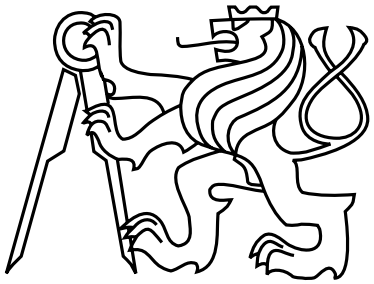
Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering



Master's Thesis

Using Refinements of Nash Equilibria for Solving Extensive-Form Games

Jiří Čermák

Supervisor: Mgr. Branislav Bošanský

Study Programme: Open informatics

Field of Study: Artificial Intelligence

May 12, 2014

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Computer Science and Engineering

DIPLOMA THESIS ASSIGNMENT

Student: **Bc. Jiří Čermák**

Study programme: Open Informatics
Specialisation: Artificial Intelligence

Title of Diploma Thesis: **Using Refinements of Nash Equilibria for Solving Extensive-Form Games**

Guidelines:

The extensive form is a mathematical model for representing finite sequential games. Designing domain-independent algorithms for solving large extensive-form games is a challenging task. Recently, an iterative double-oracle algorithm was introduced presenting an interesting alternative to the standard linear programming. One of the disadvantages of the new algorithm is that it calculates only a Nash equilibrium (NE), which is known to be a rather weak solution concept in extensive-form games. A number of refinements of NE have been designed over the years posing further restrictions on the optimal strategies. The first task of the student is to find the algorithms for computing these refinements and analyze the algorithms from the perspective of numerical stability. Then, the second task is to design and implement a new variant of the double-oracle algorithm that exploits some of these refined solution concepts and compare the performance to the original version of the algorithm.

Bibliography/Sources:

[1] Branislav Bosansky and Christopher Kiekintveld and Viliam Lisy and Jiri Cermak and Michal Pechoucek: Double-oracle Algorithm for Computing an Exact Nash Equilibrium in Zero-sum Extensive-form Games. In Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013. 2013.
[2] Eric Van Damme. Stability and perfection of Nash equilibria. Springer, 1991.
[3] Yoav Shoham, and Kevin Leyton-Brown. Multiagent systems: Algorithmic, game-theoretic, and logical foundations. Cambridge University Press, 2009.

Diploma Thesis Supervisor: Mgr. Branislav Bošanský

Valid until the end of the summer semester of academic year 2014/2015

L.S.

doc. Ing. Filip Železný, Ph.D.
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, March 3, 2014

Aknowledgements

Here I would like to thank my supervisor Mgr. Branislav Bošanský for his time and valuable assistance during the creation of this work. I would also like to thank my family and friends for their support.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 12. 5. 2014 .....

Abstract

There is a growing number of applications of game theory in the real world, especially in economical and security domains. There are two main challenges brought by these applications. First challenge is created by the need of robust solutions, caused by the non-rational nature of the human opponents encountered in these applications. Second challenge lies in the size of the games which need to be solved. This thesis addresses both of these issues. It provides first thorough experimental evaluation of existing advanced solution concepts on a set of real world inspired games. The best solution concept is then applied to the Double oracle algorithm, which is one of the most suitable algorithms for solving large domains found in these applications. The aim of this action is to even further increase the performance of this algorithm, by exploiting higher quality of solutions provided by the advanced solution concept.

Abstrakt

Existuje stále rostoucí množství aplikací teorie her ve scénářích ze skutečného světa, hlavně v bezpečnostních a ekonomický doménách. Tyto aplikace přináší dvě výzvy. První výzva je tvořena potřebou robustnějších řešení, způsobenou neracionální povahou lidských protivníků vyskytujících se v těchto aplikacích. Druhá výzva je tvořená velikostí her, které je potřeba řešit. Tato práce pomáhá řešit oba tyto problémy. Poskytuje první experimentální ohodnocení existujících pokročilých konceptů řešení na množině her, inspirovaných skutečným světem. Nejlepší z těchto řešení je pak aplikováno do Double oracle algoritmu, což je jeden z nejvhodnějších algoritmů pro řešení rozsáhlých domén, se kterými se setkáváme v těchto aplikacích. Cílem této akce je zvýšení jeho výkonu, pomocí využití kvalitnějších řešení poskytnutých tímto konceptem.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Game theory is a widely used approach for analysing multi-agent interaction using mathematical models, with an aim to find a behavior optimizing rewards obtained by players. In the recent years a growing number of real world applications of game theoretical approaches emerged, including placement of security checkpoints around airports [13], scheduling of Federal air marshals to commercial flights [19], development of poker players on the level of professional human players [15] or trading agents in auctions [24].

The main aim of game theory is to find the optimal behavior in various scenarios called games. Such a behavior is called strategy and is prescribed by a solution concept. The most famous and widely used solution concept is the Nash equilibrium [12], which is guaranteed to prescribe the optimal behavior to every player in the game, under the assumption of rational opponents. The main shortcoming of this solution concept is that it doesn't exploit mistakes made. So for example if in the game of poker one player by accident creates an opportunity for the second player to win 1000$ instead of the expected win of 1$ it is consistent with Nash equilibrium to ignore this opportunity and proceed to win the expected 1$ prize. This is caused by the fact that Nash equilibrium expects fully rational player and so it assumes that such mistake will never happen. This is a serious issue since such mistakes are common in the real world applications. This is caused by the fact that the opponents in these applications are usually humans, which are known to behave irrationally in various scenarios.

There is a number of solution concepts called refinements of the Nash equilibrium which attempt to cope with these issues. These concepts still guarantee the optimality against rational opponents and further improve the Nash equilibrium, exploiting various types of mistakes of the opponents or even by the player himself. And so when one encounters an unexpected situation, where there is a profit higher then expected achievable, these solution concepts should prescribe behavior maximizing the profit. Unfortunately, the only comparison of existing refinements available, is performed on small, artificially created domains, specifically tailored to show some desirable property. In the first part of this thesis we will fill this gap for a two player games with sequential interaction in fully competitive environment, with imperfect information caused by unobservable actions of opponents or stochastic environment. These games are called two-player zero-sum extensive-form games with imperfect information. We provide experimental evaluation of chosen refinements on real world inspired two-player zero-sum extensive-form games, such as card games or border patrolling

scenarios. Their overall performance is discussed along with implementational complications, such as numerical stability, which need to be overcome in order to compute them.

Furthermore thanks to the demand of scalability introduced by often large applications, several algorithms specifically created to solve large extensive-form games emerged. These algorithms allow further deployment of the game theoretical approaches to various, previously too complex, domains. For example, there is the Counterfactual regret minimization algorithm [25] which was successful in solving poker games orders of magnitude larger than the current state of the art. Another of such algorithms is the Double oracle algorithm [1] which solves large games by iteratively building a smaller game, computing its Nash equilibrium. This procedure is repeated until the solution of this smaller game equals to the solution of the complete game. This approach offers a possibility to solve games faster, while also saving the memory needed, since there is no need to construct the whole representation of the original game.

The main topic of this thesis is the incorporation of the best suitable refinement to the Double oracle algorithm to further improve its performance, by computing more sensible strategies in every iteration.

1.1 Thesis outline

Chapter 2 provides a brief introduction of the most used representations of games. Chapter 3 defines the Nash equilibrium and the most known refinements. Chapter 4 contains a discussion about the computational aspects of previously defined refinements. Chapter 5 is based on Čermák et al. [26] and presents results of comparison of refinements of Nash equilibrium. Chapter 6 formally introduces the Double oracle algorithm. Chapter 7 presents results of Double oracle algorithm using refined solver. Chapter 8 contains a conclusion of the thesis.

Chapter 2

Introduction to Game Theory

This chapter presents the most known representations of the games. We provide the description and formal definitions of normal-form and extensive-form games. The main focus of this thesis is on two-player zero-sum extensive-form games with imperfect information and perfect recall, however number of concepts discussed in following chapters make use of the normal-form representation and so this representation was not omitted. Furthermore every extensive-form game can be converted to the normal-form representation, and so this representation should be thoroughly analyzed. Even though it may appear that the limitations used are too restrictive, they still permit a large number of purely competitive scenarios, where the gain of one player equals to the loss of the other. In addition, they allow the possibility of unobservable moves of the opponent or the stochastic environment, dramatically increasing the number of domains consistent with such representation. The restriction to perfect recall means that every player remembers what he did in the past along with all the informations he obtained during the play. Between games consistent with these restriction belong classical games such as chess, two player poker or more realistic scenarios such as security of industrial objects.

2.1 Normal-form Games

Normal-form games typically represent one-shot simultaneous moves games, using a matrix, assigning utility value for every combination of actions of players. This representation doesn't exploit any structure, such as sequential interaction between players, of the game played. Instead it simply evaluates every pair of actions, under assumption that both players play simultaneously. These facts make normal-form unsuitable for representation of larger games.

Definition 1. *Two player normal-form game is a tuple (P, A, u) where:*

1. *P is a set of 2 players indexed by i*
2. $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$ *where* $\mathcal{A}_i$ *is a finite set of actions available to player i*
3. $u = (u_1, u_2)$ *where* $u_i : \mathcal{A} \to \mathbb{R}$ *is a utility function for player i*

	C'	D'
C	-1, -1	-4, 0
D	0, -4	-3, -3

Table 2.1: Prisoner's Dilemma as a normal-form game

As an example of the normal-form game consider the game called Prisoner's Dilemma depicted in Table 2.1. Prisoner's Dilemma describes a scenario where two prisoners sit under interrogation in different rooms. Both of them have the possibility to cooperate (with the other prisoner) C or to defect D. Both players act simultaneously, which means that neither of them knows what the other did when deciding. The utility matrix in Table 2.1 describes the outcomes for every combination of all available actions.

Let us now discuss strategies of players. A strategy can be seen as a plan prescribing behavior to a player. A set of pure strategies $\mathcal{S}_i$ corresponds to the set of actions $\mathcal{A}_i$. A set of mixed strategies Δ_i contains all the probability distributions δ_i over the set $\mathcal{S}_i$. A strategy profile is a vector of strategies, one for each player.

2.2 Extensive-form Games

Extensive-form games provide a representation which is more suitable for description of scenarios which evolve in time. This representation allows every player to choose action in each state of the game, using exponentially smaller representations in the form of a game tree instead of the utility matrix. Nodes of this tree represent the states of the game and edges actions available in the corresponding states. Leafs represent the terminal states and have the utility value for all players associated with them, representing their preference of this outcome. We distinguish two types of extensive-form games, perfect-information and imperfect information ones. In perfect-information games every player knows everything about the state of the game, whilst in imperfect-information games some of these informations may be hidden.

Definition 2. *A finite extensive-form game with perfect information has the following components:*

1. *A finite set $\mathcal{P}$ of players*

2. *A finite set $\mathcal{H}$ of sequences, the possible histories of actions, such that the empty sequence is in $\mathcal{H}$ and every prefix of a sequence in $\mathcal{H}$ is also in $\mathcal{H}$. $\mathcal{Z} \subseteq \mathcal{H}$ are the terminal histories (those which are not a prefix of any other sequences). $A(h) = \{a : (h, a) \in H\}$ are the actions available after a nonterminal history $h \in \mathcal{H}$.*

3. *A function $P(h)$ that assigns to each nonterminal history (each member of $\mathcal{H} \backslash \mathcal{Z}$) a member of $\mathcal{P} \cup \{c\}$. $P(h)$ is the player who takes an action after the history h. If $P(h) = c$ then chance determines the action taken after history h.*

4. *A function f_c that associates with every history h for which $P(h) = c$ a probability measure $f_c(.|h)$ on $A(h)$ ($f_c(a|h)$ is the probability that a occurs given h), where each such probability measure is independent of every other such measure.*

5. *For each player $i \in \mathcal{P}$ a utility function u_i from terminal states $\mathcal{Z}$ to $\mathbb{R}$. If $\mathcal{P} = \{1, 2\}$ and $u_2 = -u_1$ it is a zero sum extensive-form game.*

Definition 3. *A finite extensive-form game with imperfect information is a tuple $(\mathcal{P}, \mathcal{H}, P, f_c, \mathcal{I}, u)$, with following components:*

1. *Perfect-information extensive-form game $(\mathcal{P}, \mathcal{H}, P, f_c, u)$*

2. *For each player $i \in \mathcal{P}$ a partition $\mathcal{I}_i$ of $\{h \in \mathcal{H} : P(h) = i\}$ with the property that $A(h) = A(h')$ whenever h and h' are in the same member of the partition. For $I_i \in \mathcal{I}_i$ we denote by $A(I_i)$ the set $A(h)$ and by $P(I_i)$ the player $P(h)$ for any $h \in I_i$. $\mathcal{I}_i$ is the information partition of player i; a set $I_i \in \mathcal{I}_i$ is an information set of player i*

Informally, every information set I_i for player i contains all game states h which are indistinguishable for i.

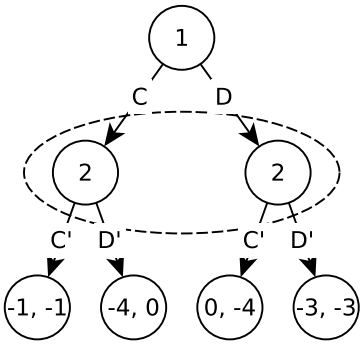


Figure 2.1: Prisoner's Dilemma as an extensive-form game

As an example consider the imperfect-information extensive-form game from Figure 2.1. It is again the Prisoner's Dilemma, this time represented as an extensive-form game. Player 1 plays first and makes a choice in the root of the game tree. As we can see the states after both choices are grouped into 1 information set, because they are indistinguishable for player 2. The leaf of the tree and therefore terminal state of the game is reached after the action of player 2 and the outcome of the game is evaluated.

Next we provide the formal definition of perfect recall, which is one of the properties required from the games used in this thesis.

Definition 4. *Player i has a perfect recall in an imperfect-information game G if for any two nodes h, h', that are in the same information set I_i, for any path $h_0, a_0, h_1, a_1, h_2, ..., h_m, a_m, h$ from the root of the game to h (where the h_j are decision nodes and the a_j are actions) and for any path $h_0, a'_0, h'_1, a'_1, h'_2, ..., h'_{m'}, a'_{m'}$, h' from the root to h' it must be the case that:*

1. $m = m'$
2. *for all* $0 \leq j \leq m$, *if* $P(h_j) = i$ *(i.e.,* h_j *is a decision node of player* i*), then* h_j *and* h'_j *are in the same equivalence class for* i*;*
3. *for all* $0 \leq j \leq m$, *if* $P(h_j) = i$ *(i.e.,* h_j *is a decision node of player* i*), then* $a_j = a'_j$*.*

Let us now discuss strategies in extensive-form games. A pure strategy for player i is a mapping $\mathcal{I}_i \to A(I_i)$. $\mathcal{S}_i$ is a set of all pure strategies for player i. A mixed strategy δ_i is again a probability distribution over elements of $\mathcal{S}_i$, with Δ_i representing the set of all mixed strategies for i. In the extensive form games, we can represent strategies as behavioral strategies b_i, which assign probability distribution over $A(I_i), \forall I_i \in \mathcal{I}_i$. For all games with perfect recall, behavioral strategies have the same expressive power as mixed strategies [6].

Finally in the games of perfect recall, we can use sequence-form representation [4]. A sequence σ_i is a list of actions of player i ordered by their occurrence on the path from the root of the game tree to some node. These sequences are used to represent the strategy as a realization plan r_i. The realization plan r_i assigns the probability to sequences σ_i of player i, assuming other players play such that the actions of σ_i can by executed. Furthermore r_i satisfies the network flow property, i.e. $r_i(\sigma) = \sum_{a \in A(I)} r_i(\sigma \cdot a)$, where I is an information set reached by sequence σ and $\sigma \cdot a$ stands for σ extended by action a.

Thanks to the equality of strategy representations above, we overload the notation of utility function to $u(s_i, s_{-i})$ as the utility of the state reached when playing according to s_i and s_{-i}, $u(\delta_i, \delta_{-i})$ as an expected utility, when playing according to δ_i and δ_{-i} and similarly $u(b_i, b_{-i})$ and $u(r_i, r_{-i})$. Furthermore let us denote $U_i(s_i|\Delta)$ as the utility obtained when i plays s_i and the rest of the players follows strategy profile Δ. $U_i(I, a|B)$ is the utility obtained by i in information set I, when playing action a in I and according to strategy profile B otherwise.

This chapter provided description of the types of games and restrictions we will assume through this thesis. The next chapter will use this background to describe various solution concepts used for solving normal-form and extensive-form games.

Chapter 3

Nash Equilibrium and Refinements

This chapter introduces optimal strategies for games described by solution concepts. The most famous solution concept of Nash equilibrium is introduced and all its shortcomings discussed. Next we describer the most known refinements of the Nash equilibrium for both normal-form and extensive-form games.

3.1 Nash equilibrium

Nash equilibrium is a solution concept due to Nash [12], which prescribes the optimal behavior under the assumption of opponents playing to maximize their outcome. Let us first define the notion of the best response.

Definition 5. *Strategy δ_i^* is the best response to strategy δ_{-i} iff $u(\delta_i^*, \delta_{-i}) \geq u(\delta_i, \delta_{-i}), \forall \delta_i \in \Delta_i$. We denote $br(\delta_i)$ as the set of all best responses to δ_i.*

Informally the Nash equilibrium is such a strategy profile where no player wants to deviate from its strategy, even when he learns the strategy of others. Note that there might be an infinite number of such strategy profiles. A player doesn't want to deviate from his strategy only when he plays according to the best response to strategies of all other players, and so we arrive to the formal definition of Nash equilibrium.

Definition 6. *Strategy profile Δ is a Nash equilibrium iff $\forall \delta_i \in \Delta : \delta_i \in br(\delta_{-i})$.*

As an example lets consider the game from Table 3.1. There are two Nash equilibria in the example, namely (α_1, α_2) and (β_1, β_2). Since $br(\alpha_1) = \{\alpha_2\}$ and $br(\alpha_2) = \{\alpha_1\}$, (α_1, α_2) is indeed Nash equilibrium. Same goes for (β_1, β_2), since $br(\beta_1) = \{\alpha_2, \beta_2\}$ and $br(\beta_2) = \{\alpha_1, \beta_1\}$.

3.2 Refinements in Normal-form Games

The need to refine Nash equilibrium strategies in normal-form games follows from the fact that the Nash equilibrium doesn't exploit possible mistakes of the opponent. Let us consider the game in Table 3.1. As shown above, this game has two Nash equilibria (α_1, α_2) and

(β_1, β_2) since in both cases there is no gain for either of the players obtained by deviating from given strategy profile. However, thanks to the structure of this particular game, neither of them can actually lose anything by changing strategy from β_i to α_i. There is only one equilibrium optimal when considering possible mistakes of the opponent (α_1, α_2). Solution concepts introduced in this section will attempt to preserve only the rational equilibria.

	α_2	β_2
α_1	1, 1	0, 0
β_1	0, 0	0, 0

Table 3.1: Normal-form game with a non-robust equilibrium

3.2.1 Undominated equilibrium

The main idea behind this equilibrium is, that if choosing between two actions, where one of them is at least as good as the other no matter what the opponent does and better for at least one action of the opponent, one should always prefer this action over the other. This relation between actions is called weak dominance. Formally strategy s_i^1 weakly dominates s_i^2 iff $\forall s_{-i} \in \mathcal{S}_{-i} : u(s_i^1, s_{-i}) \geq u(s_i^2, s_{-i})$ and $\exists s_{-i} \in \mathcal{S}_{-i} : u(s_i^1, s_{-i}) > u(s_i^2, s_{-i})$. And so the undominated equilibrium in normal-form games is such a Nash equilibrium which consists only of those strategies, which are not weakly dominated by any other strategy.

The only undominated equilibrium of the game from Figure 3.1 is (α_1, α_2), because α_i dominates β_i, $\forall i \in \mathcal{P}$. In the game from Figure 3.2, which was created from the game in Table 3.1 by adding actions γ_1 and γ_2, are two undominated equilibria, namely (α_1, α_2) and (β_1, β_2), because the only dominated actions of this game are γ_1 and γ_2, and so the only Nash equilibrium which is not undominated is (γ_1, γ_2).

3.2.2 Perfect equilibrium

Solution concept due to Selten et al. [16]. The basic idea behind this concept is, that it is possible for every player to make mistakes with a small probability. And so every player needs to consider all the outcomes of his actions, not only the ones expected when considering rational opponent. Let us first define ε-perfect equilibrium.

Definition 7. *An ε-perfect equilibrium is a fully mixed strategy profile $\Delta = (\delta_1, ..., \delta_n)$ such that*

$$U_j(s_j|\Delta) < U_j(s'_j|\Delta) \Rightarrow \delta_j(s_j) \leq \varepsilon, \quad \forall j \in \mathcal{P}, \forall s_j, s'_j \in \mathcal{S}_j \tag{3.1}$$

A perfect equilibrium is then defined as the limit of ε-perfect equilibria. That is $(\delta_1, ..., \delta_n)$ is a perfect equilibrium iff there exists sequence $\{\varepsilon\}_{k=1}^{\infty}$ and $(\delta_1^k, ..., \delta_n^k)_{k=1}^{\infty}$ such that each $\varepsilon_k > 0$ and $\lim_{k\to\infty} \varepsilon_k = 0$, each $(\delta_1^k, ..., \delta_n^k)$ is an ε_k-perfect equilibrium and $\lim_{k\to\infty} \delta_i^k(s_i) = \delta_i(s_i), \forall i \in P, \forall s_i \in S_i$.

For any game in normal form, the perfect equilibria form a non-empty subset of Nash equilibria [16]. Furthermore every perfect equilibrium is undominated and for two-player zero-sum games every undominated equilibrium is perfect [22].

Let us now examine perfect equilibria of our motivational game from Table 3.1. Consider following strategy profile.

$$\delta_1^\varepsilon(\alpha_1) = \varepsilon, \quad \delta_1^\varepsilon(\beta_1) = 1 - \varepsilon \tag{3.2}$$
$$\delta_2^\varepsilon(\alpha_2) = \varepsilon, \quad \delta_2^\varepsilon(\beta_2) = 1 - \varepsilon \tag{3.3}$$

Responses for this profile and player 1 are then evaluated.

$$U_1(\alpha_1|\delta_1^\varepsilon, \delta_2^\varepsilon) = \varepsilon \tag{3.4}$$
$$U_1(\beta_1|\delta_1^\varepsilon, \delta_2^\varepsilon) = 0 \tag{3.5}$$

As we can see, the best response here is α_1 and so this profile is not ε-perfect equilibrium. From the fact that there is only one Nash equilibrium left and the set of perfect equilibria is always non-empty follows that (α_1, α_2) is perfect equilibrium.

	α_2	β_2	γ_2
α_1	1, 1	0, 0	-9, -9
β_1	0, 0	0, 0	-7, -7
γ_1	-9, -9	-7, -7	-7, -7

Table 3.2: Normal-form game with two perfect equilibria

Now let us examine the game represented in normal form in Table 3.2, which was created from the game in Table 3.1 by adding actions γ_1 and γ_2. The outcome we would like to achieve once again is (α_1, α_2), since both added actions are dominated. And indeed this strategy profile is still Nash equilibrium. There are however two additional pure strategy Nash equilibria, namely (β_1, β_2) and (γ_1, γ_2). Of these only (γ_1, γ_2) is not perfect. Let us now show that (β_1, β_2) is perfect equilibrium. We choose following strategy profile.

$$\delta_1^\varepsilon(\alpha_1) = \varepsilon, \quad \delta_1^\varepsilon(\beta_1) = 1 - 2\varepsilon, \quad \delta_1^\varepsilon(\gamma_1) = \varepsilon \tag{3.6}$$
$$\delta_2^\varepsilon(\alpha_2) = \varepsilon, \quad \delta_2^\varepsilon(\beta_2) = 1 - 2\varepsilon, \quad \delta_2^\varepsilon(\gamma_2) = \varepsilon \tag{3.7}$$

Responses for this profile and player 1 are then evaluated as follows.

$$U_1(\alpha_1|\delta_1^\varepsilon, \delta_2^\varepsilon) = -8\varepsilon \tag{3.8}$$
$$U_1(\beta_1|\delta_1^\varepsilon, \delta_2^\varepsilon) = -7\varepsilon \tag{3.9}$$
$$U_1(\gamma_1|\delta_1^\varepsilon, \delta_2^\varepsilon) = -7 - 2\varepsilon \tag{3.10}$$

So β_1 is the best response to δ_2^ε and as required in (3.1) $\delta_1^\varepsilon(\alpha_1) \leq \varepsilon, \delta_1^\varepsilon(\gamma_1) \leq \varepsilon$. Then as ε converges to zero $(\delta_1^\varepsilon, \delta_2^\varepsilon)$ converge to the strategies which select β_1 and β_2 with probability 1, implying that (β_1, β_2) is indeed perfect equilibrium. This property is caused by adding dominated strategies (γ_1, γ_2) to the game and was pointed out by Myerson et al. [11].

α_3

	α_2	β_2
α_1	1, 1, 1	0, 0, 1
β_1	0, 0, 1	0, 0, 1

β_3

	α_2	β_2
α_1	0, 0, 0	0, 0, 0
β_1	0, 0, 0	1, 1, 0

Table 3.3: Proper equilibrium influenced by adding strictly dominated strategy

3.2.3 Proper equilibrium

Solution concept due to Myerson et al.[11], which is robust against small perturbations in strategies. These perturbations have some additional properties added with an aim to resolve the issues mentioned in previous section. They are considered to be rational, meaning that the costly mistake is expected with an order of magnitude smaller probability than the cheap one. Let us first define ε-proper equilibria.

Definition 8. *ε-proper equilibrium is a fully mixed strategy profile $\Delta = (\delta_1, ..., \delta_n)$ such that*

$$U_j(s_j|\Delta) < U_j(s'_j|\Delta) \Rightarrow \delta_j(s_j) \leq \varepsilon \cdot \delta_j(s'_j), \quad \forall j \in \mathcal{P}, \forall s_j, s'_j \in \mathcal{S}_j \tag{3.11}$$

This definition directly implies that every ε-proper equilibrium is ε-perfect since $\delta_j(s_j) \leq \varepsilon \cdot \delta_j(s'_j) \Rightarrow \delta_j(s_j) \leq \varepsilon, \quad \forall j \in \mathcal{P}, \forall s_j, s'_j \in \mathcal{S}_j$ but the opposite doesn't hold. Strategy profile $(\delta_1, ..., \delta_n)$ is a proper equilibrium iff there exist some sequences $\{\varepsilon\}_{k=1}^{\infty}$ and $(\delta_1^k, ..., \delta_n^k)_{k=1}^{\infty}$ such that each $\varepsilon_k > 0$ and $\lim_{k\to\infty} \varepsilon_k = 0$, each $(\delta_1^k, ..., \delta_n^k)$ is ε_k-proper equilibrium and $\lim_{k\to\infty} \delta_i^k(s_i) = \delta_i(s_i), \forall i \in P, \forall s_i \in S_i$. For any game in normal form, the proper equilibria form a non-empty subset of perfect equilibria [11]

Now let us return to the game from Table 3.2. Let us check that the strategy profile from (3.6) and (3.7) is not ε-proper equilibrium. Since for $0 < \varepsilon < 1$ holds that $U_1(\gamma_1|\delta_1^\varepsilon, \delta_2^\varepsilon) < U_1(\alpha_1|\delta_1^\varepsilon, \delta_2^\varepsilon)$ but from $\delta_1(\gamma_1) > \varepsilon \cdot \delta_1(\alpha_1)$ follows that (β_1, β_2) is not ε-proper equilibrium. As we can see insisting on properness removed the undesirable property of perfect equilibrium, by expecting more costly mistakes to have much smaller probability of occurrence than the less costly ones.

However to show that this issue is not fully resolved by the proper equilibrium let us discuss a 3 player game shown in Table 3.3 taken from [22]. If we limit third players action only to α_3 (left table) then $(\alpha_1, \alpha_2, \alpha_3)$ is an unique proper equilibrium of this game. However by adding second action β_3 for third player, which is strictly dominated, we create another proper equilibrium $(\beta_1, \beta_2, \alpha_3)$, and so we see that even though proper equilibrium fixes this issue in some games, there are still examples where the undesirable equilibria get picked.

3.3 Refinements in Extensive-form Games

The main shortcoming of Nash equilibrium in extensive-form games is, that it guarantees rational behavior only when the rest of the players play rationally, with no concern for gains possibly caused by mistakes of said opponents. We distinguish two types of mistakes against which one would like to be optimal. The first type is called mistakes in the past. When this

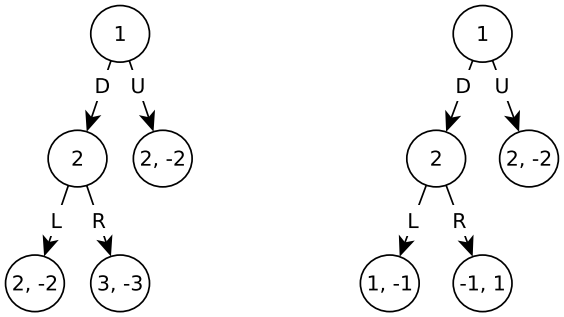


Figure 3.1: (a) A game where player 1 needs to consider mistake in the future of player 2. (b) A game where player 2 needs to consider mistake in the past.

mistake occurs a player finds himself in the part of the tree, which he didn't expect to visit, when considering rational opponent. As an example consider the game from Figure 3.1(b). In this game, there is no motivation for player 2 to prefer R over L, since he expects player 1 to always choose U immediately ending the game. The second type of mistakes is called mistakes in the future. To be optimal against this type of mistakes, player i should push his opponent to situations, where he needs to choose between for him bad and good actions. The opponent then finds himself in situations where the potential mistakes are as costly as possible. As an example consider the game from Figure 3.1(a). Here player 1 should always prefer D to U since he can only gain by playing D if player 2 makes a mistake by playing L. Following solution concepts will attempt to exploit these types of mistakes.

3.3.1 Subgame perfect equilibrium

Solution concept due to Kuhn [6]. Strategy profile B of game G is a subgame perfect equilibrium, if for every subgame G' of G holds that B prescribes behavior consistent with Nash equilibrium of G'. A subgame is a subset of the original game with following properties. (1) the root of the subgame is not in the information set with any other game state of the original game. (2) if a game state belongs to the subgame, all its successors must also belong to the subgame. (3) if a game state belongs to the subgame, all the nodes contained in the same information set must be also included in the subgame. Set of the subgame perfect equilibria forms a non-empty subset of Nash equilibria [6].

If we try this approach to our motivational game in Figure 3.1(b), we indeed see that the only subgame perfect equilibrium is (U, R), since playing L would violate equilibrium in subgame after action D of player 1. This is not a coincidence, since when the subgames are well defined subgame perfect equilibium generates strategies optimal against the mistakes in the past. In the game from Figure 3.1(a) there are two subgame perfect equilibria (U, L) and (D, R) because optimality in subgames cannot exploit mistakes in the future. There are further issues with subgame perfection. For example let us consider the game in Figure 3.2. As we can easily see, this game has no subgames (except for the game itself), since the initial action of chance is unobservable for both players and so subgame perfect equilibrium doesn't add any more restrictions to Nash equilibrium.

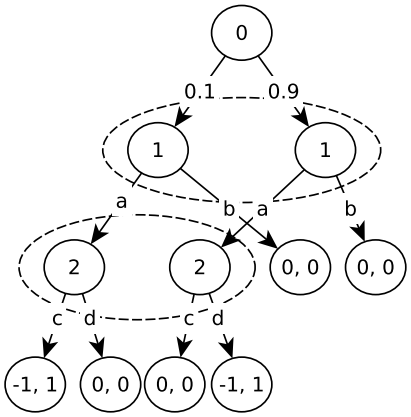


Figure 3.2: A game with no subgame

3.3.2 Undominated equilibrium

An undominated equilibrium in extensive-form games is a Nash equilibrium which consists only of those strategies, which are not weakly dominated by any other strategy. Note that the definition is equal to the definition of undominated equilibrium in normal-form games and so an undominated equilibrium of an extensive-form game is undominated equilibrium of its corresponding normal-form game and vice versa. For the two player zero-sum games holds, that every undominated equilibrium is a perfect equilibrium of the corresponding normal-form game [22].

The only undominated equilibrium of the game from Figure 3.1(a) is (D, L), because D weakly dominates U. The games from Figure 3.1(b) and 3.2 are also solved correctly with the unique undominated equilibria (U, R) and (b, d). Unfortunately, there are issues with undominated equilibrium in the game from Figure 3.3. This game is called Matching pennies on Christmas morning, which is a modification of the zero-sum game Matching pennies, where player 2 hides a penny and player 1 proceeds to guess whether it is heads up or tails up. If he guesses correctly he receives 1, 0 otherwise. When played on Christmas morning player 2 has additional possibility to give player 1 a gift of 1, no strings attached. The optimal strategy we would expect to see is for player 2 to not give player 1 the free gift and uniform distribution over actions in the rest of information sets. The undominated equilibrium however doesn't add any restriction to Nash equilibrium solutions, because neither of the strategies h' and t' dominates the other. And so all strategies where $\delta_2(TN') = \delta_2(HN) = \delta_1(t) = \delta_1(h) = 0.5$, $\delta_2(TG') = \delta_2(HG) = 0$ are consistent with both Nash and undominated equilibrium.

3.3.3 Sequential equilibrium

Solution concept due to Kreps and Wilson [5], which can be seen as a generalization of subgame perfect equilibrium to games with imperfect information, resolving issues with badly defined subgames by introducing notion of beliefs over undistinguishable states in

information sets. Insisting on sequentiality rules out equilibria not optimal against the mistakes of the opponent in the past. Let us first define two notions.

Definition 9. *A system of beliefs is a mapping* μ

$$\mu : S(I) \to [0,1], \quad \forall I \in \mathcal{I} \tag{3.12}$$

where $S(I)$ *is set of all states contained in information set* I*. It must hold that*

$$\forall I : \sum_{s \in S(I)} \mu(s) = 1 \tag{3.13}$$

Less formally, the system of beliefs is player's assumption about the real state of the game, given an information set.

Definition 10. *An assessment is a pair* (μ, B)*, where* μ *is system of beliefs and* B *is a behavioral strategy profile.*

Definition 11. *An assessment* (μ, B) *is consistent if there exists sequence* $\{\mu^\varepsilon, B^\varepsilon\}_{\varepsilon \downarrow 0}$ *where* B^ε *is completely mixed behavior strategy profile and* μ^ε *is a system of beliefs generated by* B^ε*, such that*

$$\lim_{\varepsilon \to 0} (\mu^\varepsilon, B^\varepsilon) = (\mu, B) \tag{3.14}$$

Sequential equilibrium is an assessment (μ, B) which is consistent and for which B is sequential best response against (μ, B). Every sequential equilibrium is subgame perfect and is guaranteed to exist [5]. One of the positive properties of sequential equilibrium is, that it exploits mistakes made by opponent in the past. It doesn't however exploit possible mistakes in the future. So for the game from Figure 3.1(a) there are again two sequential equilibria (U, L) and (D, L) because the introduction of beliefs doesn't in any way help with the mistakes in the future. The game from Figure 3.1(b) has only one sequential equilibrium because it further refines subgame perfection. Furthermore thanks to the beliefs the only sequential equilibrium of the game from Figure 3.2 is the only rational one (b, d).

3.3.4 Quasi-perfect equilibrium

Informally solution concept due to van Damme [21] that requires each player at every information set to take a choice optimal against mistakes of the opponent.

Definition 12. *Fix an information set* I_i*. Let* i *be player which makes decision in* I_i*. An* I_i*-local purification of behavioral strategy* b_i *is a behavioral strategy for* i *created by replacing the behavior of* i *in* I_i *and every information set of* i *encountered after* I_i*, by behavior that puts all probability mass on single action in these sets.*

As an example consider the behvioral strategy in the game from Figure 3.3 $b_2(T) = 1$, $b_2(N') = b_2(G') = 0.5$. There are two local purifications of the information set reached by the action T, namely $b_2(T) = b_2(N') = 1$ and $b_2(T) = b_2(G') = 1$.

Definition 13. *We say that an* I_i*-local purification is an* I_i*-local best response to* b_{-i} *if it achieves the best expected payoff among all* I_i*-local purifications.*

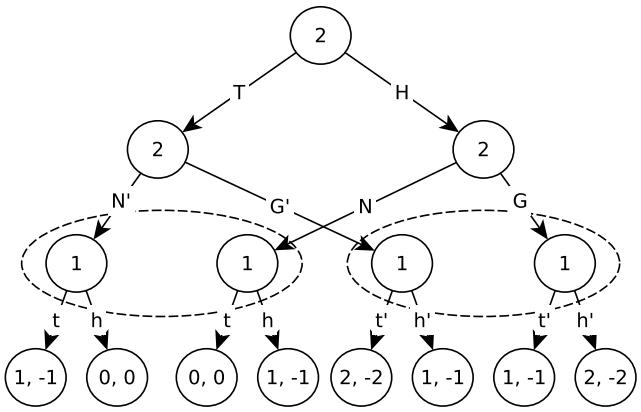


Figure 3.3: Matching Pennies on Christmas Morning [10]

Definition 14. *I_i-local purification b'_i is ε-consistent with b_i, if b_i assigns behavioral probability strictly bigger than ε to the actions to which b'_i assigns 1 in I_i and following information sets of player i.*

Strategy profile B is ε-quasi perfect if it is fully mixed and if for each player i and every information set I_i belonging to i, all I_i-local purifications of b_i that are ε-consistent with b_i are I_i-local best responses to b_{-i}. Strategy profile is quasi-perfect equilibrium if it is the limit point when ε goes to 0 of ε-quasi perfect strategy profiles. Every quasi-perfect equilibrium is sequential and every game possesses at least one quasi-perfect equilibrium [21]. Quasi-perfect equilibrium of extensive-form game is a perfect equilibrium of corresponding normal-form game [21].

Thanks to the fact that quasi-perfect equilibrium is the intersection of sequential and normal-form perfect equilibrium (undominated for two-player zero-sum games) all the games from figures 3.1(a), 3.1(b) and 3.2 are solved correctly. There is however still the issue with Matching pennies on Christmas morning from Figure 3.3, since neither sequentiality nor undominated equilibrium constraints $\delta(t')$ and $\delta(h')$.

3.3.5 Normal-form proper equilibrium

An equilibrium in behavioral strategies of an extensive-form game is said to be normal-form proper [10] if it is behaviorally equivalent to a proper equilibrium of the corresponding normal-form game. This equilibrium is optimal against mistakes of opponent in the past and in the future. Furthermore, the solution concept assumes that these mistakes are made in a rational manner, meaning that the more costly mistakes are made with exponentially smaller probability than the less costly ones. Finally, as shown in [10], every normal-form proper equilibrium is quasi-perfect and the set of normal-form proper equilibria of every extensive-form game is non-empty.

Since every normal-form proper equilibrium is also quasi-perfect, games from Figures 3.1(a), 3.1(b) and 3.2 are again solved properly. In addition Matching pennies on Christmas morning has unique normal-form proper equilibrium with $\delta(h') = \delta(t') = 0.5$.

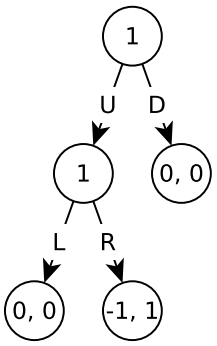


Figure 3.4: A game with unique perfect equilibrium

3.3.6 Perfect equilibrium

Solution concept due to Selten [16] that requires that each player at every information set takes a choice which is optimal against mistakes of all players (including himself) in the future and in the past. Let us first define ε-perfect equilibrium. Strategy profile B is ε-perfect equilibrium of extensive game G iff

$$U_i(I, a|B) < U_i(I, a'|B) \Rightarrow b_i(I, a) \leq \varepsilon, \quad \forall i \in \mathcal{P}, \forall I, \forall a, a' \in A(I) \tag{3.15}$$

Strategy profile is perfect equilibrium if it is the limit point of ε-perfect equilibria as ε goes to 0. Every perfect equilibrium is sequential, but the opposite doesn't hold and the set of perfect equilibria of any game is non-empty [16]. Note that perfect equilibria of normal-form game need not be perfect equilibria in corresponding extensive-form game and vice versa.

The game from Figure 3.4 has two normal-form proper (implying quasi-perfect, undominated etc.) equilibria (U, L) and (D, L), because none of the previously mentioned refinements consider mistakes of all players. The only perfect equilibrium of this game is (D, L), because action U is, according to perfection insensible thanks to the possibility of playing R by mistake.

3.3.7 Proper equilibrium

Solution concept due to Myerson [11] optimal against the mistakes of all players. These mistakes are assumed to be made in rational manner, meaning that the more costly mistakes are made with the probability of order of magnitude smaller than the less costly ones. A strategy profile in behavioral strategies is said to be proper if it is a limit point of ε-proper strategy profiles B, such that

$$U_i(I, a|B) < U_i(I, a'|B) \Rightarrow b_i(I, a) \leq \varepsilon \cdot b_i(I, a'), \quad \forall i \in \mathcal{P}, \forall I, \forall a, a' \in A(I) \tag{3.16}$$

as ε goes to 0. Every proper equilibrium is perfect and every extensive-form game has non-empty set of proper equilibria [22].

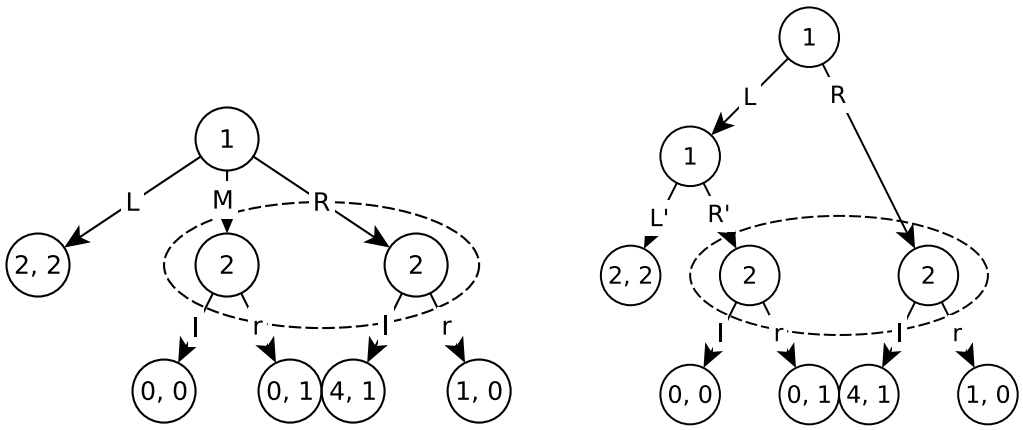


Figure 3.5: (a) The game where properness rules out all insensible equilibria (b) The game where normal-form properness rules out all insensible equilibria.

To demonstrate the improvement of the proper equilibrium over the perfect one, let us introduce the game from Figure 3.5(a), taken from [22]. There is a perfect equilibrium (L, r) supported by beliefs of player 2 that the mistake M has a bigger probability of occurrence than R. This belief is however not sensible since R dominates M. And so the only sensible equilibrium is (R, L) which is the only one consistent with the properness.

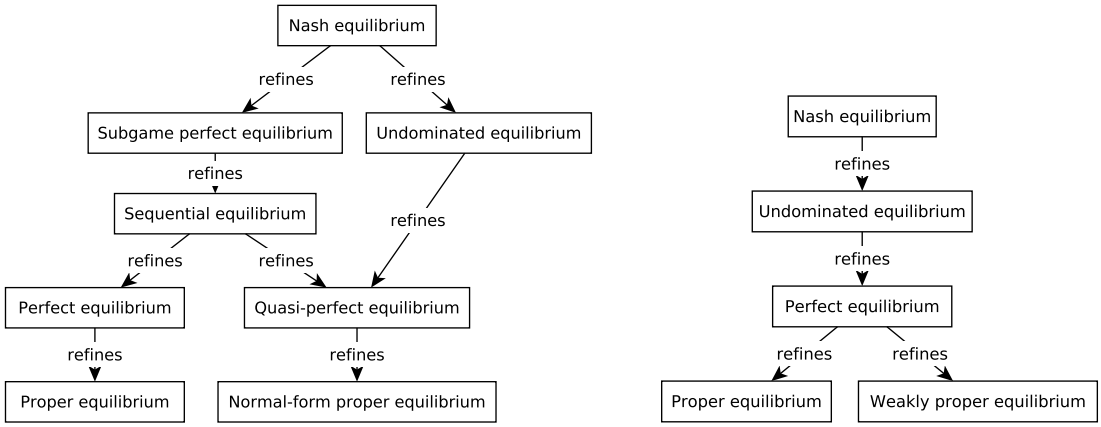


Figure 3.6: (a) The relations between refinements in two-player zero-sum extensive-form games. (b) The relations between refinements in normal-form games.

However thanks to the fact that only actions at the same information set are compared, not all insensible equilibria will be excluded. As an example see the game from Figure 3.5(b) taken from [22]. (L, L', r) is a perfect and proper equilibrium here. It is however not sensible since upon reaching his choice, player 2 should realize that player 1 will always prefer L' to R' and so he should expect that this set was reached by R, implying that he should play l.

So the only sensible equilibrium is (R, L', l). The only solution concept generating only this equilibrium as a result is normal-form proper equilibrium.

In this chapter the most known refinements of Nash equilibrium were introduced. The relations between these refinements are depicted in 3.6. Their theoretical strength was discussed using number of examples. This thesis will focus on two-player zero-sum extensive-form games and the refinements which exploit mistakes of the opponent, mainly because the Double oracle algorithm cannot take advantage of the optimality against mistakes of the player itself and also because it allows us to use fast linear programming methods based on efficient sequence-form representation. And so only undominated, quasi-perfect and normal-form proper equilibria will be used. Chapter 5 provides evaluation of these refinements on larger games, inspired in real world applications, to check if their real performance reflects their theoretical properties and to determine the most suitable refinement for application to the Double oracle algorithm.

Chapter 4

Algorithms for computing Nash equilibrium refinements

This chapter introduces algorithms behind the computation of Nash, undominated, quasi-perfect and normal-form proper equilibria. All of these algorithms make use of the linear programming exploiting sequence-form representation of extensive-form games. Main shortcomings of each algorithm, such as numerical stability issues, are discussed.

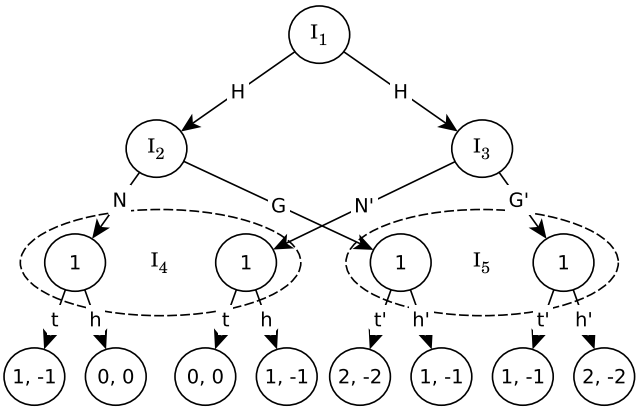


Figure 4.1: Matching Pennies on Christmas Morning [10]

4.1 Nash equilibrium

We first describe the algorithm for computing Nash equilibrium that exploits the sequence form due to Koller et al. [4]. In eqs. (4.1) to (4.4) we present a linear program (LP) for solving two player zero-sum extensive-form games. Matrix A is a utility matrix with rows corresponding to sequences of player 1 and columns to sequences of player 2. Each entry of A corresponds to the expected utility value of a game state reached by the sequence combination assigned to this entry, weighted by the probability of occurrence of this state

considering nature. If the reached state is non-terminal, or if the executing of actions from the sequence combination leads to a state, with no actions from the sequence combination applicable, yet there are still unplayed actions in the sequence combination, the entry is 0. Matrices E and F define the structure of the realization plans for player 1 and 2 respectively. Columns of these matrices are labeled by sequences and rows by information sets. Row for information set I contains -1 on a position corresponding to a sequence leading to I, 1 for the sequences leading from I and zeros otherwise. First row, corresponding to artificial information set has 1 only on position for empty sequence. These matrices ensure that for every information set I_i the probability with which we play a sequence leading to I_i is equal to sum of probabilities of sequences leaving I_i according to r_i. Vectors e, f are indexed by sequences of players and consist of zeros, but with 1 on the first position. q is a vector of variables representing values in information sets of the opponent. The constraint in (4.3) enforces the structure of realization plan and the constraint in (4.2) tightens the upper bound on value in each of the opponent's information sets I_2 for every sequence leaving I_2.

$$\max_{r_1, q} \quad f^\top q \tag{4.1}$$

$$s.t. \quad -A^\top r_1 + F^\top q \leq 0 \tag{4.2}$$

$$Er_1 = e \tag{4.3}$$

$$r_1 \geq 0 \tag{4.4}$$

As an example let us show the matrices and vectors needed for the computation of sequence-form LP of the game Matching pennies on Christmas morning from Figure (4.1) in the eqs. (4.5) to (4.7). λ_1 and λ_2 stand for the artificial information sets, which preceede the topmost information sets reached by empty sequence of the players making choice in them. The vectors e and f were omitted due to simplicity.

$$E = \begin{array}{c} \\ \lambda_1 \\ I_4 \\ I_5 \end{array} \begin{array}{c} \begin{array}{ccccc} [\,] & h & t & h' & t' \end{array} \\ \begin{pmatrix} 1 & & & & \\ -1 & 1 & 1 & & \\ -1 & & & 1 & 1 \end{pmatrix} \end{array} \tag{4.5}$$

$$F = \begin{array}{c} \\ \lambda_2 \\ I_1 \\ I_2 \\ I_3 \end{array} \begin{array}{c} \begin{array}{ccccccc} [\,] & H & T & HN & HG & TN' & TG' \end{array} \\ \begin{pmatrix} 1 & & & & & & \\ -1 & 1 & 1 & & & & \\ & -1 & & 1 & 1 & & \\ & & -1 & & & 1 & 1 \end{pmatrix} \end{array} \tag{4.6}$$

$$A = \begin{array}{c} \\ [\,] \\ h \\ t \\ h' \\ t' \end{array} \begin{array}{c} \begin{array}{ccccccc} [\,] & H & T & HN & HG & TN' & TG' \end{array} \\ \begin{pmatrix} & & & & & & \\ & & & 1 & & & \\ & & & & & 1 & \\ & & & & 2 & & 1 \\ & & & & 1 & & 2 \end{pmatrix} \end{array} \tag{4.7}$$

4.2 Undominated equilibrium

Undominated equilibrium is defined as a Nash equilibrium in undominated strategies. It can be computed using 2 LPs. First LP depicted in eqs. (4.1) to (4.4) solves the game for Nash equilibrium. The value of the game computed by this LP is then supplied to the second LP, presented in eqs. (4.8) to (4.12), via constraint (4.9) to ensure, that the realization plan r_1 computed by this LP is a Nash equilibrium. Second modification of this LP is in the objective, using r_2^m which is a uniform realization plan for the minimizing player.

$$\max_{r_1,q} \quad r_1^\top A r_2^m \tag{4.8}$$

$$s.t. \qquad f^\top q = v_0 \tag{4.9}$$

$$-A^\top r_1 + F^\top q \leq 0 \tag{4.10}$$

$$Er_1 = e \tag{4.11}$$

$$r_1 \geq 0 \tag{4.12}$$

The restriction to undominated strategies is enforced by the objective (4.8). The best response to a fully mixed strategy cannot contain dominated strategies and thus we have that r_1 is undominated and therefore normal-form perfect for two-player zero-sum games [22].

$$r_2^m = \begin{array}{ll} [\,] & \left(\begin{array}{c} 1 \\ 0.5 \\ 0.5 \\ 0.25 \\ 0.25 \\ 0.25 \\ 0.25 \end{array}\right) \\ H & \\ T & \\ HN & \\ HG & \\ TN' & \\ TG' & \end{array} \tag{4.13}$$

As an example we provide the matrices and vectors needed for computation of undominated equilibrium of the game Matching pennies on Christmas morning from Figure 4.1. The matrices E, F and A and vectors e and f are the same as in eqs. (4.5) to (4.7). The realization plan r_2^m used in the objective of the second LP is depicted in (4.13). The objective maximized in this LP is for clarity shown in (4.14). As we can see, the objective doesn't help at all in this domain, since for any $r_1(t')$ and $r_1(h')$ consistent with the definition of realization plan, the objective value remains the same. This is caused by the fact that neither of these strategies dominates the other.

$$\max_{r_1,q} \quad r_1(h) \cdot 0.25 + r_1(t) \cdot 0.25 \;+\; r_1(h') \cdot 0.75 + r_1(t') \cdot 0.75 \tag{4.14}$$

4.3 Quasi-perfect equilibrium

Quasi-perfect equilibrium is a restriction of Nash equilibrium, which prescribes the optimal play against mistakes of the opponent in the past and in the future. In (4.15) to (4.19) we present LP due to Miltersen et al. [9]. The main idea of this approach is to use symbolic

perturbations of strategies, with ε as a parameter, and then use a parameterizable simplex algorithm to solve this LP optimally. The results of such an algorithm are strategies expressed as polynomials in ε, which are then used to reconstruct the realization plans even in those information sets which are not reachable when considering a rational opponent (see [9] for the details of this transformation). Vectors $l(\varepsilon)$ and $k(\varepsilon)$ are indexed by sequences and contain above mentioned symbolic perturbations forcing this LP to create a quasi-perfect equilibrium. Vector v contains slack variables forcing the player to exploit the weak strategies of the opponent, matrices A, E and F, and vectors e,f, and q are as before.

$$\max_{r_1,v,q} \quad q^\top f + v^\top l(\varepsilon) \tag{4.15}$$

$$s.t. \quad F^\top q \leq A^\top r_1 - v \tag{4.16}$$

$$r_1 \geq k(\varepsilon) \tag{4.17}$$

$$Er_1 = e \tag{4.18}$$

$$v \geq 0 \tag{4.19}$$

Even though Miltersen et al. argue in [9] that it is possible to solve this LP using a non-symbolic perturbation, the ε required for such a computation can be too small for floating point arithmetics. Therefore, one either needs to use an unlimited precision arithmetics, or the parameterizable simplex algorithm to compute the equilibrium, which limits the scalability.

As an example let us introduce the matrices and vectors needed for computation of the quasi-perfect equilibrium on the game Matching pennies on Christmas morning from Figure 4.1. The matrices E, F and A and the vectors e and f are the same as in eqs. (4.5) to (4.7). The vectors $l(\varepsilon)$ and $k(\varepsilon)$, containing the symbolic perturbations, are depicted in eqs. (4.20) and (4.21). Every ε in these vectors has the power equal to the length of the corresponding sequence (ε^0 for $[\,]$, ε^1 for h etc.).

$$l(\varepsilon) = \begin{matrix} [\,] \\ H \\ T \\ HN \\ HG \\ TN' \\ TG' \end{matrix} \begin{pmatrix} 1 \\ \varepsilon \\ \varepsilon \\ \varepsilon^2 \\ \varepsilon^2 \\ \varepsilon^2 \\ \varepsilon^2 \end{pmatrix} \tag{4.20}$$

$$k(\varepsilon) = \begin{matrix} [\,] \\ h \\ t \\ h' \\ t' \end{matrix} \begin{pmatrix} 1 \\ \varepsilon \\ \varepsilon \\ \varepsilon \\ \varepsilon \end{pmatrix} \tag{4.21}$$

Vector v ensures that the realization plan r_1 is chosen in such a way that it exploits the mistakes made by the opponent. It uses the observation that the slack value of the constraints of type (4.16) corresponds to the exploitability of the opponents mistake in given information

set. In other words, when there is no slack in the constraint even though it is possible to achieve one, the opponents mistakes are not exploited in the corresponding informations set. The slack value effectively indicates how much one can gain over the expected value of given information set if the opponent plays the sequence corresponding to the constraint with slack. Therefore we want the slack to be present in every constraint of the type (4.16), if possible. This is exactly the purpose of the vector v, since by maximizing it, we maximize the slacks in the constraints. For clarity we depict the two constraints corresponding to the exploitable sequences containing the gift action of player 2 in the (4.22) and (4.23).

$$q(I_3) - 2r_1(t') - r_1(h') + v(TG') \leq 0 \tag{4.22}$$
$$q(I_3) - 2r_1(h') - r_1(t') + v(HG) \leq 0 \tag{4.23}$$

The issue with this approach is that the value of the objective depicted in (4.24) is the same in the case that the slack is maximal in the constraint (4.22), (4.23), or arbitrarily divided between these two. However to obtain the desired behavior $r(h') = r(t') = 0.5$ the slacks need to be equal, and so the quasi-perfect equilibrium doesn't guarantee optimality.

$$\begin{aligned} \max \quad & q(\lambda) + v([\,]) + \varepsilon \cdot v(H) + \varepsilon \cdot v(T) + \varepsilon^2 \cdot v(TN') + \\ & + \varepsilon^2 \cdot v(TG') + \varepsilon^2 \cdot v(HN) + \varepsilon^2 \cdot v(HG) \end{aligned} \tag{4.24}$$

4.4 Normal-form proper equilibrium

Normal-form proper equilibrium is a Nash equilibrium optimal against the mistakes of the opponent, while assuming that the probability of the mistakes depends on the potential loss for such a mistake. The algorithm for computing normal-form proper equilibria of extensive-form zero-sum games is due to Miltersen et al. [10] and it is based on an iterative computation of LP pairs V and W shown in eqs. (4.25) to (4.37). In the k-th iteration the LP V generates a strategy that exploits all marked exploitable sequences. The LP uses a set of vectors $\{m_1, ..., m_k\}$, where $m_i \in \{0,1\}^{|f|}$ represent labels that denote exploitable sequences based on the results of $W^{(i-1)}$, and set $\{v^{(1)}, ..., v^{(k-1)}\}$, where $v^{(i)}$ is a value of $V^{(i)}$; t is a scalar which is used in further iterations as $v^{(k)}$. The constraint (4.27) ensures, that the computed strategy is a Nash equilibrium.

$$V^{(k)}: \qquad \max_{r_1, q, t} \quad t \tag{4.25}$$
$$s.t. \quad -A^\top r_1 + F^\top q + m^{(k)} t \leq -\sum_{0<i<k} m^{(i)} v^{(i)} \tag{4.26}$$
$$f^\top q = v^{(0)} \tag{4.27}$$
$$E r_1 = e \tag{4.28}$$
$$r_1 \geq 0 \tag{4.29}$$
$$t \geq 0 \tag{4.30}$$

LP W in k-th iteration marks sequences, which are still exploitable, given previous iterations and $V^{(k)}$. Vector u is used to identify exploitable sequences and variable d is used as an auxiliary scalar for scaling purposes. This algorithm is initialized by $V^{(0)}$ which is a LP

generating Nash equilibrium from eqs. (4.1) to (4.4) and $W^{(0)}$ which is equal to $W^{(k)}$ only with the sum from constraint (4.32) omitted, since there are no results from previous iterations.

$$W^{(k)}: \quad \max_{r_1,q,u,d} \; 1^\top u \tag{4.31}$$

$$s.t. \quad -A^\top r + F^\top q + u \leq -\sum_{0<i\leq k} m^{(i)} v^{(i)} d \tag{4.32}$$

$$Er_1 - ed = 0 \tag{4.33}$$

$$f^\top q - v^{(0)} d = 0 \tag{4.34}$$

$$0 \leq u \leq 1 \tag{4.35}$$

$$r_1 \geq 0 \tag{4.36}$$

$$d \geq 1 \tag{4.37}$$

Although this procedure runs in polynomial time since the number of LP pairs is bounded by the number of sequences of the opponent, in practice this approach can suffer from numerical precision errors when used for solving larger games. The primary reason of this instability is the error that cumulates in equation (4.32).

Let us again demonstrate this approach on the game from Figure 4.1. First we need to calculate the sequence-form LP, the matrices E, F, A and the vectors e and f are again the same as in eqs. (4.5) to (4.7). Next step is the detection of exploitable sequences, consistent with the current formulation (note the constraints (4.27) and (4.34), which force every following LP to compute Nash equilibrium). The detection is performed using the variable vector u which contains 1 for those sequences which are associated with a constraint with slack. The result of W^0 is $u(HG) = 1$, $u(TG') = 1$, $u(H) = u(T) = u(TN') = u(HN) = 0$ and so the exploitable sequences marked by this LP are TG' and HG', based on the constraints (4.38) and (4.39). These constraints correspond to the sequence TG' and HG' and both have slack achievable in them.

$$q(I_3) - 2r_1(t') - r_1(h') - u(TG') \leq 0 \tag{4.38}$$

$$q(I_3) - 2r_1(h') - r_1(t') - u(HG) \leq 0 \tag{4.39}$$

The constraints corresponding to these sequences get the scalar variable t added in the following LP V^1. The resulting constraints are for clarity depicted in (4.40) and (4.41). The purpose of t is to balance slacks in constraints corresponding to exploitable sequences. Variable t is maximised in this LP and so the slack in both constraints will be equal and its value will be set to the minimum of maximal achievable slack in constraints (4.40) and (4.41). This ensures that r_1 will prescribe the desired behavior $r_1(h') = r_1(t') = 0.5$, because the slack will be balanced in both constraints. Note the improvement over the previous approach in the quasi-perfect equilibrium, where there was no quarantee that the slacks will be balanced, which caused the issues with the solution quality.

$$q(I_3) - 2r_1(t') - r_1(h') + t \leq 0 \tag{4.40}$$

$$q(I_3) - 2r_1(h') - r_1(t') + t \leq 0 \tag{4.41}$$

W^1 again attempts to detect the exploitable sequences consistent with the current formulation, but since constraints corresponding to the exploitable sequences TG' and HG (depicted in (4.42) and (4.43)) are already exploited, there are no more exploitable sequences and the algorithm terminates. The fact that there is no possible slack present in these constraints is ensured by the negative scaling variable d, added to the right side of these constraints.

$$q(I_3) - 2r_1(t') - r_1(h') - u(TG') \leq -d \tag{4.42}$$
$$q(I_3) - 2r_1(h') - r_1(t') - u(HG) \leq -d \tag{4.43}$$

This chapter introduced algorithms used for computation of Nash, undominated, quasi-perfect and normal-form proper equilibria. As mentioned, quasi-perfect and normal-form proper equilibria have limitations of applicability thanks to numerical stability issues and time constraints. The Chapter 5 resolves whether the performance makes up for these shortcomings.

Chapter 5

Refinement Comparison

This chapter compares the practical performance of the different variants of refinements of Nash equilibrium strategies. Since all the compared strategies are Nash equilibrium strategies, they cannot be exploited, and thus we are interested in the expected value of these strategies against imperfect opponents.

5.1 Imperfect Opponents

Two types of imperfect opponents were used. First type is not fully converged strategy from anytime algorithms used for solving extensive-form games in practice, and second is a game-theoretic model called Quantal-response equilibrium that simulates the decisions made by human opponents.

We use two different algorithms for generating the imperfect opponents of the first type: counter-factual regret minimization (CFR) algorithm [25] and Monte-Carlo tree search (MCTS).

5.1.1 Counter-factual regret minimization

CFR is a regret minimizing algorithm. The high-level idea of this algorithm is to iteratively traverse the whole game tree, updating the strategy with an aim to minimize the overall regret, defined as follows.

$$R_i^T = \frac{1}{T} \max_{b_i^* \in B_i} \sum_{t=1}^{T} (u_i(b_i^*, b_{-i}^t) - u_i(B^t)) \tag{5.1}$$

T denotes current iteration of the algorithm. The overall regret is decomposed to the set of additive regret terms, which can be minimized independently. These terms are called counterfactual regrets and are defined on the individual information sets as follows.

$$R_{i,imm}^T(I) = \frac{1}{T} \max_{a \in A(I)} \sum_{t=1}^{T} \pi_{-i}^{B^t}(I)(u_i(I, a|B^t) - u_i(I, B^t)) \tag{5.2}$$

$\pi_{-i}^{B^t}$ stands for the probability of reaching I given the opponent and nature, $u_i(I, a|B)$ is the expected utility in information set I when players play according to B^t except for i in I playing a and $u_i(I, B^t)$ is expected value in I when players play according to B^t. Intuitively it is players regret in I of following his strategy.

In [25] is shown that the sum of all counterfactual regrets is a upper bound of the overall regret. To minimize the counterfactual regrets, the algorithm maintains in all information sets

$$R_i^T(I, a) = \frac{1}{T} \sum_{t=1}^{T} \pi_{-i}^{B^t}(I)(u_i(I, a|B^t) - u_i(I, B^t)) \tag{5.3}$$

for all actions. We denote $R_i^{T,+}(I, a) = \max(R_i^T(I, a), 0)$. The update rule for the strategy is then defined as follows.

$$B_i^{T+1}(I, a) = \begin{cases} \frac{R_{i,imm}^{T,+}(I,a)}{\sum_{a \in A(I)} R_{i,imm}^{T,+}(I,a)} & \text{if } \sum_{a \in A(I)} R_{i,imm}^{T,+}(I, a) > 0 \\ \frac{1}{|A(i)|} & \text{otherwise} \end{cases} \tag{5.4}$$

This rule updates the strategy to minimize the counterfactual regrets, simultaneously minimizing the overall regret, causing convergence of average strategy profile defined as

$$\bar{B}^t(I, a) = \frac{\sum_{t=1}^{T} \pi_i^{B^t}(I) B^t(I, a)}{\sum_{t=1}^{T} \pi_i^{B^t}(I)} \tag{5.5}$$

to Nash equilibrium.

5.1.2 Monte-Carlo tree search

The MCTS is an iterative algorithm evaluating the domain based on a huge number of simulations and building of the tree containing most promising nodes. Each iteration consists of several steps.

1. Selection: Algorithm traverses the already build part of the tree choosing nodes based on some evaluation strategy. When it reaches node with no successors included in the partially build tree it expands this node according to the rules described next.

2. Expansion: When selection reaches leaf node of the partially build tree, expansion adds all of it's successors to this tree and runs simulation from one of them.

3. Simulation: Simulation performs one playthrough from given node to the terminal node of original game, with actions chosen randomly, by domain specific heuristic or based on opponents model.

4. Backpropagation: Backpropagation updates every node on the path from the root to the leaf node of the partially build tree with the value obtained from simulation.

MCTS is used in its most typical game-playing variant: UCB algorithm due to Kocsis et al. [3]

$$u_i = v_i + C\sqrt{\frac{ln(N)}{n_i}} \tag{5.6}$$

is used as the selection method and it is used in each information set (this variant is termed Information Set MCTS [2]). The u_i is a value of node i which will be used to choose the node, v_i is an average value of all previous visits of node i, n_i stands for the visit count of the node i and N number of visits of the parent node of i. C serves as a parameter tuning the eploitation and exploration of the tree. The lower C means that the value v_i is more important for node selection and higher C increases the value of less frequently visited nodes.

An additional modification made to MCTS is nesting—MCTS algorithm runs for certain number of iterations, and then advances to each of the succeeding information sets and repeats the whole procedure. This ensures equally reasonable strategies in all parts of the game tree, which is not the case in regular approach, since the deeper parts of the tree are visited less often then the parts closer to root.

The behavioral strategy over actions in each information set corresponds to the frequencies, with which the MCTS algorithm selects the actions in this information set. Contrary to CFR, there are no guarantees for convergence of this variant of MCTS in imperfect-information games [17].

5.1.3 Quantal-response equilibrium

The opponents of the second type correspond to quantal-response equilibrium (QRE) [8]. Calculation of QRE is based on a logit function with precision parameter λ [20]. The logit function prescribes the probability for every action in every information set as follows.

$$B(I,a) = \frac{e^{\lambda u(I,a|B)}}{\sum_{a' \in A(I)} e^{\lambda u(I,a'|B)}} \tag{5.7}$$

We can sample the strategies for specific values of the λ parameter. By setting $\lambda = 0$ we get uniform fully mixed strategy and when increasing λ we obtain a behavior, where players are more likely to make less costly mistakes rather then completely incorrect moves, with quaranteed convergence to sequential equilibrium when λ approaches ∞.

The iterative manner of MCTS and CFR allow sampling of the strategies before the full convergence is achieved, to generate opponents of increasing quality. Same can be achieved by repetitive computation of QRE with increasing λ.

5.2 Experimental domains

The performance of the refined strategies is compared on Leduc holdem, imperfect-information variant of the card game Goofspiel, and randomly generated extensive-form games. The games were chosen, because they differ in cause of imperfect information; for Leduc holdem poker the uncertainty is caused by the unobservable actions of nature at the beginning of

the game, while in imperfect information variant of Goofspiel and Random games the uncertainty is caused by partial observability of opponents moves. The size of evaluated games correspond to the maximal sizes of games, for which we were able to compute quasi-perfect and normal-form proper equilibrium in reasonable time and without numerical precision errors.

5.2.1 Leduc holdem poker

Leduc holdem poker [23] is a variant of simplified Poker using only 6 cards, namely $\{J, J, Q, Q, K, K\}$. The game starts with an ante of value 1, after which each of the players receives a single card and a first betting round begins. In this round player 1 decides to either *bet*, adding 1 to the pot, or to *check*. If he bets, second player can either *call*, adding 1 to the pot, *raise* adding 2 to the pot or *fold* which automatically ends the game in the favour of player 1. If player 1 checks, player 2 can either *check* or *bet*. If player 2 raises after a *bet*, player 1 can either *call* or *fold* ending the game in the favour of player 2. This round ends either by *call* or by *check* from both players. After the end of this round, one card is dealt on the table, and a second betting round with the same rules begins. After the second betting round ends, the outcome of the game is determined. A player wins if (1) her private card matches the table card, or (2) none of the players' cards matches the table card and her private card is higher than the private card of the opponent. If no player wins, the game is a draw and the pot is split.

5.2.2 Goofspiel

Goofspiel [14] is a card game with three identical packs of cards, two for players and one randomly shuffled and placed in the middle. In our variant both players know the order of the cards in the middle pack. The game proceeds in rounds. Every round starts by revealing the top card of the middle pack. Both players proceed to simultaneously bet on it using their own cards, which are discarded after the bet. Player with higher bet (higher value of card used) wins the card. After the end of the game, player with higher sum of values of cards collected wins.

5.2.2.1 Imperfect-information Goofspiel

In an imperfect-information version of Goofspiel, the players do not observe the bet of the opponent and after a turn they only learn whether they have won, lost, or if there was a tie.

5.2.3 Random games

Random games are games where several characteristics are randomly modified: the depth of the game (number of moves for each player) and the branching factor representing the number of actions the players can make in each information set. Moreover, each action of a player generates some observation signal (a number from a limited set) for the opponent – the states that share the same history and the same sequence of observations belong to the same information set. Therefore, by increasing or decreasing the amount of possible

observation signals we increase or decrease the number of information sets in the game (e.g., if there is only a single observation signal, neither of the players can observe the actions of the opponent). The utility is calculated as follows: each action is assigned a random integer value uniformly selected from the interval $-l, +l$ for some $l > 0$ and the utility value in a leaf is a sum of all values of actions on the path from the root of the game tree to the leaf. This method for generating the utility values is based on random *T-games* [18] that create more realistic games using the intuition of good and bad moves.

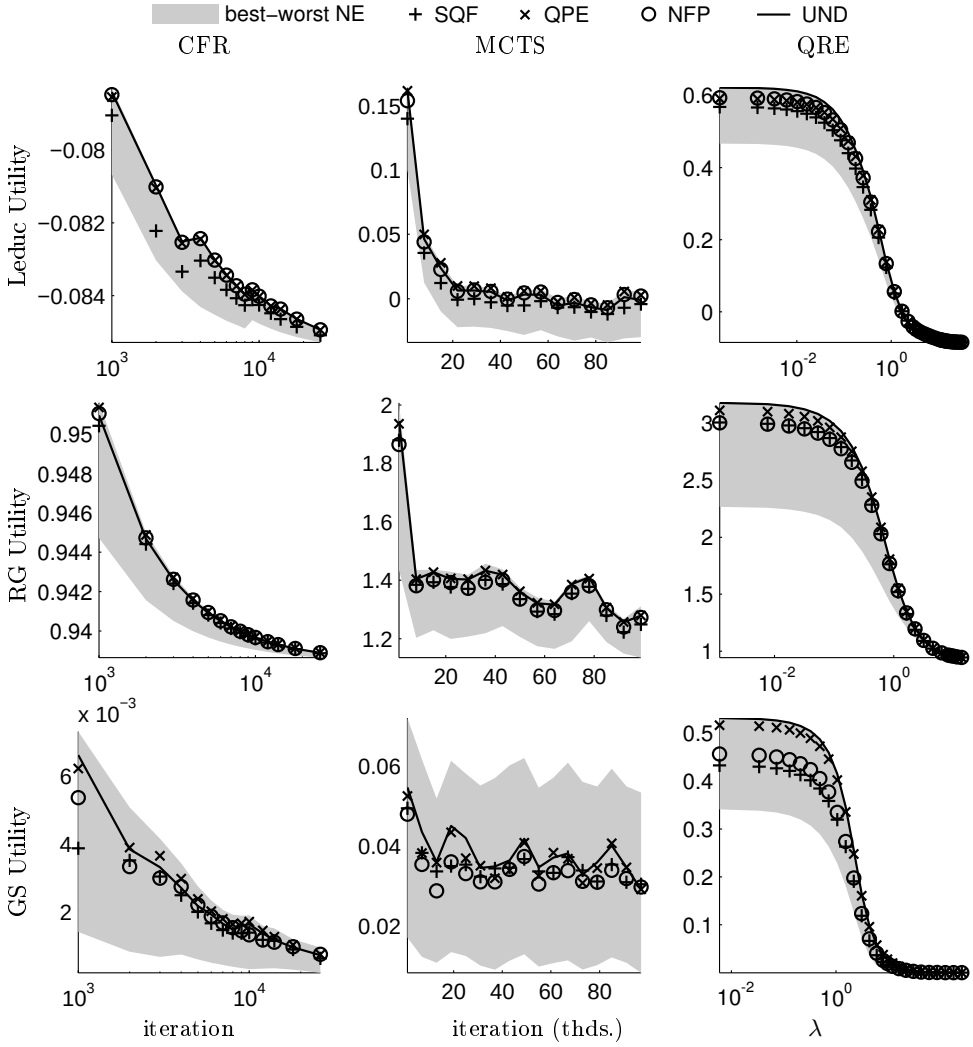


Figure 5.1: Overview of the utility value for different equilibrium strategies. Results for a single type of imperfect opponent are depicted in columns (CFR, MCTS, QRE), the results for a single domain are depicted in rows (Poker, Goofspiel, Random Games).

5.3 Experimental Settings

We have implemented the algorithms for computing Nash, undominated[1], and normal-form proper equilibrium, and we use IBM CPLEX 12.5 for solving LPs. We also implemented CFR and MCTS algorithm as described. We use Gtf framework[2] for computing quasi-perfect equilibrium and Gambit [7] for computing quantal-response equilibrium. The Gtf framework uses simplex with symbolic perturbations, which limits it's scalability.

We analyze the performance of the refined strategies within an interval determined by the worst and best possible NE strategy against a specific opponent strategy. These bounds are computed via the LPs used for the undominated equilibrium. To compute the best NE against a strategy, we use the strategy against which the refinements are currently measured in the objective of the second LP. Moreover, if we change the objective to min in such modified LP, we compute the worst NE.

5.4 Results

The overall absolute results are depicted in Figure 5.1, the interval between the worst and the best NE is the grey area; SQF denotes NE computed using sequence-form LP; UND denotes undominated equilibrium; QPE quasi-perfect; and NFP normal-form proper equilibrium. The relative results in the interval between the best and the worst NE are for clarity depicted in the Figure 5.2.

The first rows shows the absolute and relative utility values gained by different refinements against different opponents on Leduc holdem from the perspective of player 1 (note the logarithmic scale of x-axis in case of CFR and QRE). The results show that all the refinements have similar performance against all opponents and they all outperform SQF strategy. The similarity of NFP, QPE, and UND refinements can be demonstrated by the maximal difference in absolute utility values between refinements that is equal to 0.03—this occurs against QRE and it is caused by near-optimal performance of UND against QRE for small λ. This is expected since the QRE strategy for small λ is similar to uniformly mixed strategy, to which UND computes the best NE strategy. Besides that the absolute differences were mostly marginal: $6 \cdot 10^{-5}$ for CFR and $8 \cdot 10^{-3}$ for MCTS. The high relative differences for the QRE for high values of λ are caused by the interval of almost zero size. The interval is so small because the solution of QRE is very close to NE implying that all refinements score very close to the value of game. The differences are then caused by numerical precision issues, when computing relative value over this small interval.

Results on the random games with branching factor 3, depth 3 and 3 possible observations are shown in second rows of Figures 5.1 and 5.2. These results are computed as an average over 10 different random games generated with the selected properties but different structure of informations sets and utility values. The results are very similar as in poker, but the difference between the refinements and SQF decreased. The relative utility gain for QRE opponent confirms that for smaller λ the UND outperforms every other equilibrium, however with increasing λ the undominated equilibrium gets worse and both QPE and NFP improves

[1] We use fully mixed uniform strategy of the opponent as the input.
[2] Available at `http://www.cs.duke.edu/~trold/gtf.html`

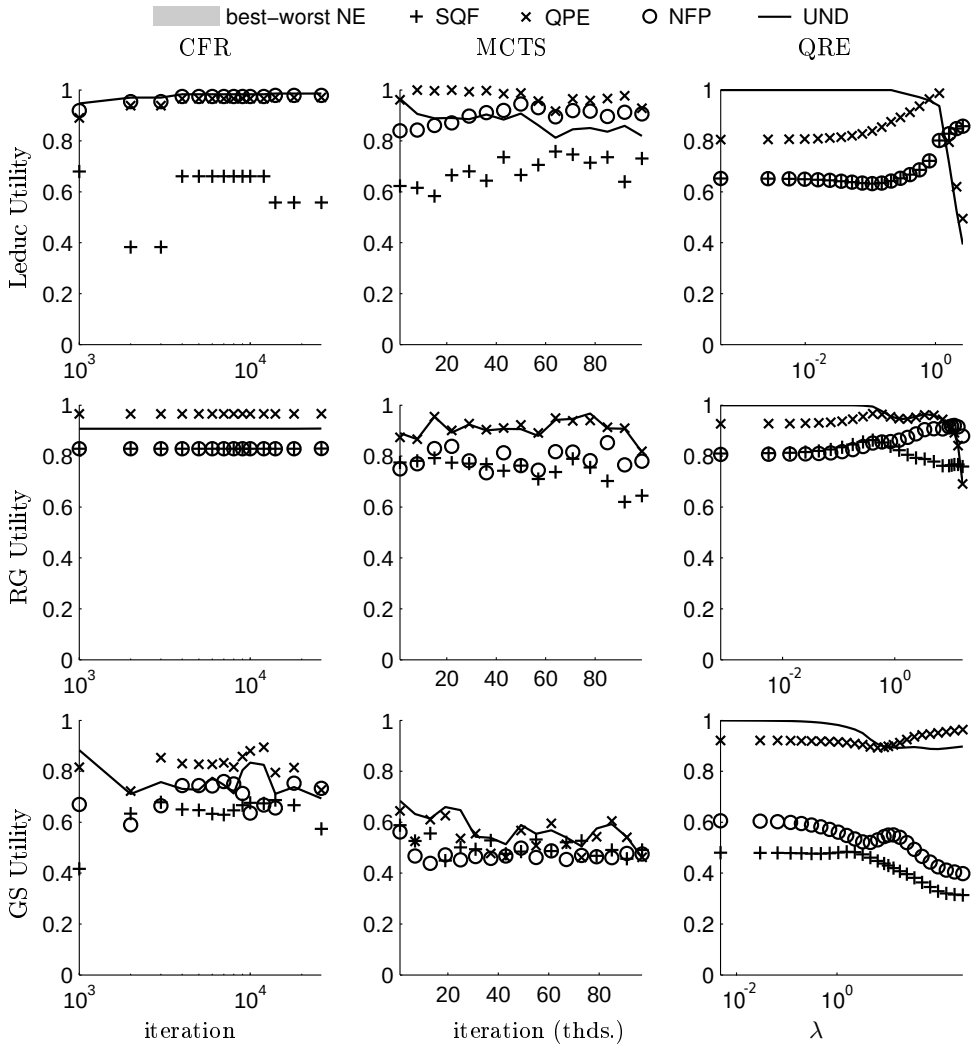


Figure 5.2: Overview of the utility value for different equilibrium strategies. Results for a single type of imperfect opponent are depicted in columns (CFR, MCTS, QRE), the results for a single domain are depicted in rows (Poker, Goofspiel, Random Games).

their performance as the QRE converges to more rational strategies. Moreover, we performed a different set of experiments by varying the size of the observation set. When set to 1, the game degenerates to a very specific imperfect-information game, where every action of a player is unobservable to the opponent. Interestingly, in this setting all NE collapsed, there was no difference between the worst and the best NE strategy, and thus neither between the refined strategies.

Finally, we present results on imperfect information Goofspiel with 4 cards in the third rows of Figures 5.1 and 5.2 (absolute utility values on y-axis for CFR are in $\times 10^{-3}$ due to very small differences). The results are again computed as means of 10 different random orderings of the middle pack of cards. Again, there is a very similar pattern of behavior against CFR and QRE opponents. Against the MCTS, however, the difference between the

refinements and the best NE strategy slightly increased. This is caused by the fact that the MCTS reaches an irrational strategy composed of the correct pure strategies, however, incorrectly mixed. This type of mistakes does not follow the model assumed in QPE and NFP, and neither UND can optimally exploit this strategy. This setting present the only case where further improvements in exploiting the mistakes of the opponent are possible.

Furthermore, to check, if the performance of undominated equilibrium remains consistent, we have performed measurement on larger games. We were unable to compare all the refinements here since the other refinements are unable to solve these domains. However the undominated equilibrium achieved similar performance with respect to the worst and best Nash equilibria.

The results on all domains and against all imperfect opponents offer several conclusions. First of all, all the refinements typically perform very well and close to optimal NE strategy. This indicates that it is unlikely that a new refinement with dramatically better performance can be defined. Secondly, the performance of all the refinements is very similar in practice (especially against CFR and MCTS) regardless of their theoretical assumptions. This is interesting and suggest that the occurrence of situations that allow to exploit the mistakes of the opponent are not that common in real-world games. Moreover, even though NFP considers likeliness of mistakes of the opponent with respect to the potential loss, its performance in practice was similar to QPE against this type of opponent. Finally, the presented results show that in practical applications, it is sufficient to use UND refinement: (1) the performance is very similar to QPE and NFP, (2) it is much easier to compute compared to more advanced solution concepts, since it does not require iterative process or unlimited precision arithmetic, and (3) due to the simple computation it allows solving of much larger games then the ones used in this thesis.

Chapter 6

Double-oracle algorithm

This chapter describes the Double-oracle algorithm operating on the sequence-form representation of two-player zero-sum extensive-form games due to Bošanský et al. [1].

This algorithm solves games iteratively. Each iteration consists of following steps

1. The algorithm creates a restricted game, where the players are allowed to select from a limited set of sequences and actions.
2. The restricted game is solved using sequence-form LP.
3. A best response algorithm is used to find suitable expansion of the restricted game.

The main strength of this algorithm lies in it's iterative manner. It solves a series of small LPs instead of single large one, trying to find as small restricted game as possible with the same solution as the original game. This approach allows solving of games too large for classical sequence-form LP along with a possible speed-up of computational time. For further discussion of the performance of Double-oracle algorithm see [1].

Let us now discuss each part of the double algorithm in more detail.

6.1 Restricted game

The restricted game is formed by sequences Σ' which are taken from the set of all sequences of the original game Σ in such a way that if $\sigma \in \Sigma'$ all the prefixes of σ also need to be contained in Σ'. Furthermore for every $\sigma_i \in \Sigma'_i$ there exists compatible sequence $\sigma_{-i} \in \Sigma'_{-i}$, meaning that the actions from these sequence can be all executed together.

6.1.1 Inconsistencies in the restricted game

There is an issues one needs to be aware of, when building restricted game. This issue arises when adding σ_i to the restricted game, containing no sequence σ_{-i} which would allow full execution of σ_i. This effectively means that the execution creates a leaf node h of the restricted game, which is an inner node of the original game, since it leads to a state where

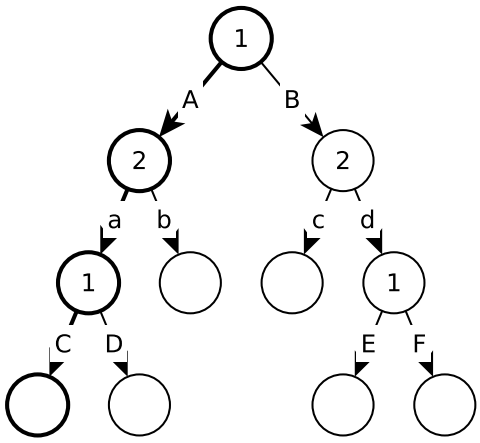


Figure 6.1: A demonstration of inconsistencies in the restricted game

player $-i$ plays, but there is no action in the restricted game applicable. The actual problem arrives when building the sequence-form LP, which requires utility value associated to every terminal node. To solve this inconsistency h becomes a temporal leaf node in the restricted game and gets a temporary utility value which needs to be set in a way which guarantees the convergence of the whole algorithm. To ensure the convergence, the value must be a lower bound for player $-i$ playing in h. This implies that if h is not expanded, even though it has the upper bound on its real utility value for player i assigned (meaning that not even the upper bound makes h relevant to the solution), the real expected value will definitely not be relevant to the solution. In our implementation we use the best response for player i against the default strategy of player $-i$ to get this value for player $-i$. The default strategy is a strategy which uses the result computed by the sequence-form LP in the restricted game space, expanded by the pure strategy returning first available action in every state outside of the restricted game.

As an example of the inconsistency in the restricted game consider the game from Figure 6.1. The restricted game created by previous iterations of the algorithm is $\Sigma_1 = \{A, AC\}$, $\Sigma_2 = \{a\}$. The best response algorithm in the current iteration returns sequences B, BE for player 1. When we try to add these sequences to the existing restricted game, we run into a problem, because player 2 has no action to execute in the game state h reached by sequence B. And so h becomes a temporal leaf node, with the utility value computed as described above. The resulting restricted game will be $\Sigma_1 = \{A, AC, B\}$, $\Sigma_2 = \{a\}$.

6.2 Best response algorithm

The best response algorithm computes a pure strategy which is the best response to the optimal strategy of the opponent of the player searching for the best response i in the current restricted game along with its value. It traverses the game tree in depth-first search manner with behavior of the opponent fixed to the strategy given by LP in the space of the

restricted game and to the default strategy outside of it. During the depth first search a bound is computed. This bound is used for domain independent pruning as will be shown later. To completely describe the best response algorithm, we need to divide it to two cases.

6.2.1 Nodes of the other players

Data: h - current node, I_i current information set, r_{-i} opponent's real. plan extended by default strategy, λ - lower bound for h
Result: expected value in h

```
1  if h ∈ Z then
2  |  return u_i(h) · r_{-i}(seq_{-i}(h)) · C(h)
3  end
4  w = Σ_{a∈A(h)} r_{-i}(seq_{-i}(h · a)) · C(h · a)
5  sort A(h) based on the probability of their occurrence
6  v^h = 0
7  for a ∈ A(h) do
8  |  w_a = r_{-i}(seq_{-i}(h · a)) · C(h · a)
9  |  λ' = λ − (v^h + (w − w_a) · u_max)
10 |  if λ' ≤ w_a · u_max then
11 |  |  v^h = v^h + bestResponse(h · a, λ')
12 |  |  w = w − w_a
13 |  else
14 |  |  return u_min · w
15 |  end
16 end
17 return v^h
```

Algorithm 1: Best response algorithm for a node of the opponent or nature

First we analyse the behavior in the nodes of other players, the opponent and the nature. Here we compute the expected value of given node based on the fixed behavior given by sequence-form LP result, default strategy or fixed distribution of nature. The actions to be evaluated are for performance reasons sorted in the descending order, according to the probability of their occurrence. For every recursive call we compute the lower bound, based on the sum of values of evaluated actions and assuming that the unevaluated actions all yield the maximal achievable utility (line 9 in Algorithm 1). If the lower bound exceeds the maximal achievable utility in given node, a cut-off occurs (line 14 in Algorithm 1).

6.2.2 Nodes of the searching player

Data: h - current node, I_i current information set, r_{-i} opponent's real. plan extended by default strategy, λ - lower bound for h
Result: expected value of the best action in h

1 **if** $h \in \mathcal{Z}$ **then**
2 | **return** $u_i(h) \cdot r_{-i}(seq_{-i}(h)) \cdot C(h)$
3 **end**
4 $H' = \{h' : h' \in I_i\}$
5 sort H' according to $r_{-i}(seq_{-i}(h')) \cdot C(h')$
6 $w = \sum_{h' \in H'} r_{-i}(seq_{-i}(h')) \cdot C(h)$
7 $a_{max} = empty$
8 $v_a = 0, \forall a \in A(h)$
9 **for** $h' \in H'$ **do**
10 | $w_{h'} = r_{-i}(seq_{-i}(h')) \cdot C(h')$
11 | **for** $a \in A(h')$ **do**
12 | | **if** a_{max} *is empty* **then**
13 | | | $\lambda' = w_{h'} \cdot u_{min}$
14 | | **else**
15 | | | $\lambda' = v_{a_{max}} + w \cdot u_{min} - (v_a + (w - w'_h) \cdot u_{max})$
16 | | **end**
17 | | **if** $\lambda' \leq w_{h'} \cdot u_{max}$ **then**
18 | | | $v_a^{h'} = bestResponse(h' \cdot a, \lambda')$
19 | | | $v_a = v_a + v_a^{h'}$
20 | | **end**
21 | **end**
22 | **if** $h' == h$ **and** $v_{a_{max}}^{h'} < \lambda$ **then**
23 | | **break**;
24 | **end**
25 | $a_{max} = \arg\max_{a \in A(h')} v_a$
26 | $w = w - w_{h'}$
27 | store $v_{a_{max}}^{h'}$
28 **end**
29 **return** $v_{a_{max}}^{h}$

Algorithm 2: Best response algorithm for a node of the searching player

In the nodes of the searching player the algorithm chooses the best action based on all the states in the current information set but returns its value only in the current game state. The game states of the information set are evaluated in descending order according to their probability of occurrence, given the strategy of the opponent and nature. Once again there is a lower bound computed for every recursive call (lines 13 and 15 in Algorithm 2). This bound represents the value required to choose current action as the best one and is again computed under the assumption that all the unevaluated actions will achieve the best

possible value. The algorithm stores the best action, as it can be used when any other state of current information set gets visited.

6.3 Player selection

In the implementation used in this thesis, every iteration of the main loop evaluates the sequence-form LP from the perspective of 1 player who has currently the worst bound on the solution quality. More precisely at the end of every iteration a player is chosen who's upper bound on utility value is farther from the current value of the restricted game.

6.4 Termination

The algorithm terminates, when the whole strategy returned by the best response algorithm is already contained in the restricted game for both players. This occurs only when the restricted game already contains all the sequences needed to solve the complete game.

6.5 Main loop

Every iteration of the Double oracle algorithm runs according to following rules

1. Compute the best response for player i to the realization plan of $-i$ from previous iteration (empty if first iteration)
2. Add the sequences from the best response algorithm and form the valid restricted game, if there is nothing to add, terminate
3. Solve the restricted game using sequence-form LP
4. Choose the player i for the next iteration

6.6 Refinements in Double oracle

It is important to realize that the Double oracle, when used with refined solver, doesn't guarantee the output consistent with any refinement, since the solution returned is computed on the last restricted game. It has therefore no implication on the solution quality in the complete game, since the restricted game is usually not fully build.

There are two main motivations behind the replacement of the sequence-form LP by the refined solver in the Double oracle algorithm;

(1) The presence of the strategies rational in all parts of the restricted game tree computed by the refined solver should improve the quality of the best response returned by the best response algorithm. The best response strategy is by definition exploitable and the quality of the strategy to which is the best response computed directly influences the quality of the best response. This effectively means that there should be less sequences, irrelevant to the

final solution, added to the restricted game. This should lead to smaller LPs and therefore faster and less memory consuming computation. This approach is experimentally evaluated in the next chapter.

(2) The use of the expected values computed by the refined solver in all the informations sets (vector q in all the LP formulations of solvers) for additional pruning in the best response algorithm. One could argue that we can use these values even from the sequence-form LP. This is true, however in the sequence-form LP, there is no guarantee of rationality outside of the equilibrium path, and so these values can be smaller, therefore less precise, than the ones from the refined solver. Since there are separate LPs for each player we have two sources of these values. When we search for the best response of the player i we can use the values from the LP where i maximizes, obtaining values in information sets of $-i$, or from the LP where player $-i$ maximizes, obtaining values in the information sets of i. As we will show neither of these values help us when pruning in the best response algorithm. The values over the information sets of player $-i$ doesn't help us in the pruning, since the best response algorithm operates directly over the concrete game states of player $-i$ (as shown in Algorithm 1). The values over information set of i are unuseful, since there is no way to decompose the value for the whole information set to the values of the concrete game states. Furthermore, these values are the lower bounds of the values obtainable by the best response algorithm, since the values from LP are achieved by the strategies consistent with Nash equilibrium. As mentioned above, the best response doesn't have such constraint, since it is optimal only against the fixed strategy of opponent, and so the values achievable are greater or equal to the values obtained from the LP. To make the pruning in the best response more efficient, we need values over concrete states of i (these values could be used to tighten the lower bound of corresponding states, since we are sure they are achievable), or at least an upper bound of the value achievable in given information set to be used in the computation of the bound for each state (it could be used in line 15 in Algorithm 2 to tighten the lower bound, since now the algorithm assumes that every unevaluated state achieves maximal utility). Since the LP provides neither, there is no significant improvement possible.

This chapter described the Double oracle algorithm. In the next we will present changes in performance, when the sequence-form LP gets replaced by solver generating one of the refinements of Nash equilibrium.

Chapter 7

Results

This chapter discusses the performance changes caused by the replacement of the sequence-form LP in the Double oracle algorithm by the undominated equilibrium solver. The choice of undominated equilibrium was based on the results presented in Chapter 5.

7.1 Experimental domains

The experimental domains chosen for this measurement were the Generic poker, Goofspiel (described in 5.2.2) and Border patrol. They were chosen to demonstrate the performance of Double oracle in games with different sources of imperfect information. The imperfect information in poker is caused by the unobservable actions of nature at the beginning of the game. In the Goofspiel, only the simultaneity of moves causes any uncertainty. In the Border patrol it is caused by the unobservable moves of the opponent.

7.1.1 Border patrol

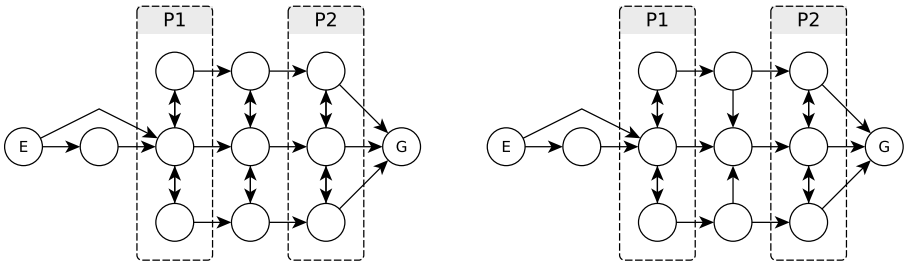


Figure 7.1: (a) Border patrol on a connected grid (b) Border patrol on a partially connected grid

Border patrol represent security scenario with simultaneous moves played on a grid. Two examples of grids used in the experiments in this chapter are depicted in the Figure 7.1. One player controls two patrolers, which move on a limited space (groups of nodes labeled $P1$

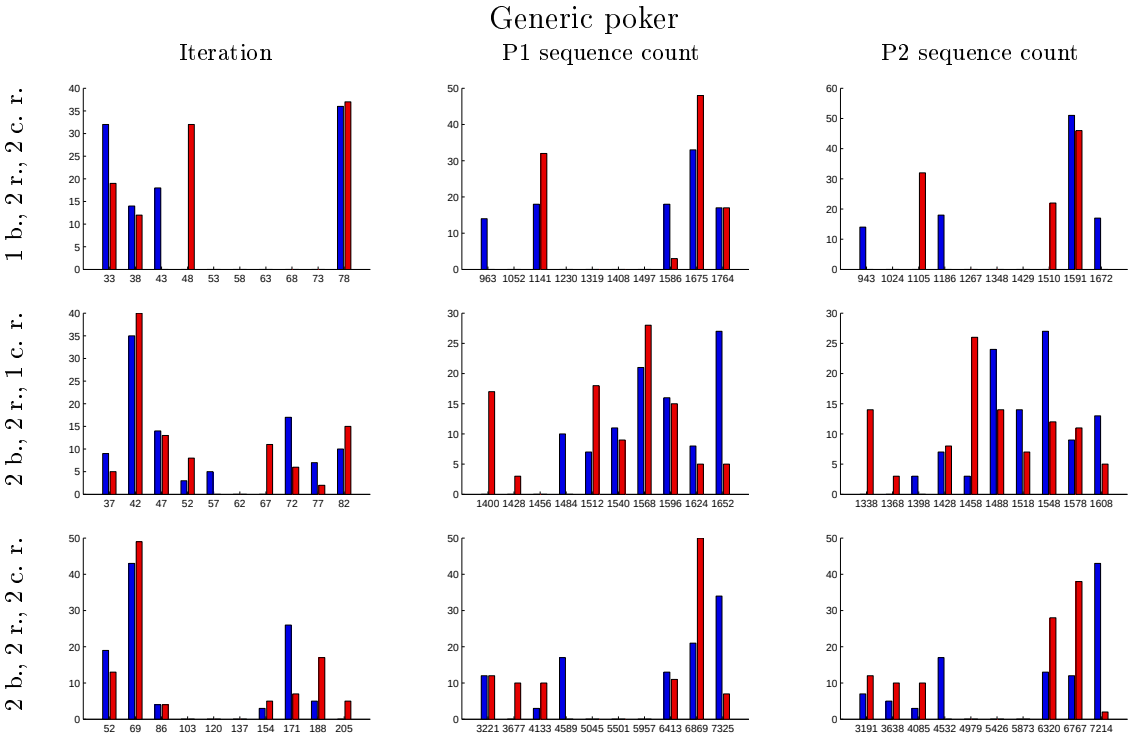


Figure 7.2: Overview of the histograms of Double oracle performance on Generic poker, over 100 different orderings of actions. The graphs show the histograms for the iterations needed for Double oracle to converge and the size of the final restricted game. The results for undominated solver are in red, for sequence-form LP in blue.

and $P2$), and where one attacker tries to move through from the start labeled in the Figure as E to the goal location, labeled as G. Neither of the players knows the location of his opponent. The game ends when the attacker reaches the goal, when he gets caught by one of the patrolers (either by stepping on the same position, or moving through the same edge) or when a given number of steps takes place. The attackers wins only when he reaches his goal.

7.1.2 Generic poker

Generic poker is a generalization of Leduc holdem poker described in 5.2.1, with adjustable deck of cards, number and value of bets and raises and even maximal number of consecutive raises.

7.2 Experimental setting

We have implemented the Double oracle algorithm and used IBM CLEX 12.5 in both sequence-form LP and undominated solver. The measurement was performed using 100 different orderings of constraints of both LPs.

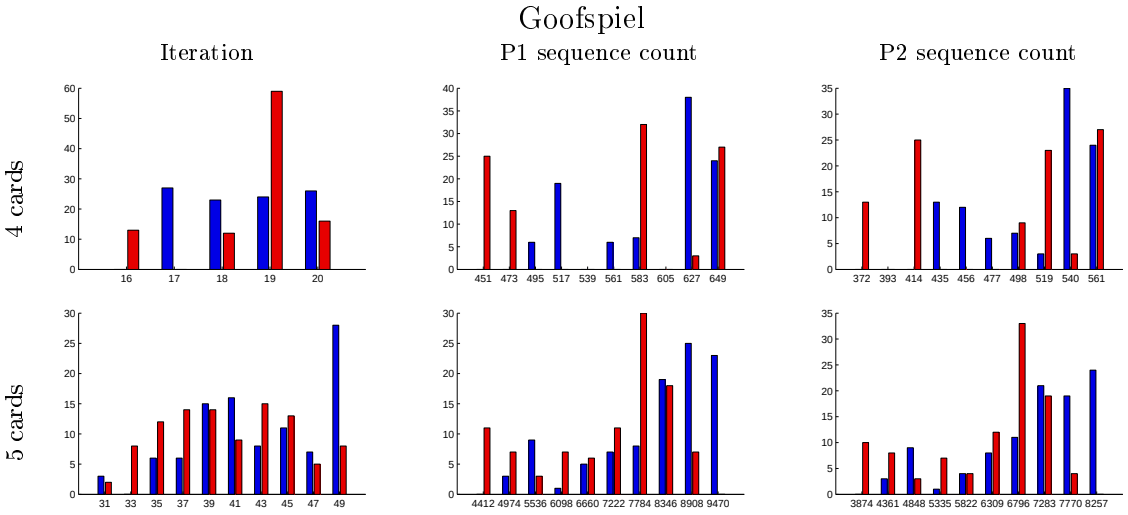


Figure 7.3: Overview of the histograms of Double oracle performance on Goofspiel, over 100 different orderings of actions. The graphs show the histograms for the iterations needed for Double oracle to converge and the size of the final restricted game. The results for undominated solver are in red, for sequence-form LP in blue.

7.3 Results

The overall results of performance of Double oracle algorithm with undominated solver and sequence-form LP are depicted in the Figures 7.2, 7.3 and 7.4. The histograms are created over 100 different orderings of actions (changing the order of constraints in LPs).

The Figure 7.2 shows results for the Generic poker. The experiments were conducted on poker with varying number of bets (b.), raises (r.) and continuous raise count (c.r.). As we can see there is no clear dominance between solvers, furthermore the result vary significantly with different orderings of constraints. The average size of the restricted game needed by undominated solver is slightly smaller than for sequence-form LP (for example in Generic poker with 2 b., 2 r., 2 c. r. 11496 sequences for undominated solver and 12218 sequences for sequence-form LP), the average number of iterations were on the other hand slightly smaller for sequence-form LP (again for Generic poker with 2 b., 2 r., 2 c. r. 100.82 iterations for sequence-form LP and 105.12 iterations for undominated solver). There is however no guarantee that the real performance on some fixed ordering will be consistent with this result.

The Figure 7.3 shows the results on Goofspiel with 4 and 5 cards in every deck. As we can see there is again no clear dominance of any solver, even though the average values were consistent with the results on poker. There was additional success of undominated solver in average number of iterations on the Goofspiel with 5 cards (44.45 iterations for sequence-form LP and 41.01 for undominated solver).

The Figure 7.4 contains the results on Border patrol with the depth of 5 and 6 and on two different grids depicted in Figure 7.1. The results are again very similar to poker for Border patrol on partially connected grid, with a small difference in iterations (5.57 for sequence-form LP and 5.73 for undominated solver) needed and with the size decrease of restricted game diminishing (2547 vs. 2550 sequences for depth 5 and 9320 sequences

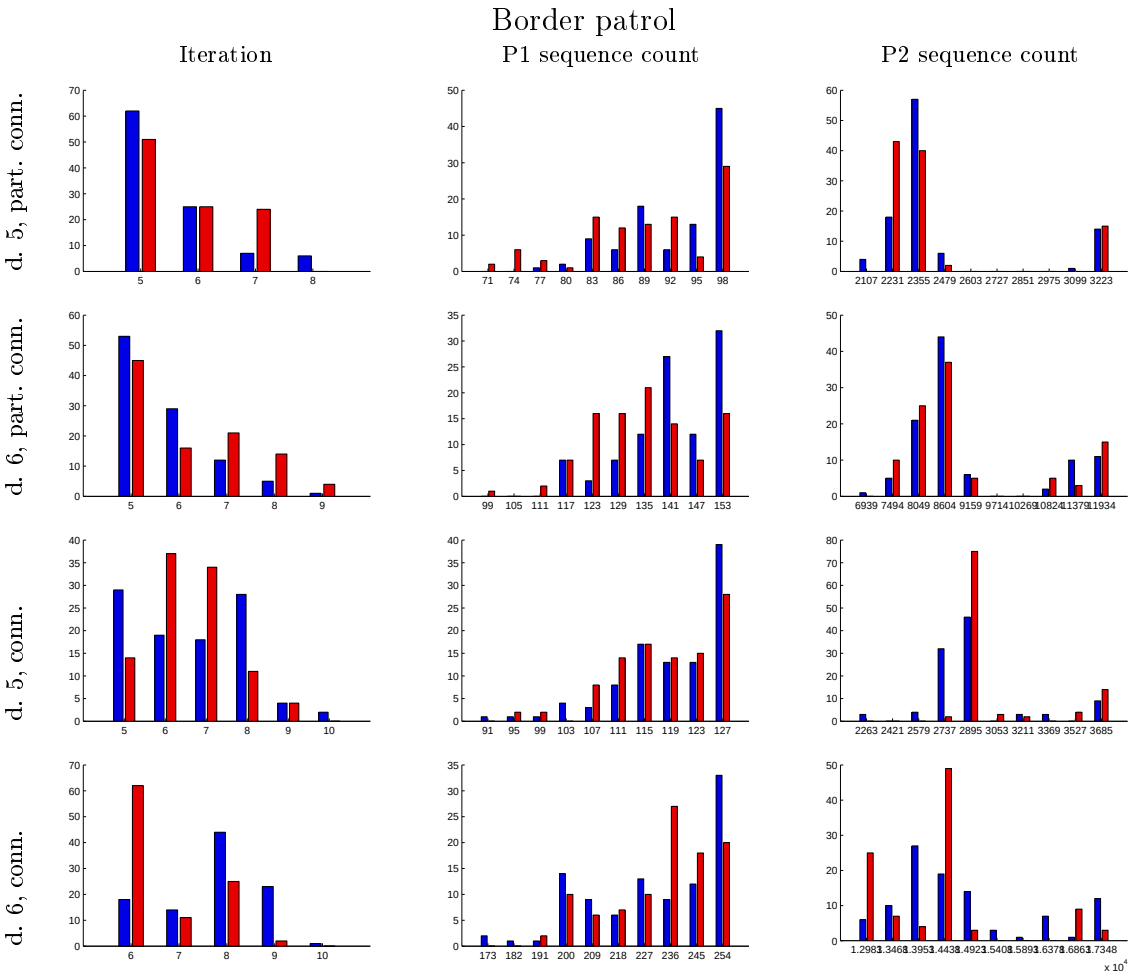


Figure 7.4: Overview of the histograms of Double oracle performance on Border patrol, over 100 different orderings of actions. The graphs show the histograms for the iterations needed for Double oracle to converge and the size of the final restricted game. The results for undominated solver are in red, for sequence-form LP in blue.

vs. 9341 sequences for depth 6). The results obtained for Border patrol on connected grid however show better average performance when considering iterations, when considering the size of the restricted game, there is no clear dominance between algorithms.

Finally we provide the overview of time spend solving LPs during the Double oracle computation depicted in Figure 7.5. These graphs show, that the undominated solver spends more time solving LPs than sequence-form LP on all domains. This is expected since the undominated solver needs to solve 2 LPs every iteration, while the sequence-form LP needs to solve only 1.

The results show that there is no clear dominance between solvers, since the results overlap each other, due to high sensitivity of CPLEX solver to the ordering of constraints. When considering the average results, the undominated solver needed smaller restricted game to solve the complete game (with exception of Border patrol on connected grid, where both

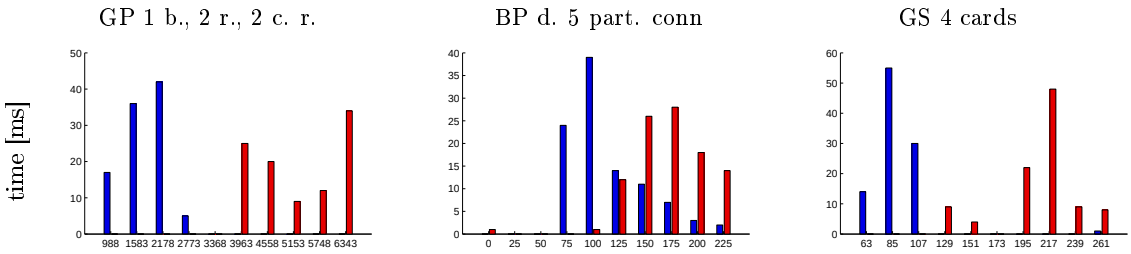


Figure 7.5: Overview of the histograms of overall time needed to solve all the LPs during the Double oracle computation. The results for undominated solver are in red, for sequence-form LP in blue.

solvers achieved similar results), which is a positive result since it implies less memory needed to solve large domains. It however needs more iterations to successfully converge to Nash equilibrium (again with exception of specific Goofspiel and Border patrol). This, combined with the fact, that the undominated solver spends more time solving LPs, suggests that one should prefer the undominated solver when the memory usage is the issue, the sequence-form LP on the other hand should be used when issues with time needed for solution arise.

7.4 Theoretical analysis

To better understand the reason behind small differences between both solvers, let us demonstrate the run of Double oracle algorithm on the game from Figure 7.6 with both the undominated and the sequence-form solver. Note that this game has 2 Nash equilibria (U, L), (U, R) but only one undominated equilibrium (U, R).

At the beginning the Double oracle computes the best response for player 1. Since there is no strategy of player 2 from the previous iteration, the best response algorithm assumes default strategy $r_2(L) = 1$ (L is chosen because it is the first action available in the state where player 2 plays). As we can see both D and U yield the expected value of 0 against this strategy and so the choice is based entirely on the implementation of the best response algorithm. For clarity we will discuss both cases.

Lets assume that the best response algorithm returned U. The restricted game created is trivial, since it contains only one sequence and so the solver, be it sequence-form LP or undominated solver, returns strategy $r_1(U) = 1$. In the second iteration the best response of player 2 to this strategy is computed. But since playing U doesn't lead to the state where player 2 plays, no sequence for player 2 is added and the algorithm terminates.

In the second case, when D is returned in the first iteration, the solver again solves trivial restricted game consisting of a single sequence D, generating the result $r_1(D) = 1$. In the second iteration, best response for player 2 to this strategy is again computed, now in the case where the state where player 2 plays is reached. Since the expected value of L is 0 and expect value of R is 1, the output of the best response algorithm is R. The restricted game, now containing D and R, has again a trivial solution $r_2(R) = 1$. In the next iteration player 1 is chosen and the best response algorithm against $r_2(R) = 1$ returns U, since it has expected value of 0 which is higher then -1 for D. The solver then again solves the restricted

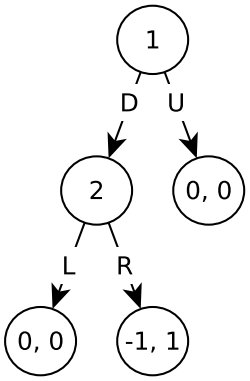


Figure 7.6: Domain for demonstration of Double oracle with different solvers

game containing R and $\{D, U\}$ computing strategy $r_1(U) = 1$. In the next iteration the best response for player 2 doesn't add anything new, since the state where player 2 plays isn't reached and the algorithm again terminates.

As we can see the above mentioned computation is identical for both solvers, even though their results are different when applied to the complete game. This fact suggests an explanation for the results in this chapter. The solvers operate on the restricted game, which has a special structure, since it's iteratively constructed from best responses of players, and so the situations, where there are mistakes exploitable, are not common. This implies that even on games where the undominated equilibrium offers better behavior than Nash equilibrium, there is no guarantee of speed-up when considering Double oracle computation.

Chapter 8

Conclusion

This thesis describes normal-form and extensive-form representation of games along with the overview of existing solution concepts used to solve these games. From these solution concepts undominated, quasi-perfect and normal-form proper equilibria were experimentally evaluated on variety of real world inspired games. The description of the Double oracle algorithm and a proposal of the approach to increase its performance using the most suitable of the evaluated solution concepts follows. Finally we performed experimental evaluation of the Double oracle using this solution concept with standard Double oracle algorithm on domains with different sources of imperfect information.

The experimental evaluation of the refinements produced several surprising results. First, all the measured refinements achieved very similar results, even though their theoretical performance varies dramatically. The most encouraging implication of this result is that one can use the undominated equilibrium, which is the easiest to compute, without the need to worry about the loss of the solution quality. Second, all the refinements scored very close to the best achievable value on given domains and against given opponents, which implies that at least for the zero-sum games there is not much space for improvement of the solution quality by introducing new, even more complex solution concepts.

Based on the results of the experimental evaluation we have used the solver generating undominated equilibrium in the Double oracle algorithm and measured its performance compared to the sequence-form LP on various domains with different sources of imperfect information. The results obtained by this measurements are however not strictly positive. Even though the undominated solver provided improvement in the average size of the restricted game needed for solution on almost all domains, there were still exceptions where the results of undominated and sequence-form solver were almost identical. From the perspective of iterations is the situation reversed. When using the undominated solver, the number of iterations was typically higher than when using the sequence-form LP. Furthermore the results varied dramatically when using different orderings of the constraints in LPs and so there is no guarantee that one will obtain better values when using undominated solver even on those domains where it is on average better. This results suggest that in general there is no motivation to use refinements in the Double oracle algorithm, since the increase in performance is not guaranteed, and is often overshadowed by increased resources needed to compute even the simplest refinement in comparison to the sequence-form LP. However as

the results suggest, there exist domains on which the undominated solver offers improvement in both the average size of the restricted game and the average number of iterations needed.

8.1 Future work

This thesis offers two straightforward directions for future work. First direction could aim for deeper analysis of the behavior of the refinements on more general games, such as general sum games. Second direction is to further experiment with the Double oracle algorithm, for example to use some domain specific improvements to solve poker or to introduce some alternative to default strategy to obtain closer bounds on the expected utility in the temporal leafs along with more reasonable best responses.

Appendix A

CD Content

Attached CD contains source files of the Game theoretical library, including all the solvers and experiments implemented as a part of this thesis, along with two runnable jar files used for experiments comparing refinements and Double oracle performance. Note that all the experiments require IBM CPLEX 12.1 installed to run successfully, furthermore the refinement comparison requires Gambit tool [7] installed.

A.1 Game theoretical library

The implementation of the Double oracle algorithm, together with the best response algorithm and solvers, is contained in the package *algorithms.sequenceform.doubleoracle*. The class running experiments for comparison of refinements is called RefCompExperiments and is located in the package *algorithms.sequenceform*. The class running experiment for measurement of performance of Double oracle is called DoubleOracleRefExperiments and is located in the package *algorithms.sequenceform.doubleoracle*. The implementation of standalone solvers computing refinements are located in package *algorithms.sequenceform.refinements*. The domains used through this thesis can be found in the package algorithms.domains.

A.2 Parameters of experiments

A.2.1 Refinement comparison experiment parameters

Following are the parameters needed to successfully run refComparison.jar.

1. Number of samples of the MCTS opponent strategy.
2. Iteration count between samples of MCTS.
3. How many times should be the MCTS rerun.
4. Number of samples of the CFR opponent strategy.
5. Iteration count between samples of CFR.

6. Domain type:
   (a) IIGS (Imperfect-information Goofspiel): number of cards, depth, number of different orderings of the cards in the middle deck
   (b) KP (Kuhn poker): no parameters
   (c) RG (Random game): max. branching factor, max depth, max. observations, center modification, utility correlation (true/false), bin. utility (true/false), seed count
   (d) GP (Generic poker): card type count, card count per type

For clarity we provide an example: "*java -Djava.library.path="path_to_CPLEX" -jar refComparison.jar 100 1000 10 100 1000 RG 2 2 2 3 true false 10*". This command will evaluate all the refined solvers against MCTS, CFR and QRE opponents on the parametrized Random game.

A.2.2 Double oracle performance experiment parameters

Following are the parameters needed to successfully run DOPerformance.jar.

1. LP solver (nash/undom)
2. Player selection heuristic (both/single_alt/single_impr)
3. Domain type:
   (a) GS (Goofspiel): number of cards, depth, number of different orderings of the cards in the middle deck
   (b) KP (Kuhn poker): no parameters
   (c) GP (Generic poker): number of bets, number of raises, number of cont. raises, card type count, card count per type
   (d) BPG (Border patroll): depth
4. number of different orderings of constraints

For clarity we provide an example: "*java -Djava.library.path="path_to_CPLEX" -jar DOPerformance.jar nash single_imp BP 5 100*". This command will run 100× Double oracle with sequence-form LP, using the player selection described in Chapter 7, on Border patroll with depth 5.

Bibliography

[1] Branislav Bošanský, Christopher Kiekintveld, Viliam Lisý, Jiří Čermák, and Michal Pěchouček. Double-oracle algorithm for computing an exact nash equilibrium in zero-sum extensive-form games. *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 2013.

[2] Peter I. Cowling, Edward J. Powley, and Daniel Whitehouse. Information set monte carlo tree search. *Computational Intelligence and AI in Games, IEEE Transactions on*, 4:120–143, 2012.

[3] Levente Kocsis, Csaba Szepesvári, and Jan Willemson. Improved Monte-Carlo Search. 2006.

[4] Daphne Koller, Nimrod Megiddo, and Berhard von Stengel. Fast algorithms for finding randomized strategies in game trees. *Proceedings of the 26th annual ACM symposium on Theory of computing*, 1994.

[5] David M. Kreps and Robert Wilson. Sequential equilibria. *Econometrica*, 1982.

[6] Harold W. Kuhn. Extensive games and the problem of information. *Annals of Mathematics Studies*, 1953.

[7] Richard D. McKelvey, Andrew M. McLennan, and Theodore L. Turocy. Gambit: Software tools for game theory. 2010.

[8] Richard D. McKelvey and Thomas R. Palfrey. Quantal response equilibria for normal form games. *Games and economic behavior*, 10(1):6–38, 1995.

[9] Peter Bro Miltersen and Troels Bjerre Sørensen. Computing a quasi-perfect equilibrium of a two-player game. *Economic Theory*, 2008.

[10] Peter Bro Miltersen and Troels Bjerre Sørensen. Fast algorithms for finding proper strategies in game trees. *In proceedings of 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 874–883, 2008.

[11] Roger B. Myerson. Refinements of the nash equilibrium concept. *Game Theory*, 7:73–80, 1978.

[12] John Nash. Non-cooperative Games. *Annals of Mathematics*, 1951.

[13] James Pita, Manish Jain, Janusz Marecki, Fernando Ordóñez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. Deployed armor protection: the application of a game theoretic model for security at the los angeles international airport. *In Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, 2008.

[14] Sheldon M. Ross. Goofspiel — the game of pure strategy. *Journal of Applied Probability*, 8(3):621–625, 1971.

[15] Tuomas Sandholm. The State of Solving Large Incomplete-Information Games, and Application to Poker . *AI Magazine*, pages 13–32, Winter 2010.

[16] Reinhard Selten. Reexamination of the perfectness concept for equilibrium points in extensive games. *International Journal of Game Theory*, 4:25–55, 1975.

[17] Mohammad Shafiei, Nathan Sturtevant, and Jonathan Schaeffer. Comparing uct versus cfr in simultaneous games. *Proceeding of the IJCAI Workshop on General Game-Playing*, 2009.

[18] Stephen J. J. Smith and Dana S. Nau. An analysis of forward pruning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1386–1386, 1995.

[19] Jason Tsai, Christopher Kiekintveld, Fernando Ordóñez, Milind Tamble, and Shyamsunder Rathi. Iris - a tool for strategic security allocation in transportation networks categories and subject descriptors. *In Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*, 2009.

[20] Theodore L. Turocy. Computing sequential equilibria using agent quantal response equilibria. *Economic Theory*, 42:255–269, 2010.

[21] Eric van Damme. A relation between perfect equilibria in extensive form games and proper equilibria in normal form games. *Game Theory*, 13:1–13, 1984.

[22] Eric van Damme. *Stability and Perfection of Nash Equilibria*. Springer-Verlag, 1991.

[23] Kevin Waugh, Nolan Bard, and Michael Bowling. Strategy Grafting in Extensive Games. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2026–2034, 2009.

[24] Michael P. Wellman. *Trading Agents*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Pub., 2011.

[25] Martin Zinkevich, Michael Bowling, and Neil Burch. A new algorithm for generating equilibria in massive zero-sum games. *In Proceednigs of the 22nd National Conference on Artificial Intelligence*, 2007.

[26] Jiří Čermák, Branislav Bošanský, and Viliam Lisý. Practical performance of refinements of nash equilibria in extensive-form zero-sum games. *European Conference on Artificial Intelligence*, in-press.