in Algorithm 9.

Algorithm 9 PBS Computation with the IH

```
1: function GetPBS(h)
2:     (x, y) ← GetPivot(h)
3:     if (x, y) ∈ E_h then                          ▷ The pivot exists and has a child.
4:         b'(σ(x, y)) ←GetPBS(σ(x, y))
5:     end if
6:     if (x, y) ≠ 0 then
7:         b'(h)((x, y)) ← 1, b'(h)(¬(x, y)) ← 0
8:     end if
9:     (x', y') ←TestPivot(h,b',(x, y))
10:    if (x, y) ≠ (x', y') ∧ (x', y') ≠ 0 then                          ▷ New pivot.
11:        if (x', y') ∈ E_h then
12:            b'(σ(x', y')) ←GetPBS(σ(x', y'))
13:        end if
14:        b'(h)((x', y')) ← 1, b'(h)(¬(x', y')) ← 0
15:        return b'(h)
16:    else if (x', y') = 0 then                          ▷ No pivot found in h
17:        for i ← 1, ..., |A_h^0| do
18:            for j ← 1, ..., |A_h^1| do
19:                if (i, j) ∈ E_h then
20:                    b'(σ(i, j)) ←GetPBS(σ(i, j))
21:                    u(i, j) ← U(b' | σ(i, j))
22:                end if
23:            end for
24:        end for
25:        b'(h) ←GetEquilibrium(h)
26:    end if
27:    return b'(h)
28: end function
```

Example 4.3.10. *Let us compute a pbs of the game from Example 4.3.1 with Algorithm 9.*

GetPivot($\emptyset$) *returns* (t, L) *(see Example 4.3.8). Since* $(t, L) \in E_\emptyset$ *has a child, the algorithm calls itself recursively with the child-node* $\sigma(t, L) = tL$ *as the input parameter.*

GetPivot(tL) *returns* (a, X) *as the pivot. Since* (a, X) *is terminal, it is already a pure equilibrium (Lemma 4.3.9) with expected payoff* $(2, 3)$.

Now we are back in the root node $\emptyset$ *and the algorithm calls* TestPivot$(\emptyset, (t, L))$. *Player* 0*'s expected payoff* $U_0(t, L) = 2$ *of the pivot is compared with the maximum of player* 0*'s utility interval* $\max \Omega^0_{(b,L)} = 1$. *Since* $2 > 1$, *the algorithm proceeds with the comparison of* $U_1(t, L) = 3$ *with* $\max \Omega^1_{(t,R)} = 3$. *Since they are equal, the procedure is*

finished and returns (t, L)*. Therefore* (t, L) *is a pure equilibrium in the root node.*

Now the main procedure GetPBS($\emptyset$) *is finished and returns a pbs* b' *with expected payoff* $(2, 3)$ *defined by*

$$b'(\emptyset)((t, L)) = 1, b'(\emptyset)(\neg(t, L)) = 0,$$
$$b'(tL)((a, X)) = 1, b'(tL)(\neg(a, X)) = 0.$$

4.3.2. Example Computation with the Interval Heuristic

We demonstrate the IH algorithm step by step on the game from Figure 4.12. The pure Nash equilibria are bold faced in the terminal nodes.

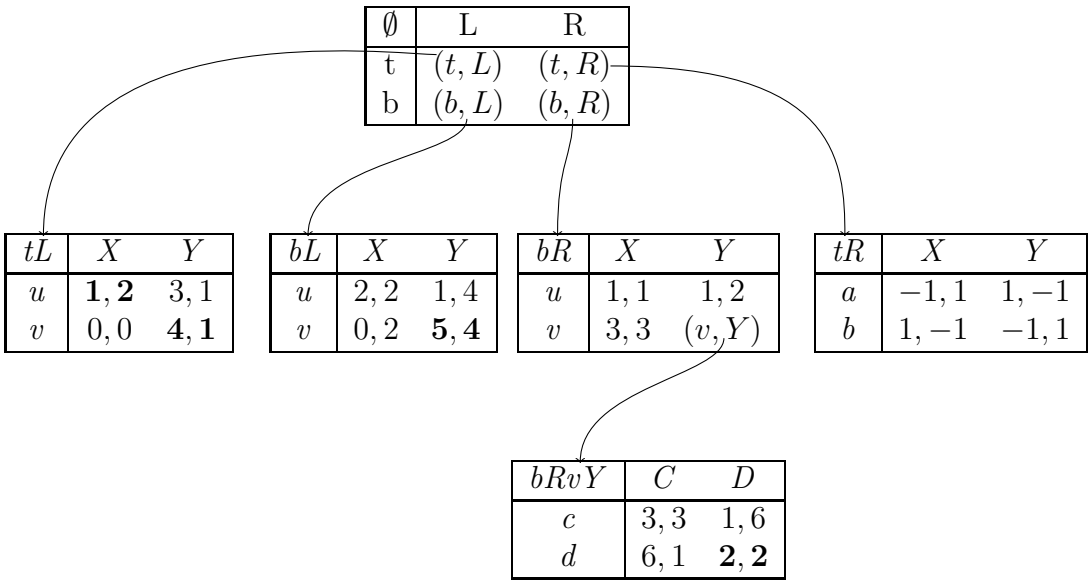


Figure 4.12.: Example Computation with the Interval Heuristic

In the first step the IH algorithm computes the utility intervals in every node.

$$\Omega^0_{tL} = [0, 4] \qquad \Omega^1_{tL} = [0, 2]$$
$$\Omega^0_{tR} = [-1, 1] \qquad \Omega^1_{tR} = [-1, 1]$$
$$\Omega^0_{bL} = [0, 5] \qquad \Omega^1_{bL} = [2, 4]$$
$$\Omega^0_{bRvY} = [1, 6] \qquad \Omega^1_{bRvY} = [1, 6]$$
$$\Omega^0_{bR} = [1, 6] \qquad \Omega^1_{bR} = [1, 6]$$
$$\Omega^0_{\emptyset} = [-1, 6] \qquad \Omega^1_{\emptyset} = [-1, 6]$$

In the next step the algorithm compares the utility intervals of the entries in the root node and computes the *in* and *out* values. The players' in_{ab} and out_{ab} values of some

entry (a, b) are stored in a (2×2) dimensional matrix $\begin{pmatrix} in^0_{ab} & out^0_{ab} \\ in^1_{ab} & out^1_{ab} \end{pmatrix}$. The next figure shows the *in* and *out* values in the root node:

$\emptyset$	L	R
t	$\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$
b	$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$

Figure 4.13.: *in* and *out* values in the root node.

The IH algorithm returns (b, R) as the pivot entry in the root node since:

$$out^0_{tL} = out^1_{tL} = 0 \text{ and}$$
$$in^0_{tL} + in^1_{tL} = \max_{(a,b) \in E_\emptyset} in^0_{ab} + in^1_{ab} = 2.$$

In the next step the algorithm computes the pivot entry in the node bR. The *in* and *out* values are shown in the next figure.

bR	X	Y
u	$\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$
v	$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$

Figure 4.14.: *in* and *out* values in the node bR.

The algorithm returns (v, Y) as the pivot entry in the node bR since its *out* values are both zero and the sum of the *in* values is maximal: $in^0_{tL} + in^1_{tL} = 2$.

Therefore the procedure is repeated in the node $bRvY$ which is a terminal node. The IH algorithm returns the pivot entry (d, D) which is a pure Nash equilibrium (Lemma 4.3.9) with expected payoff $(2, 2)$. The expected payoff $(2, 2)$ is then propagated to the parent entry (v, Y) in the node bR and now the algorithm tests if the pivot entry (v, Y) is a pure equilibrium there: it compares player 1's expected payoff in (v, Y) to player 1's payoff in the terminal entry (v, X). Since $2 < 3$, the algorithm changes the pivot entry from (v, Y) to (v, X) and compares player 0's payoff in (v, X) to player 0's payoff in (u, X). Since $3 > 1$, the algorithm returns the pivot entry (v, X) with payoff $(3, 3)$ in the node bR. The payoff $(3, 3)$ is then propagated to the corresponding entry (b, R) of the root node.

In the next step the algorithm tests if the pivot entry (b, R) with expected payoff $(3, 3)$ is a pure Nash equilibrium in the root node. Player 1's expected payoff in the pivot entry

(b, R) is not greater than the maximum of the utility interval Ω^1_{bL} of the entry (b, L). Therefore the algorithm computes the equilibrium $b'(bL)$ in the node bL such that player 1's expected payoff is minimal. It finds there the only equilibrium (v, Y) with expected payoff $(5, 4)$. This payoff is propagated to the parent entry (b, L) in the root node. Since $3 < 4$, the algorithm changes the pivot from (b, R) to (b, L). Now it compares player 0's expected payoff in (b, L) with the maximum of the utility interval Ω^0_{tL}. Since they are both equal, the algorithm returns the pivot (b, L) in the root node. Now the IH algorithm is finished and it has found the following partial behavioral strategy b' with expected payoff $(5, 4)$:

$$\begin{aligned} b'(\emptyset)((b, L)) = 1,\, b'(\emptyset)(\neg(b, L)) = 0,\\ b'(bL)((v, Y)) = 1,\, b'(bL)(\neg(v, Y)) = 0,\\ b'(bR)((v, X)) = 1,\, b'(bR)(\neg(v, X)) = 0,\\ b'(bRvY)((d, D)) = 1,\, b'(bRvY)(\neg(d, D)) = 0. \end{aligned}$$

4.3.3. Analysis of the Interval-Heuristic Algorithm

In this section we analyse the interval-heuristic algorithm (Algorithm 9) and show that it is sound and complete. The next lemma shows that a partial behavioral strategy returned by the IH algorithm is a part of a subgame perfect equilibrium.

Lemma 4.3.11. *Let b' be a partial behavioral strategy computed by the IH algorithm (Algorithm 9). Then there is a SPE b^* such that $b^*(h) = b'(h)$ for all $h \in \text{supp}(b')$.*

Proof. Let $\Gamma = \langle H, (A^0_h)_{h\in H}, (A^1_h)_{h\in H}, (E_h)_{h\in H}, (T_h)_{h\in H}, \sigma, u\rangle$ be a 2-player extensive game with simultaneous moves. We prove this lemma analogically to Theorem 4.2.1 using induction over the depth of the subgame trees.
Step 1 ($d(h) = 0$): Let $h^0 \in H$ such that $d(h^0) = 0$. Therefore h^0 is a terminal node. We have shown in Lemma 4.3.9 that the function GetPivot(h^0) either returns a pure Nash equilibrium of the terminal node h^0 if it exists there or 0 otherwise. If GetPivot(h^0) returns 0, then the main function GetPBS(h^0) computes a Nash equilibrium of h^0 using mixed integer programming. Therefore the IH algorithm finds a pbs $b'(h^0)$ which is a Nash equilibrium in the input extensive game with simultaneous moves which consists only of the terminal node h^0. The condition on a Nash equilibrium in mixed strategies (Definition 2.1.9) implies the sequential rationality (Definition 4.1.19):

$$U_i(b'(h^0)|\, h^0) = U_i(b'_i(h^0), b'_{1-i}(h^0)) = \max_{b'_i(h^0)\in\Delta(A^i_{h^0})} U_i(b'_i(h^0), b_{(1-i}(h^0)).$$

Step 2 ($n \to n+1$): Now assume that the IH algorithm finds a pbs $b'(h^n)$ in all input nodes h^n with $d(h^n) \leq n$ such that there is a SPE b^* with $b^*(h) = b'(h)$ for all $h \in \text{supp}(b')$. We want to show that it is also true for an input node h^{n+1} with $d(h^{n+1}) = n + 1$. We have two following cases:

- **(Pure NE in h^{n+1})** If GetPBS(h^{n+1}) finds the pivot entry (x, y) in h^{n+1} and verifies that (x, y) is a pure equilibrium in h^{n+1}, then we are finished since we can determine b^* in h^{n+1} with $b^*(h^{n+1}) = b'(h^{n+1})$ and we have assumed that b^* is sequentially rational in all nodes h^n with $d(h^n) \leq n$ and $b^*(h) = b'(h)$ for all $h \in \text{supp}(b')$.

- **(Mixed NE in h^{n+1})** If there is no pivot in node h^{n+1}, then GetPBS(h^{n+1}) computes a pbs $b'(\sigma(c,d)) = (b'_0(\sigma(c,d)), b'_1(\sigma(c,d)))$ in every child-node $\sigma(c,d)$ with $(c,d) \in E_{h^{n+1}}$. Then player i's expected payoff $U_i(b' \mid \sigma(c,d))$ in the child-node $\sigma(c,d)$ is propagated to the corresponding entry $(c,d) \in E_{h^{n+1}}$. After that a Nash equilibrium $(b'_0(h^{n+1}), b'_1(h^{n+1}))$ is computed there using mixed integer programming. Let us assume that GetPBS(h^{n+1}) returns a pbs b' such that there is no subgame perfect equilibrium b^* with $b^*(h) = b'(h)$ for all $h \in \text{supp}(b')$. Since $d(\sigma(c,d)) \leq n$, b' must be sequentially rational in every subgame rooted at $\sigma(c,d)$ for all $(c,d) \in E_{h^{n+1}}$. Therefore the sequential rationality of b' does not hold in the node h^{n+1}. But this would be a contradiction to the fact that the expected payoff of $b'_i(h^{n+1})$ must be maximal in h^{n+1} for $i \in \{0,1\}$ since $(b'_0(h^{n+1}), b'_1(h^{n+1}))$ is a mixed NE:

$$U_i(b' \mid h^{n+1}) = U_i(b'_i(h^{n+1}), b'_{1-i}(h^{n+1})) = \max_{b_i(h^{n+1}) \in \Delta(A^i_h)} U_i(b_i(h^{n+1}), b'_{1-i}(h^{n+1})).$$

  Thus (b'_0, b'_1) is sequentially rational in h^{n+1} and therefore there is a behavioral strategy b^* such that b^* is a subgame perfect equilibrium in Γ and $b^*(h) = b'(h)$ for all $h \in \text{supp}(b')$.

□

Corollary 4.3.12. (Soundness and Completeness of the IH Algorithm)
The IH algorithm (Algorithm 9) is sound and complete.

Proof. The soundness follows directly from Theorem 4.3.11. The IH algorithm is complete since a Nash equilibrium exists in each node (Lemma 2.1.11) and is computed with the complete algorithms for finding a NE. Thus the baseline algorithm always finds a pbs which is a part of a subgame perfect equilibrium in the input 2-player extensive game with simultaneous moves. □

In the worst case the IH algorithm finds no pivot in each node. Then it solves a MIP in each node and performs like the baseline algorithm (Algorithm 5). In the next section we show some experimental results of the IH algorithm and compare it with the baseline algorithm.

4.3.4. Interval-Heuristic Algorithm in Practice

In this section we compare the runtime of the baseline algorithm (Algorithm 5) with the IH algorithm (Algorithm 9). Both algorithms are implemented in *C++*. We test them

on 100 randomly generated extensive games with simultaneous moves. The tests are taken similar to the tests in Section 4.2.3. The random games are generated with two constants: a and d. The constant a determines the number of actions available for every player $i \in \{0, 1\}$ in each node. The constant d determines the depth of the game tree. The payoffs at the terminal entries are random numbers uniformly distributed between -1 and 1. In contrast to the tests in Section 4.2.3 each random game has the constant number of nodes: $|H| = \frac{(a^2)^{d+1}-1}{a^2-1}$ (see Lemma 4.1.11). Table 4.3 compares the minimal ($\min$ IH / $\min$ BL), the average ($\varnothing$ IH / $\varnothing$ BL) and the maximal ($\max$ IH / $\max$ BL) runtime in seconds of the interval-heuristic (IH) and the baseline (BL) algorithms. The a and d values are represented in the vector (a, d) in the first column. The second column contains the number of nodes $|H|$ which depends on a and d.

	$\lvert H\rvert$	$\min$ **IH**	$\min$ **BL**	$\varnothing$ **IH**	$\varnothing$ **BL**	$\max$ **IH**	$\max$ **BL**
$(2,1)$	5	0.000509	0.000382	0.0015443	0.0013296	0.005963	0.008979
$(2,2)$	21	0.001188	0.00199	0.0029126	0.00629	0.015339	0.013094
$(2,3)$	85	0.003222	0.009599	0.006555	0.022944	0.022267	0.034232
$(2,4)$	341	0.01037	0.065686	0.014904	0.091228	0.032477	0.12459
$(2,5)$	1365	0.036613	0.28522	0.04652	0.37423	0.078904	0.45508
$(2,6)$	5461	0.074494	0.69772	0.11576	1.006	0.21397	1.5966
$(2,7)$	21845	0.27043	2.6918	0.43457	3.8134	0.91563	6.2575
$(3,1)$	10	0.000551	0.000636	0.0024934	0.0025174	0.010104	0.009255
$(3,2)$	91	0.003562	0.017587	0.0078991	0.03174	0.018246	0.048896
$(3,3)$	820	0.023331	0.20434	0.033523	0.29627	0.085498	0.37996
$(3,4)$	7381	0.16045	1.8315	0.20789	2.3332	0.53461	3.1056
$(3,5)$	66430	1.3939	16.2054	2.7884	29.6404	8.0111	48.984

Table 4.3.: Runtime Comparison between the Baseline and the IH Algorithms.

The next figure shows the graphs of the mean runtimes of the IH and the baseline algorithms for each pair of a and d.

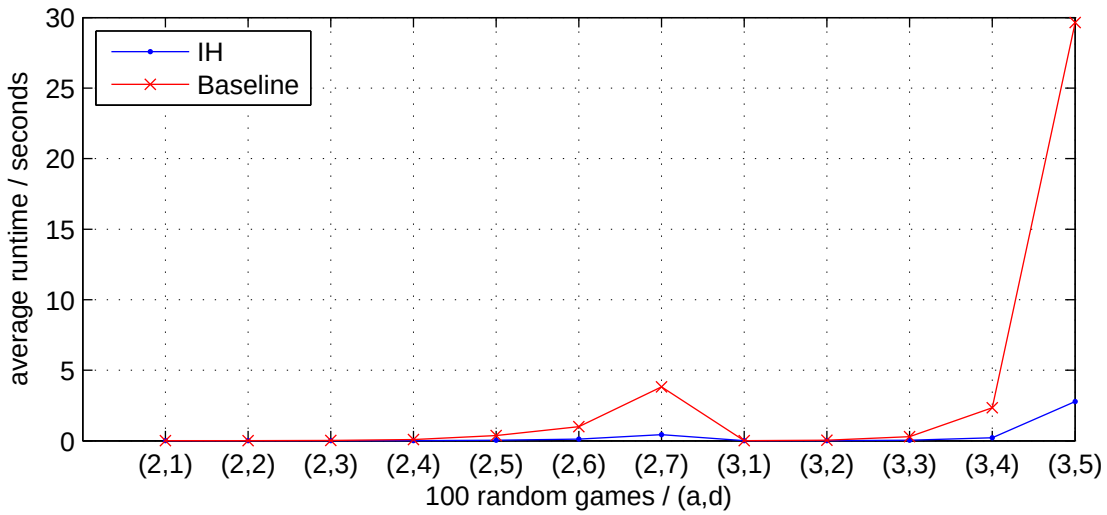


Figure 4.15.: Runtime tests

The particular tests are visualized in Appendix A. The tests validate that the IH algorithm outperforms the baseline algorithm in the runtime. This result seems natural since the IH algorithm is constructed in the way that it skips parts of the game tree using the interval-heuristic. The IH algorithm is especially useful for extensive games with simultaneous moves with a high number of nodes. However it returns not a full SPE but just a partial behavioral strategy which is a part of a subgame perfect equilibrium. It can be used to quickly find the expected payoff of a SPE in an extensive game with simultaneous moves with the high number of nodes. In the next chapter we test the IH and the Baseline algorithm on a version of simultaneous Tic-Tac-Toe game. We see that the IH algorithm finds a pbs which corresponds to a possible game course. Thus a possible way to use the IH algorithm could be an implementation in general game playing, especially in games with simultaneous moves.

5. Solving a Simultaneous Tic-Tac-Toe Game

In addition to the tests in Sections 4.2.3 and 4.3.4 we provide some experimental results based on a version of a real game. In this chapter we test the baseline algorithm and the IH algorithm on a version of the simultaneous Tic-Tac-Toe game from the General Game Playing (GGP).

5.1. Simultaneous Tic-Tac-Toe Game

Tic-Tac-Toe is a 2-player game played on a (3×3) field. The players are called 'player X' and 'player O'. They take turns in marking a field with 'X' or with 'O' respectively. The goal for each player is to place three of his marks in a horizontal, vertical or diagonal line. Simultaneous Tic-Tac-Toe game is a variant of Tic-Tac-Toe game. The difference is that both players play simultaneously. If they both try to mark the same field, then this field remains unmarked. The payoffs are determined in the following way: If only one player marks the line, then he receives 100 and the opposite player receives 0. If no player marks the line or both of them do, then they both receive 50. The full description of simultaneous Tic-Tac-Toe game in Game Description Language[1] (GDL) format is given in Appendix B. The next figure shows a possible course of a simultaneous Tic-Tac-Toe game if the players choose the following sequence of action profiles:

$$(11, 11) \rightarrow (11, 12) \rightarrow (22, 13) \rightarrow (21, 33) \rightarrow (31, 23).$$

	1	2	3
1			
2			
3			

	1	2	3
1			
2			
3			

	1	2	3
1	X	O	
2			
3			

	1	2	3
1	X	O	O
2		X	
3			

	1	2	3
1	X	O	O
2	X	X	
3			O

	1	2	3
1	X	O	O
2	X	X	O
3	X		O

Figure 5.1.: Possible Course of a Simultaneous Tic-Tac-Toe Game.

At the end of the game player X and player O mark the lines of 'X' and 'O' respectively and therefore they both receive the payoff 50.

We implement simultaneous Tic-Tac-Toe as an extensive game with simultaneous moves $\Gamma = \langle H, (A_h^0)_{h \in H}, (A_h^1)_{h \in H}, (E_h)_{h \in H}, (T_h)_{h \in H}, \sigma, u \rangle$ in the following way:

[1] See the Game Description Language specification [14].

Since simultaneous Tic-Tac-Toe is a game of perfect recall (see Definition 3.2.4), every node has the unique history leading from the root node to it. Therefore we identify the nodes with their histories. If both players choose different actions, then they cannot play these actions in the next turn again. Let us denote the set of nodes after the k-th simultaneous move where the players choose different actions in their last move with H_k and the set of nodes after the k-th simultaneous move where they choose the same action in the last move with H_k^* for for $k > 0$.

For $k = 0$, $H_0 := \{\langle\rangle\}$ consists only of the root node $\langle\rangle$. The action sets are defined with

$$A^0_{\langle\rangle} = A^1_{\langle\rangle} := \{xy \,|\, x, y \in \{1, 2, 3\}\}.$$

Every entry of the root node is non-terminal. Therefore we have

$$E_{\langle\rangle} := A^0_{\langle\rangle} \times A^1_{\langle\rangle},\ T_{\langle\rangle} = \emptyset.$$

For $k > 0$ define

$$\begin{aligned} H_k := \{\langle (a^0_0, a^1_0), \ldots, (a^0_k, a^1_k)\rangle \,|\, & a^0_k \neq a^1_k, \\ & a^i_t := x^i_t y^i_t,\ x^i_t, y^i_t \in \{1, 2, 3\} \\ & \forall\, i \in \{0, 1\},\, t \in \{0, \ldots, k\}\}. \end{aligned}$$

Let $\gamma := \langle (a^0_0, a^1_0), \ldots, (a^0_k, a^1_k)\rangle \in H_k$ with $k > 0$. We define player i's set of actions A^i_γ in γ with:

$$A^i_\gamma := A^i_{\langle (a^0_0, a^1_0), \ldots, (a^0_{k-1}, a^1_{k-1})\rangle} \backslash \{a^0_k, a^1_k\}.$$

Where for $k = 1$ the history $\langle (a^0_0, a^1_0), \ldots, (a^0_{k-1}, a^1_{k-1})\rangle$ is the initial history $\langle\rangle$.

To determine the terminal entries T_γ in γ we define the 'line' function $L^i(\gamma) : A^i_\gamma \to \{0, 1\}$ which determines if player i marks the line of 'X' or 'O' respectively if he takes an action $a^i_{k+1} = x^i_{k+1} y^i_{k+1}$ with $x^i_{k+1}, y^i_{k+1} \in \{1, 2, 3\}$ and given the history $\gamma = \langle (a^0_0, a^1_0), \ldots, (a^0_k, a^1_k)\rangle$:

$$L^i(\gamma)(x^i_{k+1} y^i_{k+1}) := \begin{cases} 1, & \exists\, u, v, w \in \{0, \ldots, k+1\} \text{ with } u < v < w : \\ & x^i_u = x^i_v = x^i_w (\text{horizontal line}) \vee \\ & y^i_u = y^i_v = y^i_w (\text{vertical line}) \vee \\ & (\text{diagonal lines}): \\ & x^i_u y^i_u, x^i_v y^i_v, x^i_w y^i_w \in \{11, 22, 33\} \wedge x^i_u y^i_u \neq x^i_v y^i_v \neq x^i_w y^i_w \neq x^i_u y^i_u \vee \\ & x^i_u y^i_u, x^i_v y^i_v, x^i_w y^i_w \in \{13, 22, 31\} \wedge x^i_u y^i_u \neq x^i_v y^i_v \neq x^i_w y^i_w \neq x^i_u y^i_u, \\ 0, & \text{otherwise.} \end{cases}$$

Now we can define the set of the terminal entries T_γ in $\gamma \in H_k$ with $k > 0$:

$$T_\gamma := \{(a^0_{k+1}, a^1_{k+1}) \in A^0_\gamma \times A^1_\gamma \,|\, a^0_{k+1} \neq a^1_{k+1} \wedge (L^0(\gamma)(a^0_{k+1}) = 1 \vee L^1(\gamma)(a^1_{k+1}) = 1)\}.$$

Thus the set of the non-terminal entries is determined by:

$$E_\gamma := (A^0_\gamma \times A^1_\gamma) \backslash T_\gamma.$$

We define player i's utility function $u_i(a^0_{k+1}, a^1_{k+1})$ for a terminal entry $(a^0_{k+1}, a^1_{k+1}) \in T_\gamma$ in the following way :

$$u_i(a^0_{k+1}, a^1_{k+1}) = \begin{cases} 100, & \text{if } L^i(\gamma)(a^i_{k+1}) = 1 \wedge L^{1-i}(\gamma)(a^{1-i}_{k+1}) = 0, \\ 50, & \text{if } (L^i(\gamma)(a^i_{k+1}) = 1 \wedge L^{1-i}(\gamma)(a^{1-i}_{k+1}) = 1) \vee \\ & (L^i(\gamma)(a^i_{k+1}) = 0 \wedge L^{1-i}(\gamma)(a^{1-i}_{k+1}) = 0), \\ 0, & \text{if } L^i(\gamma)(a^i_{k+1}) = 0 \wedge L^{1-i}(\gamma)(a^{1-i}_{k+1}) = 1. \end{cases}$$

Now we take a closer look at the case where both players choose the same action. Our definition of extensive game with simultaneous moves requires that the game is an acyclic tree. We cannot implement the condition that the field remains unmarked, since the child node of such entry would be the node itself. We change the condition in the following way: If both players choose the same action in a node $\gamma \in H_k$ for $k \geq 0$, then it leads to a special so-called 'middle' node where player i keeps his choice and the opposite player $1 - i$ has to choose a different action. In order to construct an equitable game, we let player 0 and player 1 keep his choice in turns. The first player who keeps his choice is determined by chance at the beginning of the game construction.

For $k \geq 0$ define

$$\begin{aligned} H^*_k := \{\langle (a^0_0, a^1_0), \ldots, (a^0_k, a^1_k)\rangle \,|\, & a^0_k = a^1_k, \\ & a^i_t := x^i_t y^i_t,\ x^i_t, y^i_t \in \{1,2,3\} \\ & \forall\, i \in \{0,1\},\, t \in \{1,\ldots,k\}\}. \end{aligned}$$

Let $\gamma^* := \langle (a^0_0, a^1_0), \ldots, (a^i_k, a^i_k)\rangle \in H^*_k$. We define the set of actions of player i and $1 - i$ in γ^* in the following way:

$$A^i_{\gamma^*} := \{a^i_k\},\ A^{1-i}_{\gamma^*} = A^{1-i}_{\langle (a^0_0,a^1_0),\ldots,(a^0_{k-1},a^1_{k-1})\rangle} \backslash \{a^i_k\}.$$

Where for $k = 1$ the history $\langle (a^0_0, a^1_0), \ldots, (a^0_{k-1}, a^1_{k-1})\rangle$ is the initial history $\langle\rangle$.

The terminal and non-terminal entries of $\gamma^* \in H^*_k$ are defined in the same way as for a node $\gamma \in H_k$ using the 'line' function $L^i(\gamma^*)$ for player $i \in \{0,1\}$.

The set of the nodes is determined by H_k and H^*_k: $H := \bigcup_{k\geq 0} H_k \cup H^*_k$. At the end we need to define the successor function $\sigma : \bigcup_{h\in H} E_h \to H$ to complete our definition of

simultaneous Tic-Tac-Toe game. Let $\gamma := \langle (a_0^0, a_0^1), \ldots, (a_k^0, a_k^1) \rangle \in H$ and $(a_{k+1}^0, a_{k+1}^1) \in E_\gamma$, then:

$$\sigma(a_{k+1}^0, a_{k+1}^1) := \langle (a_0^0, a_0^1), \ldots, (a_k^0, a_k^1), (a_{k+1}^0, a_{k+1}^1) \rangle$$

Figure 5.2 shows the part of the game tree of simultaneous Tic-Tac-Toe which is induced by the game course from Figure 5.1. To reduce the notations we define the following histories:

$$\begin{aligned}
\gamma_0 &:= \langle \rangle \\
\gamma_1^* &:= \langle (11, 11) \rangle \\
\gamma_1 &:= \langle (11, 11), (11, 12) \rangle \\
\gamma_2 &:= \langle (11, 11), (11, 12), (22, 13) \rangle \\
\gamma_3 &:= \langle (11, 11), (11, 12), (22, 13), (21, 33) \rangle
\end{aligned}$$

The entries which are played in the game course from Figure 5.1 are bold faced. The empty entries are non-terminal. The tables on the right side of the nodes show the game state after the actions are simultaneously played in the corresponding node.

γ_0	11	12	13	21	22	23	31	32	33
11	$\langle (\mathbf{11}, \mathbf{11}) \rangle$	$\langle (11, 12) \rangle$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\langle (11, 33) \rangle$
12	$\langle (12, 11) \rangle$	$\langle (12, 12) \rangle$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\langle (12, 33) \rangle$
13	$\vdots$	$\vdots$	$\ddots$						$\vdots$
21	$\vdots$	$\vdots$		$\ddots$					$\vdots$
22	$\vdots$	$\vdots$			$\ddots$				$\vdots$
23	$\vdots$	$\vdots$				$\ddots$			$\vdots$
31	$\vdots$	$\vdots$					$\ddots$		$\vdots$
32	$\vdots$	$\vdots$						$\ddots$	$\vdots$
33	$\langle (33, 11) \rangle$	$\langle (33, 12) \rangle$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\langle (33, 33) \rangle$

	1	2	3
1			
2			
3			

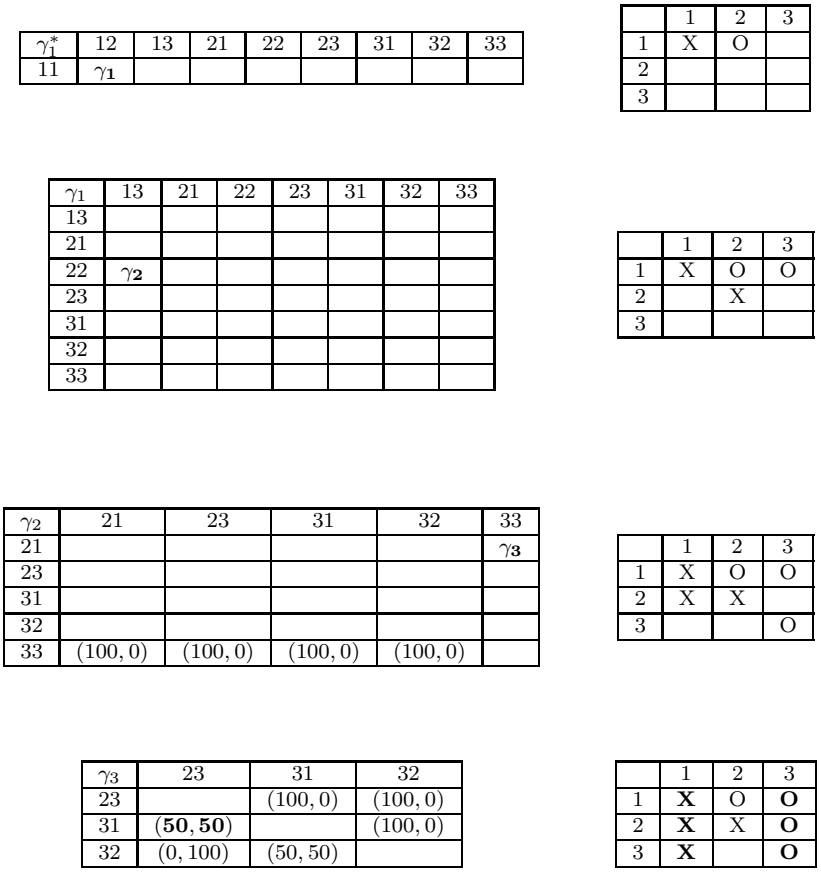


Figure 5.2.: A Part of the Simultaneous Tic-Tac-Toe Game Tree.

5.2. Finding a Solution with the Baseline and IH Algorithms

In this section we test the IH and the baseline algorithms on our implementation of the simultaneous Tic-Tac-Toe game. In our test player 1 is determined to be the first one to keep his choice if both players choose the same actions.

The IH algorithm returns the following pbs b' after 12.33 seconds. We represent b' as the sequence of action profiles which are played with probability 1:

$$(11, 11) \to (12, 11) \to (13, 13) \to (13, 21) \to (31, 22) \to (23, 33)$$

The pbs b' has expected payoff $(0, 100)$ and corresponds to the following course of simultaneous Tic-Tac-Toe game:

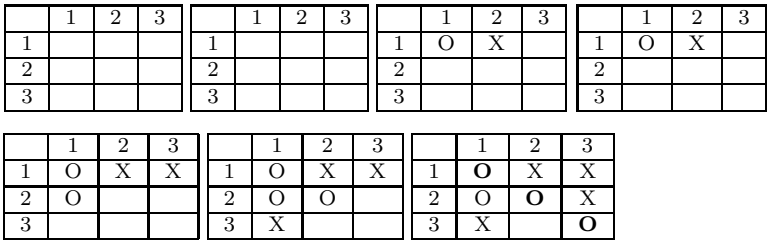


Figure 5.3.: Corresponding Course of Simultaneous Tic-Tac-Toe Game.

The baseline algorithm finds a subgame perfect equilibrium (b_0, b_1) with expected payoff $(58, 42)$ such that

$$b_0(\langle\rangle)(13) = 0.36 \qquad b_0(\langle\rangle)(31) = 0.32 \qquad b_0(\langle\rangle)(33) = 0.32$$
$$b_1(\langle\rangle)(11) = 0.64 \qquad b_1(\langle\rangle)(13) = 0.08 \qquad b_1(\langle\rangle)(33) = 0.28$$

The baseline algorithm needs 102.343 seconds to compute b.

It is remarkable that the IH algorithm finds a partial behavioral strategy b' which is pure Nash equilibrium in every node from $\mathrm{supp}(b')$, while the baseline algorithm returns a behavioral strategy b which is mixed Nash equilibrium in some nodes.

6. Conclusion and Future Work

In this thesis we focus on 2-player extensive games with simultaneous moves. We introduce a new efficient data structure for representing such games and develop two algorithms for finding a sample equilibrium based on the data structure. The baseline algorithm finds a sample subgame perfect equilibrium in extensive games with simultaneous moves using mixed integer programming. The interval-heuristic algorithm finds a partial behavioral strategy which is a part of a subgame perfect equilibrium. We compare the baseline algorithm with the gambit implementation of Lemke's algorithm and show that the baseline algorithm is much more efficient in the runtime. We also test the heuristical algorithm and compare its runtime with the runtime of the baseline algorithm.

An interesting direction for future work would be to vary the IH in the following ways:

- Other payoff distributions within the utility intervals;
- Another way of handling the situations where a node h has no pivot: e.g. trying to find the smallest support for a pbs in h instead of using all actions.
- Different tie-breaking rules for the pivot choice.

Another interesting goal would be to expand the baseline algorithm to find all equilibria in a 2-player extensive game with simultaneous moves. It would be interesting to implement the IH algorithm for General Game Playing competitions.

Bibliography

[1] M. Osborne and A. Rubinstein, *A Course in Game Theory*, MIT Press, Cambridge, MA, 1994.

[2] M. Osborne, *An Introduction to Game Theory*, Oxford University Press, Oxford, 2004.

[3] H. Peters, *Game theory: a multi-leveled approach*, Springer, 2008.

[4] N. Nisan, T. Roughgarden, E. Tardos, V. Vazirani, *Algorithmic Game Theory*, Cambridge University Press, 2007.

[5] Y. Shoham and K. Leyton-Brown, *Multiagent Systems*, Cambridge University Press, 2009.

[6] D. M. Kreps and R. Wilson, *Sequential Equilibria*, Econometrica, Vol. 50, No. 4, pp. 863–894, The Econometric Society, 1982.

[7] T. Sandholm, A. Gilpin and V. Conitzer, *Mixed-Integer Programming Methods for Finding Nash Equilibria*, American Association for Artificial Intelligence, 2005

[8] R. D. McKelvey and A. McLennan, *Computation Of Equilibria in Finite Games*, California Institute of Technology, University of Minnesota, 1994

[9] C. E. Lemke and J.J. Howson, *Equilibrium Points of Bimatrix Games*, SIAM Journal on Applied Mathematics, Vol. 12, No. 2, pp. 413–423, 1964.

[10] A. Perea and H. Peters, *Limit Consistent Solutions in Noncooperative Games*, Journal of Optimization Theory and Applications: Vol. 98, No. 1, pp. 109–129, 1998.

[11] A. Perea, *Rationality in extensive form games*, Kluwer Academic Publishers, Boston/Dordrecht/London, 2001.

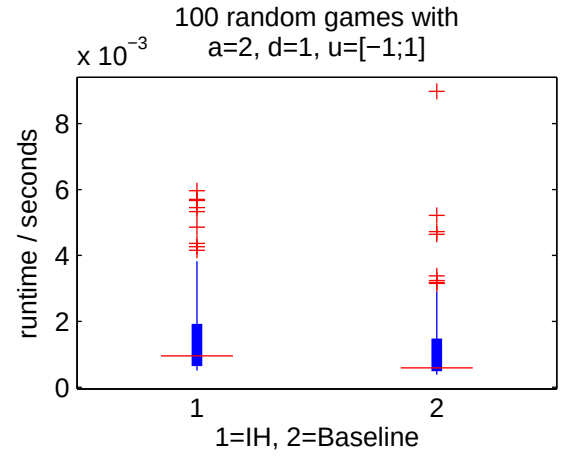
[12] D. Koller, N. Megiddo and B. Stengel, *Efficient Computation of Equilibria for Extensive Two-Person Games*, Games and Economic Behavior, Vol. 14, pp. 247–259, 1996.

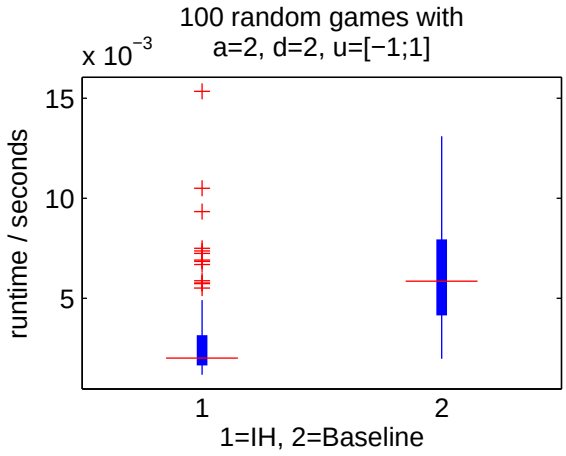
[13] Game Representation Formats for Gambit:
http://gambit-project.org/doc/formats.html

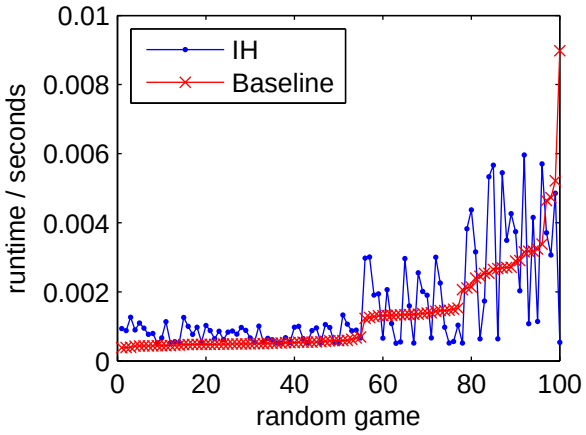
[14] N. Love, T. Hinrichs, D. Haley, E. Schkufza, M. Genesereth, *General Game Playing: Game Description Language Specification*, Stanford, CA 94305, 2008.

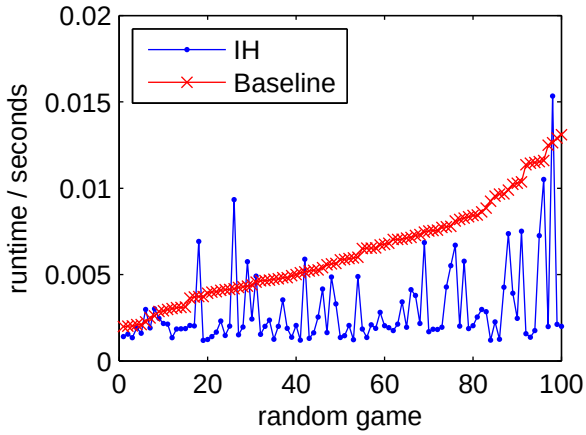
A. Runtime Comparison between the Baseline and the Interval-Heuristic Algorithms

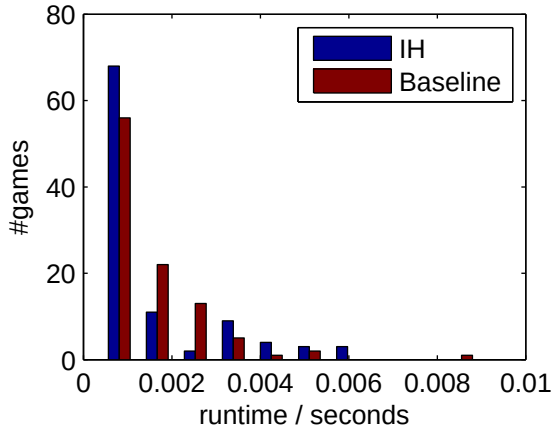
The tests are visualized in three graphs for each pair of a and d. The first graph is a box plot. The second one shows the runtime of both algorithms sorted in ascending order of the runtime of the baseline algorithm. The third graph is a distribution plot which shows several runtime frequencies.

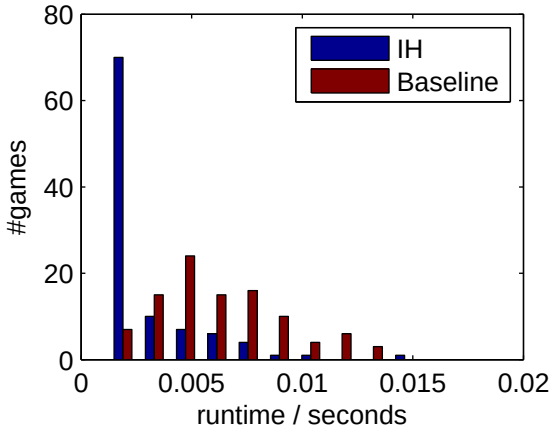

100 random games with
a=2, d=1, u=[−1;1]
x 10^−3
runtime / seconds
1=IH, 2=Baseline

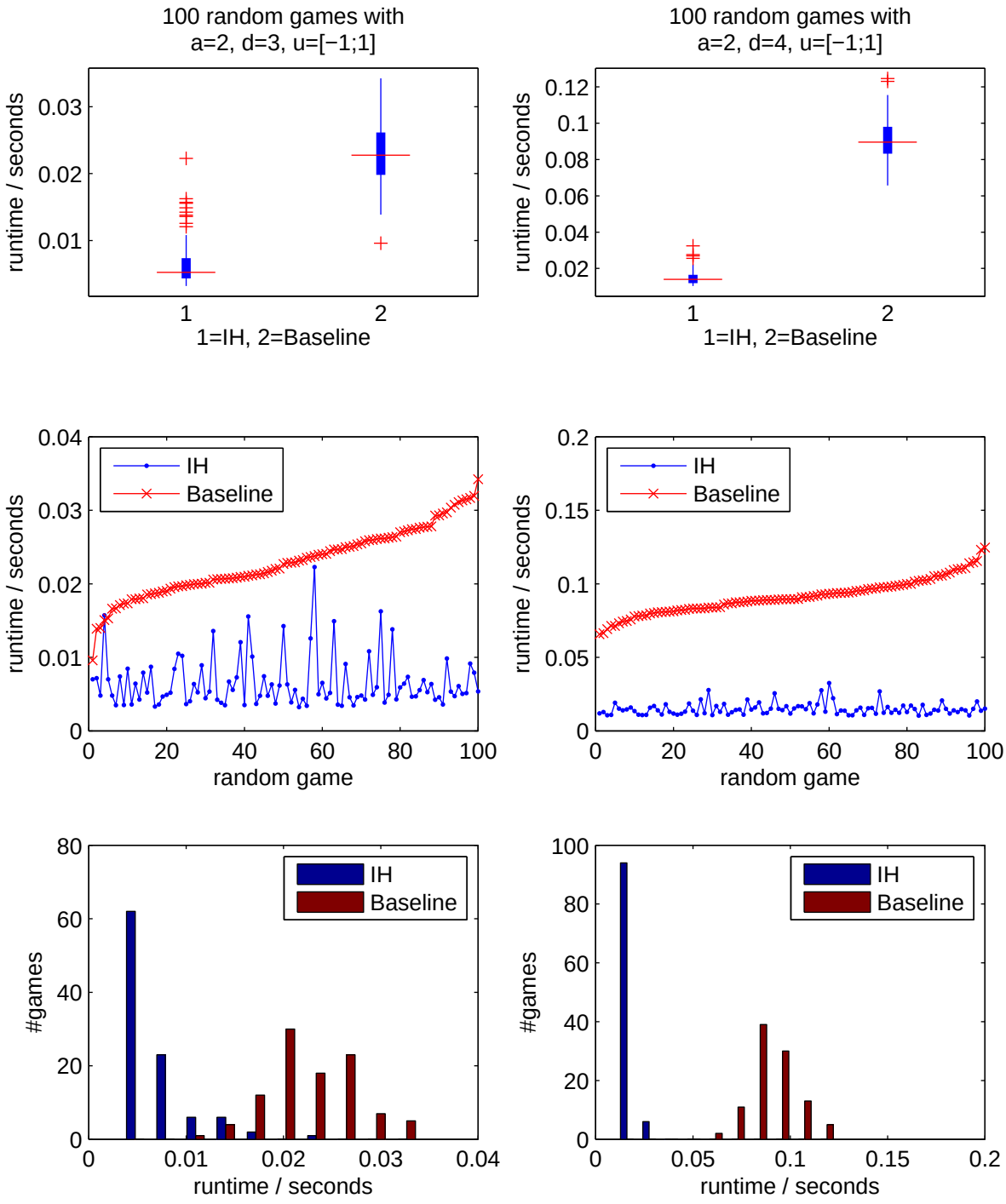

100 random games with
a=2, d=2, u=[−1;1]
x 10^−3
runtime / seconds
1=IH, 2=Baseline

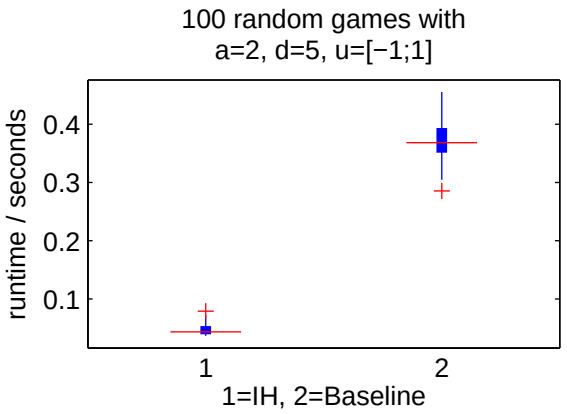

IH
Baseline
runtime / seconds
random game

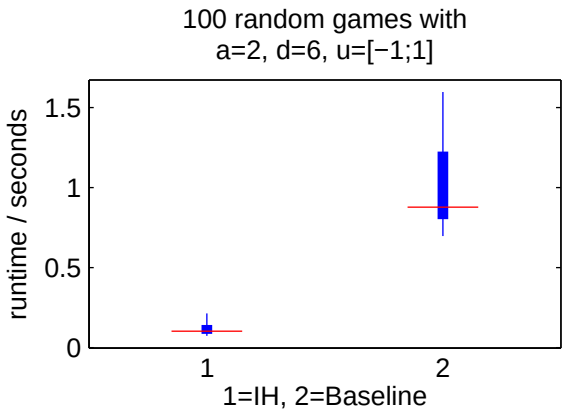

IH
Baseline
runtime / seconds
random game

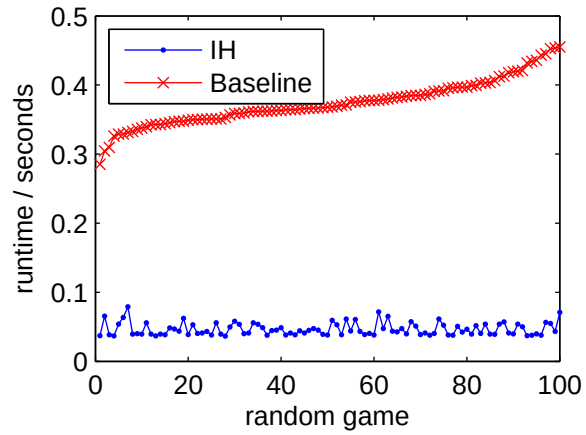

IH
Baseline
#games
runtime / seconds

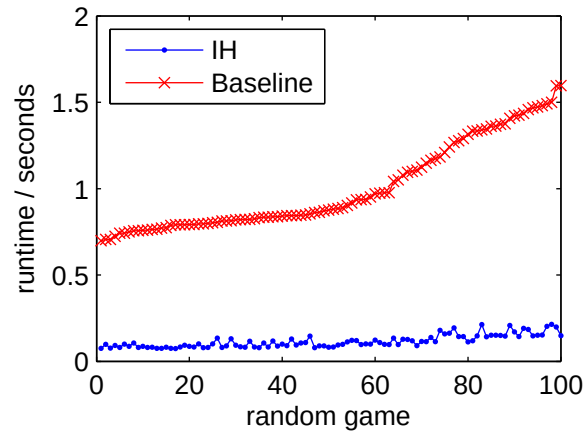

IH
Baseline
#games
runtime / seconds

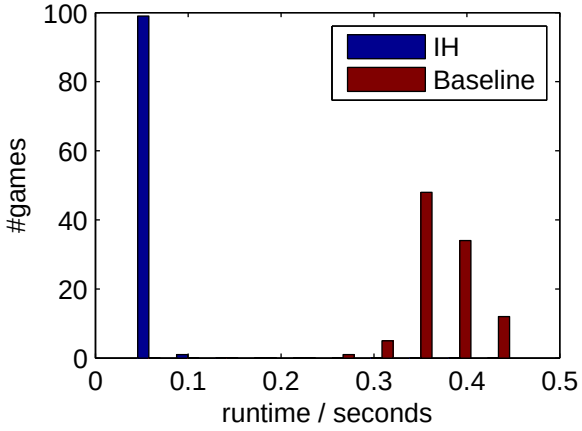

100 random games with
a=2, d=3, u=[−1;1]
runtime / seconds
0.03
0.02
0.01
1
2
1=IH, 2=Baseline
100 random games with
a=2, d=4, u=[−1;1]
runtime / seconds
0.12
0.1
0.08
0.06
0.04
0.02
1
2
1=IH, 2=Baseline
runtime / seconds
0.04
0.03
0.02
0.01
0
IH
Baseline
0
20
40
60
80
100
random game
runtime / seconds
0.2
0.15
0.1
0.05
0
IH
Baseline
0
20
40
60
80
100
random game
#games
80
60
40
20
0
IH
Baseline
0
0.01
0.02
0.03
0.04
runtime / seconds
#games
100
80
60
40
20
0
IH
Baseline
0
0.05
0.1
0.15
0.2
runtime / seconds

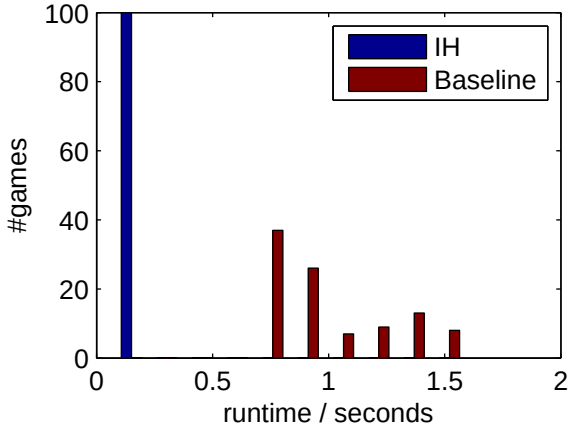

100 random games with
a=2, d=5, u=[−1;1]
runtime / seconds
0.4
0.3
0.2
0.1
1
2
1=IH, 2=Baseline

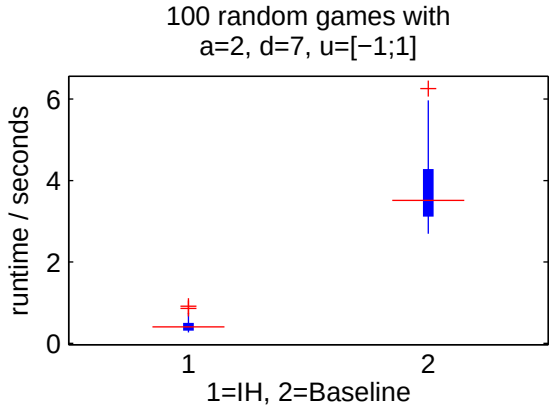

100 random games with
a=2, d=6, u=[−1;1]
runtime / seconds
1.5
1
0.5
0
1
2
1=IH, 2=Baseline

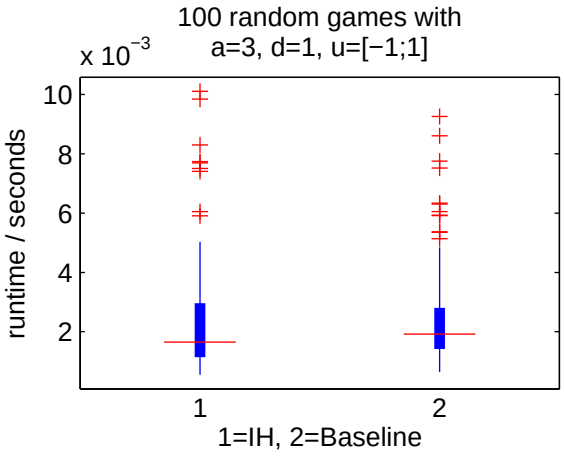

IH
Baseline
runtime / seconds
0.5
0.4
0.3
0.2
0.1
0
0
20
40
60
80
100
random game

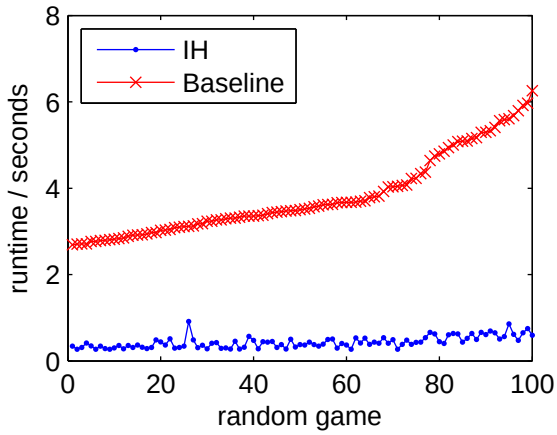

IH
Baseline
runtime / seconds
2
1.5
1
0.5
0
0
20
40
60
80
100
random game

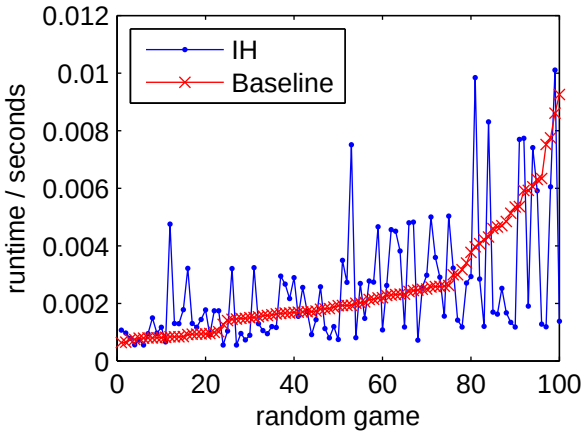

IH
Baseline
#games
100
80
60
40
20
0
0
0.1
0.2
0.3
0.4
0.5
runtime / seconds

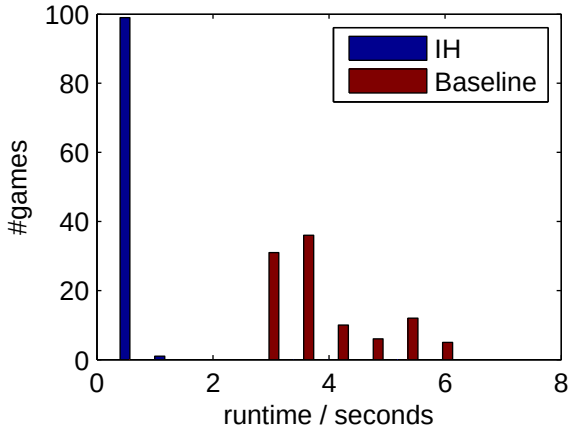

IH
Baseline
#games
100
80
60
40
20
0
0
0.5
1
1.5
2
runtime / seconds

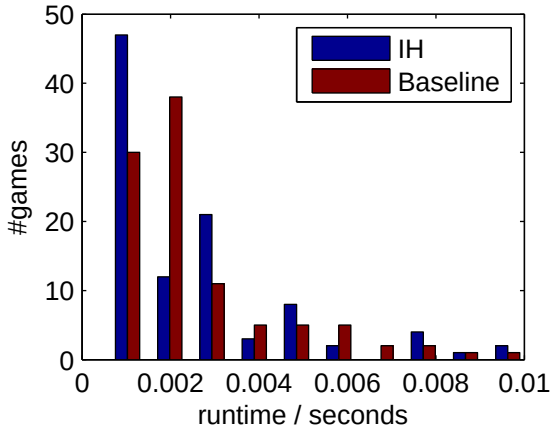

100 random games with
a=2, d=7, u=[−1;1]
runtime / seconds
0
2
4
6
1
2
1=IH, 2=Baseline

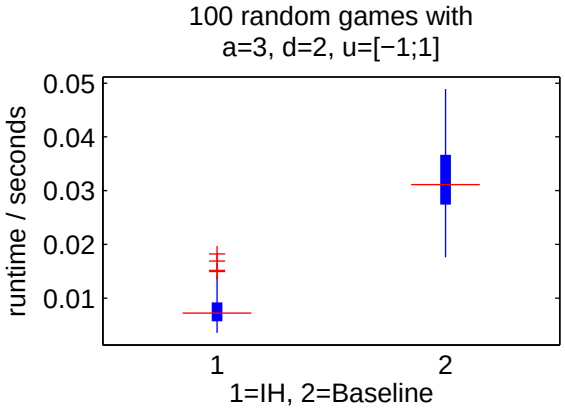

100 random games with
a=3, d=1, u=[−1;1]
x 10^−3
runtime / seconds
2
4
6
8
10
1
2
1=IH, 2=Baseline

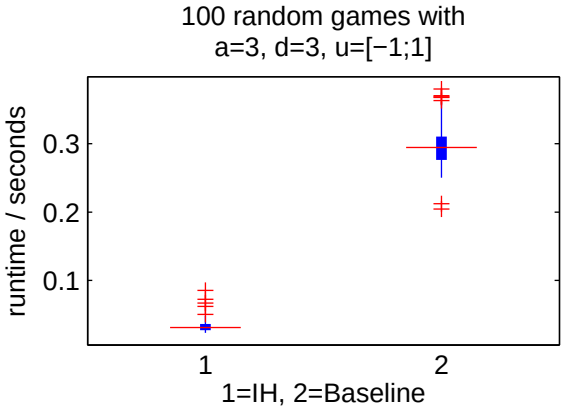

IH
Baseline
runtime / seconds
0
2
4
6
8
0
20
40
60
80
100
random game

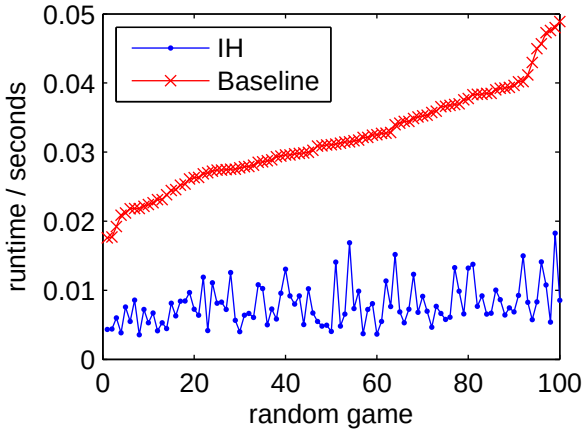

IH
Baseline
runtime / seconds
0
0.002
0.004
0.006
0.008
0.01
0.012
0
20
40
60
80
100
random game

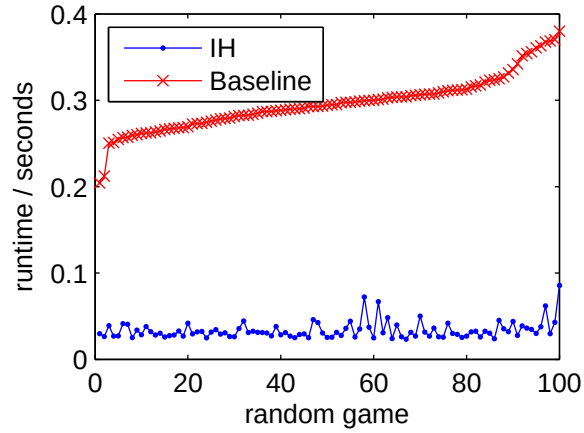

IH
Baseline
#games
0
20
40
60
80
100
0
2
4
6
8
runtime / seconds

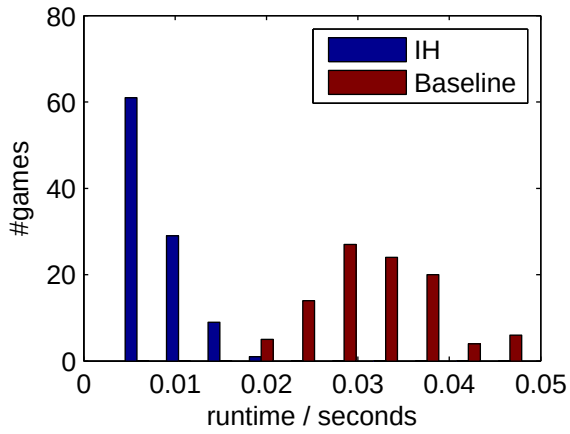

IH
Baseline
#games
0
10
20
30
40
50
0
0.002
0.004
0.006
0.008
0.01
runtime / seconds

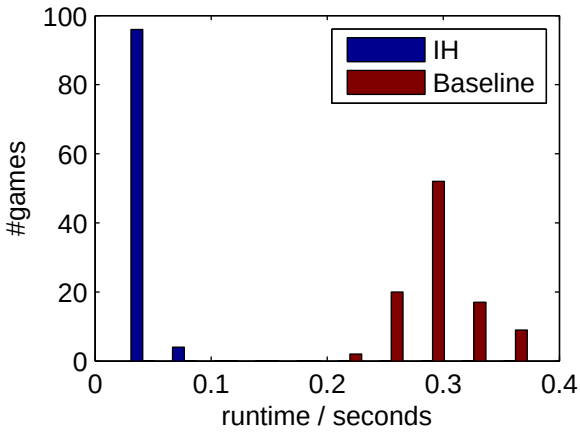

100 random games with
a=3, d=2, u=[−1;1]
runtime / seconds
0.05
0.04
0.03
0.02
0.01
1
2
1=IH, 2=Baseline

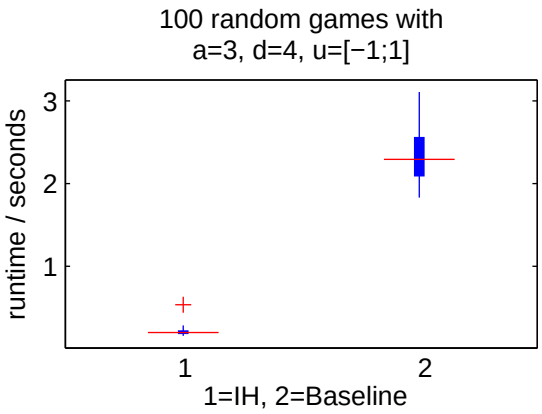

100 random games with
a=3, d=3, u=[−1;1]
runtime / seconds
0.3
0.2
0.1
1
2
1=IH, 2=Baseline

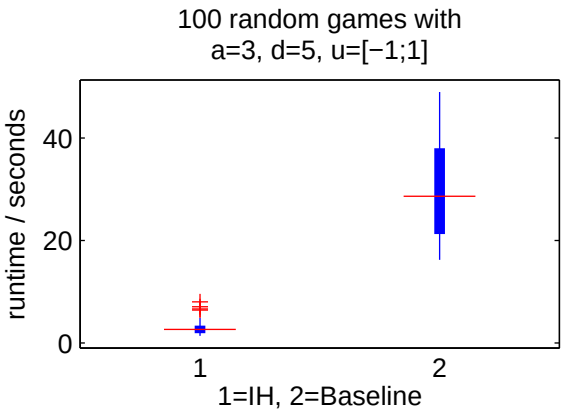

runtime / seconds
0.05
0.04
0.03
0.02
0.01
0
IH
Baseline
0
20
40
60
80
100
random game

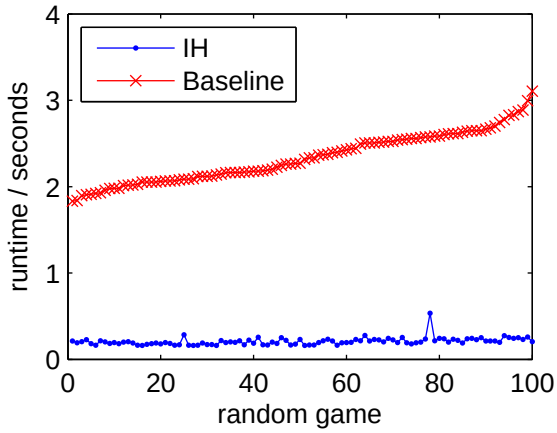

runtime / seconds
0.4
0.3
0.2
0.1
0
IH
Baseline
0
20
40
60
80
100
random game

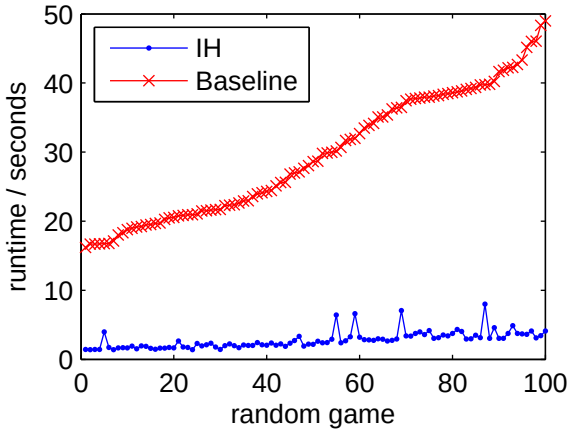

#games
80
60
40
20
0
IH
Baseline
0
0.01
0.02
0.03
0.04
0.05
runtime / seconds

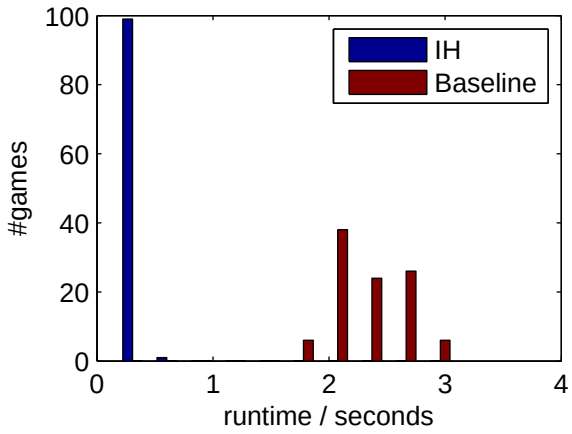

#games
100
80
60
40
20
0
IH
Baseline
0
0.1
0.2
0.3
0.4
runtime / seconds

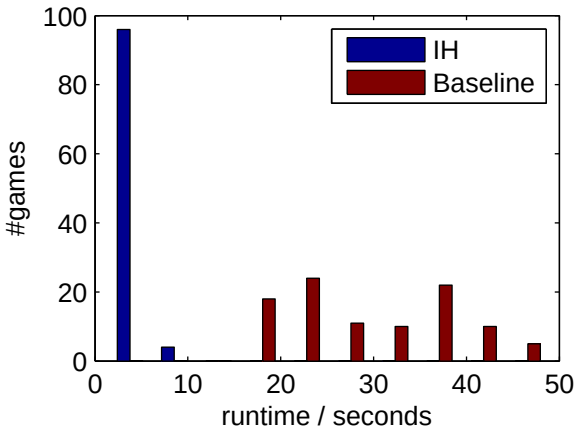

100 random games with
a=3, d=4, u=[−1;1]
runtime / seconds
1=IH, 2=Baseline


100 random games with
a=3, d=5, u=[−1;1]
runtime / seconds
1=IH, 2=Baseline


IH
Baseline
runtime / seconds
random game


IH
Baseline
runtime / seconds
random game


IH
Baseline
#games
runtime / seconds


IH
Baseline
#games
runtime / seconds

B. Simultaneous Tic-Tac-Toe GDL

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;  Simultaneous Tic-Tac-Toe
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Roles
  (role p0)
  (role p1)

; Initial State
  (init (cell 1 1 b))
  (init (cell 1 2 b))
  (init (cell 1 3 b))
  (init (cell 2 1 b))
  (init (cell 2 2 b))
  (init (cell 2 3 b))
  (init (cell 3 1 b))
  (init (cell 3 2 b))
  (init (cell 3 3 b))
  (init (step 1))

;Moves
  (<= (next (cell ?j ?k x))
      (true (cell ?j ?k b))
      (does p0 (mark ?j ?k))
      (does p1 (mark ?m ?n))
      (or (distinct ?j ?m) (distinct ?k ?n)))
  (<= (next (cell ?m ?n o))
      (true (cell ?m ?n b))
      (does p0 (mark ?j ?k))
      (does p1 (mark ?m ?n))
      (or (distinct ?j ?m) (distinct ?k ?n)))
  (<= (next (cell ?m ?n b))
      (true (cell ?m ?n b))
      (does p0 (mark ?m ?n))
      (does p1 (mark ?m ?n)))
  (<= (next (cell ?p ?q b))
      (true (cell ?p ?q b))
      (does p0 (mark ?j ?k))
      (does p1 (mark ?m ?n))
      (or (distinct ?j ?p) (distinct ?k ?q))
      (or (distinct ?m ?p) (distinct ?n ?q)))
  (<= (next (cell ?m ?n ?w))
      (true (cell ?m ?n ?w))
      (distinct ?w b))
  (<= (next (step ?y))
      (true (step ?x))
      (succ ?x ?y))

  (succ 1 2)
  (succ 2 3)
  (succ 3 4)
  (succ 4 5)
  (succ 5 6)
  (succ 6 7)

; Goal Concepts
  (<= (row ?m ?x)
```

```
      (true (cell ?m 1 ?x))
      (true (cell ?m 2 ?x))
      (true (cell ?m 3 ?x)))
  (<= (column ?n ?x)
      (true (cell 1 ?n ?x))
      (true (cell 2 ?n ?x))
      (true (cell 3 ?n ?x)))
  (<= (diagonal ?x)
      (true (cell 1 1 ?x))
      (true (cell 2 2 ?x))
      (true (cell 3 3 ?x)))
  (<= (diagonal ?x)
      (true (cell 1 3 ?x))
      (true (cell 2 2 ?x))
      (true (cell 3 1 ?x)))
  (<= (line ?x) (row ?m ?x))
  (<= (line ?x) (column ?m ?x))
  (<= (line ?x) (diagonal ?x))
  (<= nolinex
      (not (line x)))
  (<= nolineo
      (not (line o)))

; Legal Moves
  (<= (legal p0 (mark ?x ?y))
      (true (cell ?x ?y b)))
  (<= (legal p1 (mark ?x ?y))
      (true (cell ?x ?y b)))

; Goals and Payoffs
  (<= (goal p0 50)
      (line x)
      (line o))
  (<= (goal p0 100)
      (line x)
      nolineo)
  (<= (goal p0 0)
      nolinex
      (line o))
  (<= (goal p0 50)
      nolinex
      nolineo)
  (<= (goal p1 50)
      (line x)
      (line o))
  (<= (goal p1 100)
      nolinex
      (line o))
  (<= (goal p1 0)
      (line x)
      nolineo)
  (<= (goal p1 50)
      nolinex
      nolineo)

;Terminal States
  (<= terminal
      (true (step 7)))
  (<= terminal
      (line x))
  (<= terminal
      (line o))
```

Erklärung

Hiermit versichere ich, dass ich diese Diplomarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Stellen meiner Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen sind, habe ich in jedem Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht. Dasselbe gilt sinngemäß für Tabellen und Abbildungen. Diese Arbeit hat in dieser oder einer ähnlichen Form noch nicht im Rahmen einer anderen Prüfung vorgelegen.

Freiburg, April 2011

(Alexander Bisaliev)