CZECH TECHNICAL
UNIVERSITY
IN PRAGUE

F3

Faculty of Electrical Engineering
Department of computer science

Master's Thesis

Reducing Variance in Monte Carlo Counterfactual Regret Minimization

Pavel Kuchař

Aug 2020

Supervisor: Mgr. Viliam Lisý, MSc., Ph.D.

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení:	**Kuchař**	Jméno: **Pavel**	Osobní číslo: **368849**
Fakulta/ústav:	**Fakulta elektrotechnická**		
Katedra/ústav:	**Katedra počítačů**		
Studijní program:	**Otevřená informatika**		
Studijní obor:	**Umělá inteligence**		

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Reducing Variance in Monte Carlo Counterfactual Regret Minimization

Název diplomové práce anglicky:

Reducing Variance in Monte Carlo Counterfactual Regret Minimization

Pokyny pro vypracování:

High variance has a strong negative impact on the speed of convergence of the family of MCCFR algorithms. There is wide literature on reducing variance of Monte Carlo estimators in general, as well as in sequential decision making. The goal of the student is to:
1) Review the existing variance reduction techniques usable for Monte Carlo estimators.
2) Review the techniques for variance reduction developed in computer poker research, namely generalized sampling for MCCFR and AIVAT.
3) Propose a novel variance reduction technique for MCCFR (e.g., based on AIVAT).
4) Experimentally evaluate the achieved variance reduction and the impact on the speed of MCCFR convergence and compare them to the state of the art.

Seznam doporučené literatury:

High variance has a strong negative impact on the speed of convergence of the family of MCCFR algorithms. There is wide literature on reducing variance of Monte Carlo estimators in general, as well as in sequential decision making. The goal of the student is to:
1) Review the existing variance reduction techniques usable for Monte Carlo estimators.
2) Review the techniques for variance reduction developed in computer poker research, namely generalized sampling for MCCFR and AIVAT.
3) Propose a novel variance reduction technique for MCCFR (e.g., based on AIVAT).
4) Experimentally evaluate the achieved variance reduction and the impact on the speed of MCCFR convergence and compare them to the state of the art.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Mgr. Viliam Lisý, MSc., Ph.D., centrum umělé inteligence FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **01.10.2018** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **20.09.2020**

[redacted]
Mgr. Viliam Lisý, MSc., Ph.D.
podpis vedoucí(ho) práce

[redacted]
podpis vedoucí(ho) ústavu/katedry
V z. doc. Ing. Jiří Hoštovec, Ph.D.

[redacted]
prof. Ing. Pavel Ripka, CSc.
podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

Datum převzetí zadání

Podpis studenta

Acknowledgement / Declaration

I would like to thank my supervisor Viliam Lisý for patience and helpful guidance. I would like to kindly thank my family and close friends for their patience, support and tolerance.

Author statement for undergraduate thesis:

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, 14 Aug 2020

.....

Abstrakt / Abstract

Práce se zaměřuje na redukci variance Monte Carlo CFR algoritmech v hrách s neúplnou informací. Po představení základního názvosloví a základních konceptů z teorie her jsou rozebrány nedávné úspěchy v redukci variance v MCCFR algoritmech jako je AIVAT, Generalized Sampling a VR-MCCFR. V rámci práce je navrhnut nový algoritmus pro redukci variance MCCFR za použití baselines a control variates založený na VR-MCCFR přístupu. Na konec je experimentálně ověřena funkčnost navrženého algoritmu.

Klíčová slova: Teorie her, Counterfactual regret, Monte Carlo Conterfactual Regret, MCCFR, CFR

This thesis focused on variance reduction of Monte Carlo CFR algorithm in game with incomplete information. After introduction basic game theoretic terminology and theoretical concepts there are analyzed recent achievement in variance reduction of MCCFR algorithms like AIVAT, Generalized Sampling and VR-MCCFR. Within this work is proposed novel algorithm for variance reduction with usage of baselines and control variates based on VR-MCCFR approach. Finally the functionality of proposed algorithm is experimentally verified.

Keywords: Game theory, Counterfactual regret, Monte Carlo Conterfactual Regret, MCCFR, CFR

/ Contents

Tables /

Chapter 1

Introduction

The **game** according to the Encyclopedia Brittanica is a "universal form of recreation generally including any activity engaged in for diversion or amusement and often establishing a situation that involves a contest or rivalry". For our purposes more accurate definition is provided by Oxford dictionary. A game is form of competitive activity or sport played according to rules.

The game theory is formal framework which makes possible to accurately represent games and subsequently define optimal strategy to play the game. The game is approached as decision making problem where participating players are deciding between multiple actions and are getting payoffs based on actions played. Also some sort of randomness can be involved in interaction between players and in the rules of the game. The randomness in the rules of the game can be modeled by adding one more player called nature with his own set of actions impacting the state of the game.

In this theses I will be focusing solely on non-cooperative games of two players. For such games John Forbes Nash Jr. defined Nash Equilibrium. The Nash equilibrium is defined as a set of strategies such that no player has the incentive to deviate from his strategy unilaterally, because unilateral deviation from strategy in the equilibrium will lead to less outcome for deviating player.

One of the hardest games to solve are games that involves "bluffing" phenomena. These games inherently contains dynamics, where some information is hidden to some of participants. First attempt to conceptually describe bluffing is ascribed to von Neumann in [1]. *" ... Not only do his minimax solutions of Poker variants prescribe Bluffing as a rational activity, rather than a psychic one, but they also specify, in general, definite probabilities with which Bluffing should be employed at each opportunity."* as is written in [2].

In game theory these games are called *imperfect information games* and were unsolvable for a long time. But recent progress enabled major breakthroughs. In 2000 was introduced new regret matching algorithm [3], which laid foundations for new self play counterfactual regret matching algorithm (CFR) [4]. Improved version CFR+ [5] taking advantage of counterfactual values was able to solve Two-Player Limit Texas Holdem Poker in 2015 [6].

Both CFR and CFR+ algorithms require to iteratively visit each possible state in the game to find Nash equilibrium. Whereas visiting each possible state and then storing particular optimal decision for each player in each state of the game is good way to approach some smaller games it is costly or unrealistic for bigger games. Also the CFR based algorithms are not able evaluate any strategy before full game is passed trough, so they are impractical for variety of applications.

Next big breakthrough was taken when in 2016 the Deepstack program as the first managed to beat professional human players at Two-Player No-Limit Texas Holdem [7]. Deepstact uses CFR reasoning recursively to handle information asymmetry but evaluates the explicit strategy on the fly rather than compute and store it prior to play. Deepstack is taking advantage of deep learning to learn estimator for the payoffs of the particular state of the game, which can be viewed as intuition - estimate of the value of holding particular cards in particular poker situation. Deepstack than uses this intuition to estimate payoffs beyond some time horizon to speed up reasoning about the game.

Method to find good strategy and do not have to traverse whole game can be achieved with use of Monte Carlo Tree Search techniques [8] in particular with combination of CFR reasoning [9]. This approach performs only some of possible gameplays each iteration and improve the strategy on the fly by iterative self play. Monte Carlo conterfactual regret minimization algorithm (MCCFR) made good results in various games [10].

Most recently big achievement was made also in field of multiplayer no-limit poker. The Pluribus, bot learned to play 6-Player No-Limit Texas Holdem poker, has beaten decisively professional human players in 2019 [11]. Even though Pluribus uses MCCFR based algorithms designed for two-player zero-sum games Pluribus applies it to multiplayer game. Nevertheless these algorithms guarantees successfully converge to optimal strategy only in Two-Player game, the Pluribus managed to empirically defeat human opponents and proved that the algorithms are able to compute good strategies in more general games than two-player zero-sum games.

1.1 Problem statement

High variance has strong negative impact on the speed of convergence of the family of MCCFR algorithms. This work focuses on reduction variance in family of Monte Carlo CFR algorithms using control variates.

We review CFR algorithm and Monte Carlo variants of CFR algorithm which can be considered as state of the art algorithms for solving imperfect information games. Then we will propose algorithm based on recent research in variance reduction of MCCFR algorithms. The algorithm will introduce baselines as control variates for conterfactual values. Finally the performance of proposed algorithm will be experimentally evaluated and confronted with Monte Carlo CFR algorithm as reference.

Chapter 2

Background

This chapter makes basic theoretical foundation for this work. First game theoretic framework setting boundaries of this work is defined, then related terminology is summed up and finally theoretical concepts used in current state of the art algorithms are introduced. Content respects common notation and is based on these works with related topics [12], [9] and [8].

2.1 Extensive form representation of sequential game

One of the most general representations of the game is called Extensive form game. It is often used to represent finite sequential games with imperfect information. It formalizes the game as a tree of nodes, where every node represents one game state and each edge represents action that leads from one state to another state. The state contains whole information about the game – e.g. concrete values of game properties, which player is on the move, actions that player can take, probability distribution of outcomes for every action. The nodes are grouped into Information sets, which contains game states indistinguishable for current player. Formally speaking, an extensive form game G is a tuple $\langle \mathcal{N}, \mathcal{A}, \mathcal{H}, \mathcal{Z}, \chi, \mathcal{P}, \varphi, \mathcal{I}, u_i \rangle$

- $\mathcal{N}$ finite set of players including nature player c
- $\mathcal{A}$ finite set of actions
- $\mathcal{H}$ finite set of states, each represents one possible history h - sequence of all actions performed by players to get from root to the particular state
- $\mathcal{Z}$ finite set of terminal states
- $\chi : \mathcal{H} \to \mathcal{A}$ function representing available actions in state
- $\mathcal{P} : \mathcal{H} \to \mathcal{N}$ player function determining for each non-terminal history a member of $calN$ who takes an action

- $\varphi : \mathcal{H} \times \mathcal{A} \to \Delta\{H \cup Z\}$ stochastic successor function transitioning game to another state
- $\mathcal{I}$ set of all information set classes $\mathcal{I}_i$. $\mathcal{I}_i$ is set of all information sets I of player i.

$$\mathcal{I} = \{h | h \in \mathcal{H}, \forall h' \in I : \mathcal{P}(h) = \mathcal{P}(h') = i, \chi(h) = \chi(h')\}$$

- $u : \{\mathcal{N} \setminus c\} \times \mathcal{Z} \to \mathbb{R}$ utility function, $u_i(z)$ denotes utility of player i in terminal state $z \in \mathcal{Z}$

In this work the h can be used instead of $\mathcal{I}$ meaning the information set that contains state $h \in \mathcal{H}$. Also following notation is used for simplicity.

- $A(I)$ denotes set of actions available in information set $I \in \mathcal{I}$
- $h \cdot a$ or ha is state h' which is achieved after legal action a was taken in state h
- $h' \sqsubseteq h$ means h' is a prefix sequence or equals to h
- by player $-i$ is meant the second player other than i

2.2 Strategy

Pure strategy $\sigma_i : I \to A(I)$ of player i is a function that maps each information set I to a single action admissible in that information set. If a player plays according to the pure strategy he always chooses action mapped to particular information set.

Mixed strategy or generally strategy without specification we call similar function $\sigma_i : I \to \Delta A(I)$ that maps each information set I to a probability distribution over available actions $\Delta A(I)$ rather than to single action. Player playing according the strategy may chooses any action with non-zero probability, in repeated game the player behaves according to the probability distribution.

A strategy profile profile $\sigma = [\sigma_1, ..., \sigma_{|N|}]$ is vector of strategies, one for each player.

$\delta(a, I)$ denotes probability of playing action a in information set I.

2.3 Expected utility

The probability of reaching particular history h can be expressed as

$$\pi^{\sigma}(h) = \prod_{h'a \sqsubseteq h} \delta(h', a)$$

Expected utility is the expected payoff of the player i playing according to the strategy σ_i whereas opponent plays according to the strategy σ_{-i}. Expected

utility is formally defined as the

$$u_i(\sigma_i, \sigma_{-i}) = \sum_{s \in \mathcal{S}, s' \sqsubset S_{ii}} \pi^{\sigma}_{-i}(h)\pi^{\sigma}(z|h)u_p(z)$$

2.4 Best response

Best response BR_i of player i to a strategy profile σ is a strategy that maximizes expected value of player i if the other players play according to σ. Best response can be also used as the name of this expected utility value.

2.5 Nash equilibrium

A Nash equilibrium is a strategy profile such that no player has the incentive to deviate from his strategy unilaterally. Formally strategy profile $\sigma = [\sigma_1, ..., \sigma_{|N|}]$ is a Nash equilibrium if $\forall i \in \mathcal{N}, s_i \in BR_i(s_{-i})$.

2.6 Counterfactual value

Counterfactual value is expected value that does not consider the player's own probability of reaching the state. Counterfactual value for player i in state h is

$$v^{\sigma}_i(I) = \sum_{z \in \mathcal{Z}, h \sqsubset z} \pi^{\sigma}_{-i}(h)\pi^{\sigma}(z|h)u_p(z)$$

Counterfactual regret of not taking action a at information set represented with history h is

$$r(I, a) = v_i(\sigma, ha) - v_i(\sigma, h)$$

2.7 Control variate

Control variate is a random variable Y with known mean μ_Y and positive correlation with random variable X which we want to estimate. By sampling (X, Y) and introduction new random variable $Z_i = X_i + c(Y_i - \mu_Y)$ we can use an estimator of Z in place of estimator of X.

Because mean value of Z is same as mean value of X the variance can be written as $\mathbf{Var}(Z_i) = \mathbf{Var}(X_i) + c^2\mathbf{Var}(Y_i) + 2c\mathbf{Cov}(X_i, Y_i)$ and effectively be reduced with this method when $\mathbf{Cov}(X, Y) > \frac{c^2}{2}\mathbf{Var}(Y)$.

2.8 Regret Matching

Regret matching is algorithm where new strategy distribution σ in particular information set I is obtained by normalizing the positive portions of regret vectors r or using uniform distribution if all regrets are non-positive.

$$R^+_{sum} = \sum_{a \in A(I)} \max(r_I(a), 0)$$

for $\forall a \in A(I)$ **do**

  if $R^+_{sum} > 0$ **then** $\sigma(I,a) = \frac{\max(r_I(a),0)}{R^+_{sum}}$

  else $\sigma(I,a) = \frac{1}{|A|}$

end for

Figure 2.1. Regret matching algorithm

2.9 CFR algorithm

Counterfactual regret minimization introduced in [4] is self-play algorithm that uses regret matching to find Nash equilibrium in two player zero sum games.

The algorithm starts with uniform strategy profile σ^0 and iteratively passes by depth-first search trough tree of extensive form game. In each iteration the regret matching is used to compute strategy profile σ^t from cumulative counterfactual regrets of all previous iterations. The σ^t is then played and cumulative regret of every action in each information set is updated using new counterfactual values.

$$r_I[a] = r_I[a] + \pi_{-i} \cdot (v_{\sigma_{I \to a}}[a] - v_\sigma)$$

Average strategy profile $\bar{\sigma}^t_i(a|I)$ of player i is accumulated from immediate strategy profile σ^t every iteration.

$$\bar{\sigma}^t_i(a|I) = \frac{\sum_{t=1}^{T} \pi_i^{\sigma^t}(I)\sigma^t(a|I)}{\sum_{t=1}^{T} \pi_i^{\sigma^t}(I)}$$

The accumulated average strategy converges to the optimal strategy $s_I(a)$.

Initialize regret tables: $\forall I, \forall a \in A(I)$: $r_I(a) = 0$
Initialize cumulative strategy tables: $\forall I, \forall a \in A(I)$: $s_I(a) = 0$
Initialize initial profile: $\sigma^1(I,a) = \frac{1}{|A|}$

function CFR(h, i, t, π_1, π_2):
if h is terminal **then return** $u_i(h)$
I is information set containing h
$R^+_{sum} = \sum_{a \in A(I)} \max(r_I(a), 0)$
for $a \in A(I)$ **do**
    if $R^+_{sum} > 0$ **then** $\sigma^t(I,A) = \frac{\max(r_I(a),0)}{R^+_{sum}}$
    else $\sigma^t(I,A) = \frac{1}{|A|}$
end for
$v_\sigma = 0$
$\forall a \in A(I)$: $v_{\sigma_{I\to a}}(a) = 0$
for $a \in A(I)$ **do**
    for $\pi, h' \in h \cdot a$ **do**
        if $\rho(h) = 1$ **then** $v_{\sigma_{I\to a}}(a) = v_{\sigma_{I\to a}}(a) + CFR(h', i, t, \sigma^t(I,a) \cdot \pi_1, \pi \cdot \pi_2)$
        if $\rho(h) = 2$ **then** $v_{\sigma_{I\to a}}(a) = v_{\sigma_{I\to a}}(a) + CFR(h', i, t, \pi \cdot \pi_1, \sigma^t(I,a) \cdot \pi_2)$
    end for
    $v_\sigma = v_\sigma + \sigma^t(I,a) \cdot v_{\sigma_{I\to a}}(a)$
end for
if $\rho(h) = 1$ **and** $i = 1$
    for $a \in A(I)$ **do**
        $r_I(a) = r_I(a) + \pi_2 \cdot \left(v_{\sigma_{I\to a}}(a) - v_\sigma\right)$
        $s_I(a) = s_I(a) + \pi_1 \cdot \sigma^t(I,a)$
    end for
end if
return v_σ

function Solve():
for $t \in \{1, 2, \ldots, T\}$ **do**
    for $i \in \{1,2\}$ **do**
        CFR(*root*, i, t, 1, 1)
    end for
end for

Figure 2.2. Vanilla CFR algorithm

2.10 MCCFR algorithm

For large games full traversal of the game tree with CFR may be very expensive. This problem solves Monte Carlo CFR, family of algorithms that rather than visiting entire tree to obtain counterfactual values samples only portion of the tree at each iteration. In [9] it is shown that estimated counterfactual regrets computed from sampled subtree approaches true values with high probability if "care is taken in how to sample these subtrees".

MCCFR algorithms can incorporate various sampling schemata. For example outcome sampling [9] traverses only one terminal history and regrets along are updated. High level view of MCCFR algorithm is listed below.

Initialize regret tables: $\forall I, \forall a \in A(I): r_I(a) = 0$
Initialize cumulative strategy tables: $\forall I, \forall a \in A(I): s_I(a) = 0$
Initialize initial profile: $\sigma^1(I,a) = \frac{1}{|A|}$
for $t \in \{1, 2, \dots, T\}$ **do**
  for $i \in \{1,2\}$ **do**
    Sample block of terminal histories Q using S
    for each prefix z[I] of each terminal history $z \in Q$ with $\rho(z[I]) = i$ **do**
      $R^+_{sum} = \sum_{a \in A(I)} \max(r_I(a), 0)$
      for $a \in A(I)$ **do**
        if $R^+_{sum} > 0$ **then** $\sigma^t(I,A) = \frac{\max(r_I(a),0)}{R^+_{sum}}$
        else $\sigma^t(I,A) = \frac{1}{|A|}$
        $\tilde{r} = \tilde{r}(I,a)$ the sampled regret for not taking a
        $r_i[a] = r_i[a] + \tilde{r}$
        $s_I[a] = s_I[a] + AverageStrategyIncrement(s_I, t, \sigma_i, I, a)$
      end for

Figure 2.3. High level view of MCCFR algorithm

2.11 Variance in MCCFR

Generally variance measures the spread of values around average value of measurement. Variance $var(X)$ of random variable X with mean μ_X is defined as

$$var(X) = E[(X - \mu_X)^2]$$

The variance is consequence of sampling game tree with Monte Carlo algorithms. The effect of the variance was thoroughly examined in [13]. When action with large value is not sampled a lot of variance is introduced in the MCCFR, because every action that is not sampled is assumed to provide zero counterfactual value to the strategy. Similarly the less the values are sampled in particular node the more sampled counterfactual regret is inaccurate.

In [13] was presented general upper bound on the average regret $R_i^T = \max_{\sigma' \in \Sigma_i} \sum_{t=1}^{T} (u_i(\sigma', \sigma_{-i}^t) - u_i(\sigma_i^t, \sigma_{-i}^t))$

$$\frac{R_i^T}{T} \leq \left(\hat{\Delta}_i + \frac{\sqrt{\mathbf{Var}}}{\sqrt{p}} \right) \frac{|\mathcal{I}|\sqrt{A_i}}{\sqrt{T}}$$

$$\mathbf{Var} = \max_{t \in T, I \in \mathcal{I}_i, a \in A(I)} \mathbf{Var}[r_i^t(I, a) - \hat{r}_i^t(I, a)]$$

This upper bound relates the variance of the estimator directly with average regret and therefore there is direct relation between variance of the estimator and convergence rate of the MCCFR algorithm.

Chapter 3

Variance reduction in Monte Carlo

Variance reduction in Monte Carlo estimators has been approached in various angles. For example numerical integration is one field where applications of Monte Carlo estimators leads to great results [14].

From the random nature of Monte Carlo methods some variance of random samples is always present and unfortunately has significant impact on the final result of the integration. There are three main directions to reduce the variance in field of Monte Carlo Integration [15]

- partial integration - lowers variance by replacing parts of integrals by their averages
- systematic sampling - multiple sampling scheme e.g. stratified sampling [14]
- control variates

In game theory perspective various types of systematic sampling found usage. Notably importance sampling [16] and its extensions [9]. Recently also variance reduction were introduced in [17] and [18].

3.1 AIVAT

Action-informed value assessment tool (AIVAT) [17] is low variance unbiased estimator that uses arbitrary heuristic estimate of state value and explicit strategy of a subset of the agents. AIVAT is an extension of previous MIVAT [19] which used control variates with heuristic value estimates to reduce the variance caused by chance events. AIVAT combines MIVAT with importance sampling over imaginary observations [16] into one estimator. The AIVAT was designed for computer poker research where was able to reduce number of hands needed to statistically significant conclusions in no-limit poker by factor of 10.

3.2 Generalized sampling

In [13] based on the analysis of relation between variance of the estimator and convergence rate was introduced bounded and unbiased estimator with empirically

lower variance than MCCFR. The algorithm extends MCCFR with the estimator which replaces counterfactual values of non-sampled nodes with estimates of true counterfactual values. The estimator probes with single Monte Carlo roll-out according to current strategy to get unbiased estimate of counterfactual value of non-sampled action. The algorithm shows 10%-30% improvement over MCCFR in dependence on the domain.

3.3 VR-MCCFR

During work on this thesis new article was published [18], which thoroughly describes usage of control variates in combination with outcome sampling [9]. The VR-MCCFR served as foundation for new MCCFR algorithm with baselines.

The article [18] inspect analogies between MCCFR algorithms and reiforcement learning. Viewing imperfect information game as partially-observable MDP without any cycles enables redefines MCCFR as off-policy Monte Carlo analog.

Because incorporating baselines in reinforcement learning gradient methods is successful the VR-MCCFR article introduced the idea to add baselines to the MCCFR algorithm as counterfactual values estimate. This technique leads to shrink variance and speeding up the convergence of MCCFR by an order of magnitude.

3.4 MCCFRb

Monte Carlo counterfactual regret matching algorithm with baselines (MCCFRb) enhances standard MCCFR algorithm described in previous chapter with baselines which serves as control variates. As baselines we use iteratively updated values that converges to the expected value of playing particular action in an information set by player i.

Initialize regret tables: $\forall I, \forall a \in A(I)$: $r_I(a) = 0$
Initialize cumulative strategy tables: $\forall I, \forall a \in A(I)$: $s_I(a) = 0$
Initialize baselines: $\forall i, \forall I, \forall a \in A_i(I)$: $b_i(I, a) = 0$
Initialize visit counters: $\forall i, \forall I$: $k_i(I) = 0$
Initialize information set counters: $\forall I$: $c_I = 0$
Initialize initial profile: $\sigma^1(I, a) = \frac{1}{|A|}$

function MCCFRb($h, i, t, \pi_i, \pi_{-i}, s$):
if h is terminal **then return** $u_i(h)$
if h is chance **then** sample $\hat{a}$ and **return** MCCFRb($h\hat{a}, i, t, \pi_i, \pi_{-i}, s$)

I is information set containing h
$R^+_{sum} = \sum_{a \in A(I)} \max(r_I(a), 0)$
for $\forall a \in A(I)$ **do**
    if $R^+_{sum} > 0$ **then** $\sigma^t(I, a) = \frac{\max(r_I(a), 0)}{R^+_{sum}}$
    else $\sigma^t(I, a) = \frac{1}{|A|}$
end for
if $P(h) = i$:
    sample action $\hat{a} \in A(I)$ with probability $\xi(h, a)$ with ϵ on-policy $\sigma^t(I)$ sampling
else
    sample action $\hat{a} \in A(I)$ with probability $\xi(h, a)$ with on-policy $\sigma^t(I)$ sampling
end if
$u =$ MCCFRb($h\hat{a}, i, t, \pi_i \cdot \sigma^t(I, a), \pi_{-i}, s \cdot \xi(h, a)$)
$\hat{u} = \sigma^t(I, \hat{a}) \left(b_i(I, \hat{a}) + \frac{u - b_i(I, \hat{a})}{\xi(h, a)} \right) + \sum_{a \in \{A(I) \setminus \hat{a}\}} \sigma^t(I, a) b_i(I, a)$
if $P(h) = i$:
    $r_I(a) = r_I(a) + \frac{\pi_{-i}}{s} \left(\frac{u - b_i(I, \hat{a})}{\xi(h, a)} - \hat{u} \right) + \sum_{a \in \{A(I) \setminus \hat{a}\}} \frac{\pi_{-i}}{s} (b_i(I, a) - \hat{u})$
else
    for $\forall a \in A(I)$ **do**
        $s_I(a) = s_I(a) + (t - c_I)\pi_{-i}\sigma^t(I, a)$
        $c_I = t$
    end for
end if
$b_i(I, \hat{a}) = \frac{b_i(I, \hat{a}) \cdot k_i(I) + b_i(I, \hat{a}) + \frac{u - b_i(I, \hat{a})}{\xi(h, a)}}{k_i(I) + 1}$
$k_i(I) = k_i(I) + 1$

function Solve():
for $t \in \{1, 2, \ldots, T\}$ **do**
    for $i \in \{1,2\}$ **do**
        MCCFRb(*root*, i, t, 1, 1, 1)
    end for
end for

Figure 3.1. Monte Carlo counterfactual regret matching with baselines (MCCFRb)

The algorithm uses outcome sampling, optimistic averaging for computing optimal strategy [9] and ϵ on-policy exploration. In bellow listed algorithm direct average is used rather than in [18] proposed exponentially-decaying average for the baseline updates.

On the beginning the algorithm initiates all structures with reasonable values:

- regret tables - initiate immediate counterfactual regret of playing any action for any information set to 0
- cummulative strategy tables - initiate accumulated probability of playing any action in any information set to 0
- baselines - initiate value of baseline for any action in any information set for both players to 0
- visit counters - initiate value of counters tracking how many times an information set was visited to 0
- information set counters - for each information set initiate counter tracking last iteration in which particular information set was visited and cumulative strategy was updated
- initial strategy profile - initiate starting immediate strategy profile to uniform strategy

In each of the iterations t one terminal history is ϵ on-policy sampled for each player according to the immediate strategy σ^t of the players. During sampling immediate strategy is computed on the fly with regret matching algorithm normalizing positive regrets or using uniform distribution if all regrets are non-positive.

The sampled history is backtracked propagating counterfactual value u of the child node to the parent. During backtracking the new variable $\hat{u}$ is introduced as baseline-enhanced version of counterfactual value u and is used to accumulate immediate regret $r_I(a)$ with current new regret value in nodes belonging to the player under simulation. If the node belongs to the opponent cumulative strategy s_I is accumulated with immediate strategy weighted with iteration-information set counter difference and current iteration number is assigned to the information set counter c_I. Finally baseline value of the sampled action $\hat{a}$ is updated as average over all baseline values of the action $\hat{a}$ in current information set visited so far.

At the end whole cumulative strategy profile should be updated one more time to accumulate each strategy value to the same level according to the last iteration number. Final strategy profile can be obtained as normalization of the cumulative strategy tables. This approach of computing average strategy is called optimistic averaging [9].

Chapter 4

Implementation

MCCFR and MCCFRb algorithms were implemented as part of GTlib2, Game Theoretic library written in C++ 14. The library is developed internally by AI Center in Department of Computer Science at Faculty of Electrical engineering CTU in Prague. The implementation of the algorithms can be found in `\algorithms\mccfr.cpp` and `\algorithms\mccfr.h` files.

The library is successor to the free Java written GTlib available on GitHub [20] and aims to provide framework to evaluate algorithms operating over extensive form games. GTlib2 contains various domains used commonly in game theoretic articles and good support for experiment evaluation.

GTlib2 supports development of domain-independent algorithms for solving the extensive form games. The library provides implementation of game tree with information sets, basic operations over the game tree and information sets, data structures for strategy representation. GTlib2 also contains support for modelling extensive form games and various domains for evaluation of algorithms.

4.1 Game tree representation and data structures

GTlib2 contains extensively inherited collection of classes for representation of all aspects of game trees and strategy. This collection serves as interface for domain-independent work with the game tree.

New structures were implemented to store MCCFRb algorithm dependent information, namely `RegretTable`, `CummulativeStrategyTable`, `Baselines` and `BaselineVisits`. Structures used in implementation of MCCFR and MCCFRb are listed in the table 4.1.

Structure	Description
`Action`	class representing an action
`AOH`	class representing action-observation history describing information set
`Baselines`	structure pairing each permutation of `InformationSet` and `Action` with decimal number to represent baseline value
`BaselineVisits`	structure pairing each `InformationSet` with integer to represent number of visits of the `InformationSet`
`BehavioralStrategy`	structure pairing each permutation of `InformationSet` and `Action` with decimal number to represent strategy
`CummulativeStrategyTable`	structure pairing each permutation of `InformationSet` and `Action` with decimal number to represent cummulative strategy
`Domain`	class representing general domain supporting domain independent extensive form game tree traversal
`EFGNode`	class representing a node in extensive game tree
`InformationSet`	class representing an information set
`RegretTable`	structure pairing each permutation of `InformationSet` and `Action` with decimal number to represent conterfactual regret

Table 4.1. GTlib2 structures used in implementation of MCCFR and MCCFRb algorithms.

4.2 Domains

GTlib2 provides various domains for evaluating algorithms. Domain classes are responsible for extensive form game tree generation. They provide methods to obtain available actions for any node in the tree, methods for traversing the tree by applying actions and prescribes player's outcomes in terminal nodes.

Extensive form game approach of modelling the game can be applied on any game domain and supports uniform interface which makes easy to design domain-independent algorithms. GTlib2 fully utilizes this advantage.

MCCFR and MCCFRb algorithms implemented in this work were evaluated primarily on the `IIGoofSpielDomain` which represents Goofspiel game with given number of cards and seed value. The order in whitch hidden cards are played by the nature is dependent on the seed. Same seed always generates same card order. Besides evaluation on Goofspiel domain also matching pennies and other more trivial domains were tested.

4.3 Algorithms

Algorithms provided in GTlib2 apart of finding optimal strategy also supports experiment evaluation and debugging. MCCFR and MCCFRb algorithms were added to the `algorithms` namespace as one of the options to solve any domain provided by the library. Other algorithms used for experiments in this work are listed below.

- `bestResponseTo` - algorithm which takes strategy for some given domain and computes best response strategy to the given strategy and expected value of playing best response strategy against given strategy.
- `treeWalk` - algorithm which takes function traverses whole extensive game tree of given domain and calls given function on every node.

4.4 Experimets

Evaluation of the algorithms requires invasive intervention to the code of the function introducing inexcusable time complexity. Therefore C++ macros are used to enable or disable parts of the code required in particular build of the project. The defines switches over modes of execution, enables debugging options or configurates hyper-parameters. The defines can be found in the very beginning of the `mccfr.h` header file.

Chapter 5

Evaluation and Experimental results

The MCCFR with outcome sampling and control variates was evaluated on multiple instances of goofspiel game and compared to the MCCFR with outcome sampling without use of control variates as reference state of the art algorithm.

To measure quality of the algorithm the ”optimal” strategy of current iteration must be computed from cumulative strategies for both players and then based on this strategy exploitability can be computed as

$$BR_1(\sigma) + BR_2(\sigma)$$

The $BR_i(\sigma)$ is expected reward of the opponent playing against player i best response to his strategy σ.

Tracked parameter is convergence rate of the exploitability in dependence on number of iterations ran so far. The quality of convergence of various algorithms can be easily confronted on the exploitability-iteration charts.

5.1 Goofspiel

Goofspiel(N) is two player card game which is played with three sets of cards with values from 1 to N. Every player has one private set of bid cards. The last set of cards is randomly shuffled and placed face up on the pile.

Players then simultaneously bid on the topmost card on the pile by discarding exactly one bid card. It is crucial to reveal bid card simultaneously to prevent information leak to other player. The player who bids a card with higher value takes the top card of the pile as reward. If the bids are same the reward card is discarded and the game continues with new bidding until there are no face-up cards on the pile and players have no bid cards.

When the bidding is ended players tot up gathered reward cards and the player with higher score wins with utility equals the difference in the scores. Whereas the player with lower score loses with negative utility equals to the difference in the scores.

Goofspiel(N) is zero sum imperfect information game with generally multiple mixed Nash equilibria. For $N > 3$ optimal strategies are always mixed because every pure strategy has pure counter-strategy, which guarantees win. Trivially to construct counter-strategy bid the lowest card when opponent bids his highest card and then each other turn bid one point higher card than opponent.

5.2 Leduc Hold'em

Poker games generally are good examples of zero-sum imperfect information games. There are hundreds of poker variants some even played on professional level for a lot of money and some not played at all but designed solely for purpose of game theoretic research (e.i. Khun poker, Leduc Hold'em).

Leduc Hold'em is simplified poker variant for two players. The game is played with six cards deck containing Js, Jh, Qs, Qh, Ks and Kh and one stack of chips belonging to each player. At the beginning one random card that is dealt to each player and each player put 1 chip to the pot. Then first betting round takes place and afterwards one community card is dealt face up. There is another betting round and if both players are still in the game after second betting round, they reveal their private cards. If one player's private card is the same rank as the board card, he wins the game otherwise the player with higher rank private card wins.

This poker variant was first introduced in [21] and it's corresponding game tree contains 61336 different states.

5.3 Experimental results

To compare convergence rate of proposed MCCFRb algorithm with directly averaged baselines the exploitability of both MCCFR (no baseline updates) and MCCFRb was computed for 20 different games of Goofspiel(4), Goofspiel(5) and Leduc Holdem. Different games were instantiated by different random seeds (obtained via `random.org` service [22]).

The exploitability was computed every order of magnitude number iteration - that means on the first, 10th, 1000th, ..., 1 000 000th iteration. Standard deviation and confidence intervals were computed with NumPy nad SciPy python library [23] and always plotted with logaritmic iteration axis.

5.3.1 Goofspiel(4) Direct average, Exponentially decaying average, No Updates

The comparison of exploitability MCCFRb with baselines updated via direct and exponentially decaying average with MCCFR (no updates) algorithm on goofspiel(4) game computed every one order of magnitude of the number of iterations to inspect convergence rate on far horizon is depicted below. Experimental results with computed mean and standard deviation from 20 runs on different games are contained in following table.

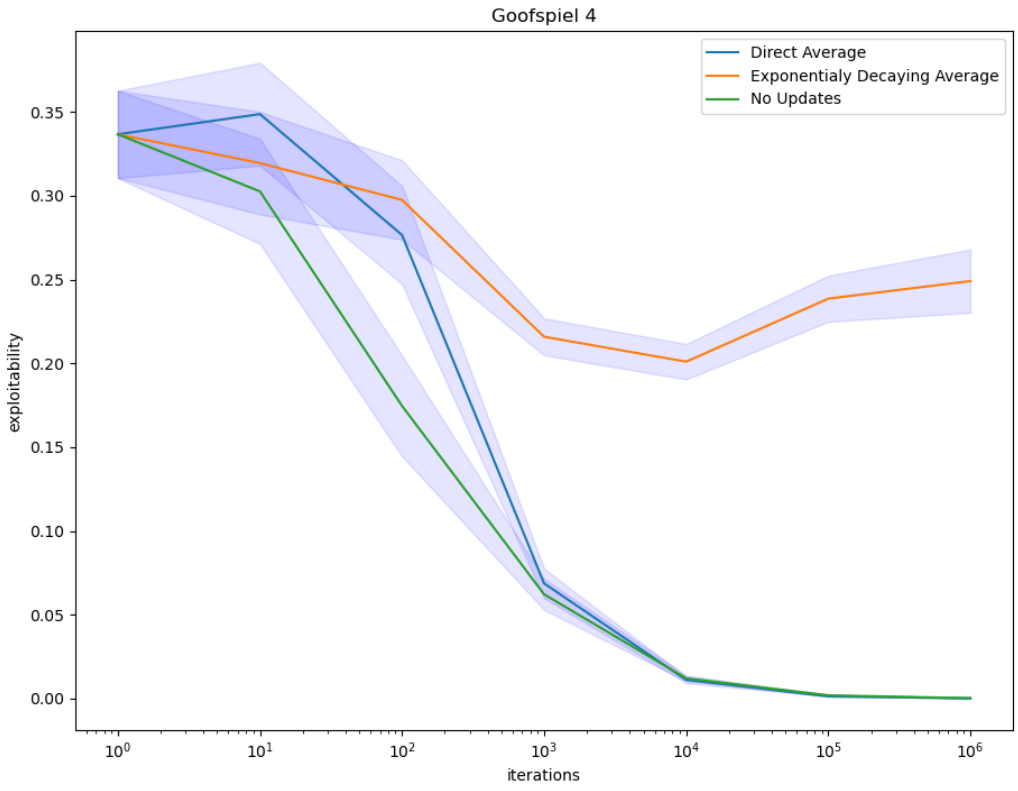

Figure 5.1. Exploitability comparison of MCCFR (No Updates) with outcome sampling and MCCFRb with outcome sampling and direct and exponentially decaying baseline averaging with confidence interval over 20 different Goofspiel(4) games.

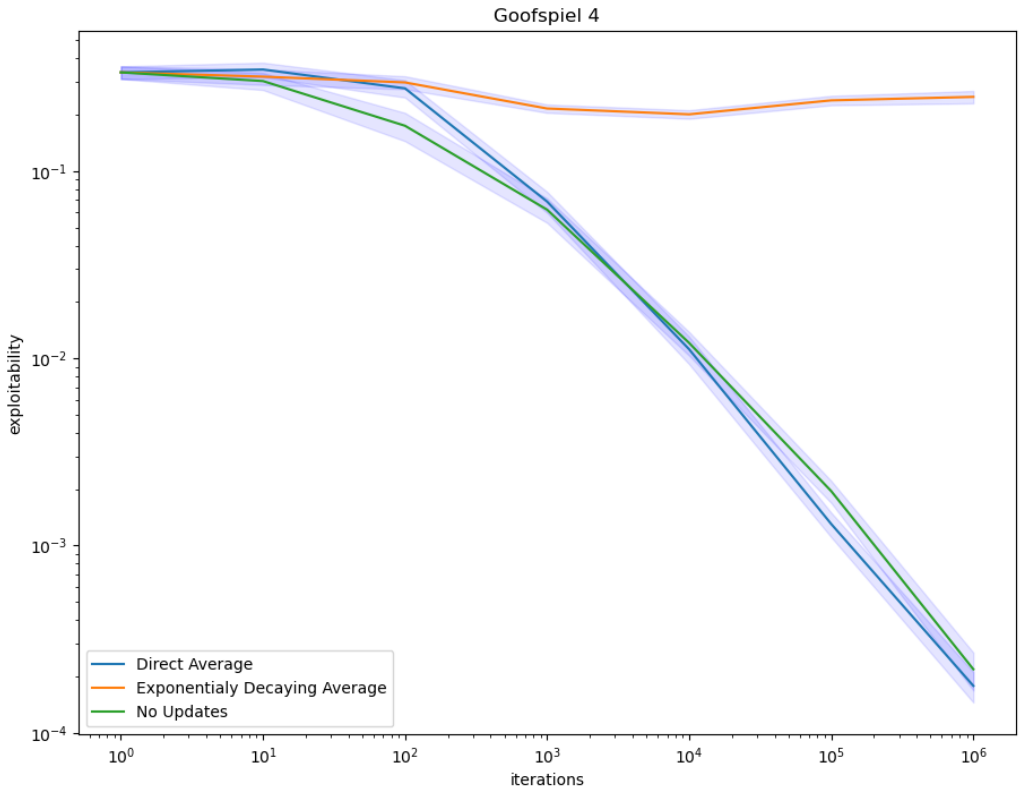

Figure 5.2. Logaritmically scaled exploitability comparison of MCCFR (No Updates) with outcome sampling and MCCFRb with outcome sampling and direct and exponentially decaying baseline averaging with confidence interval over 20 different Goofspiel(4) games.

Variant	Iteration	Mean	Standard deviation
`Direct average`	1	0.34	0.06
	10	0.35	0.07
	100	0.28	0.07
	1000	0.07	0.02
	10000	0.011	0.004
	100000	0.0013	0.0004
	1000000	0.00018	0.00007
`Exponentially decaying average`	1	0.34	0.06
	10	0.32	0.07
	100	0.30	0.05
	1000	0.22	0.03
	10000	0.20	0.02
	100000	0.24	0.03
	1000000	0.25	0.04
`No updates`	1	0.34	0.06
	10	0.30	0.07
	100	0.17	0.07
	1000	0.06	0.02
	10000	0.012	0.004
	100000	0.0020	0.0006
	1000000	0.00022	0.00011

Table 5.1. Goofspiel(4) experimental results from 20 runs on different games.

5.3.2 Goofspiel(5) Direct average, No Updates

The comparison of exploitability MCCFRb with baselines updated via direct average with MCCFR (no updates) algorithm on goofspiel(5) game computed every one order of magnitude of the number of iterations is depicted below. Experimental results with computed mean and standard deviation from 20 runs on different games are contained in following table.

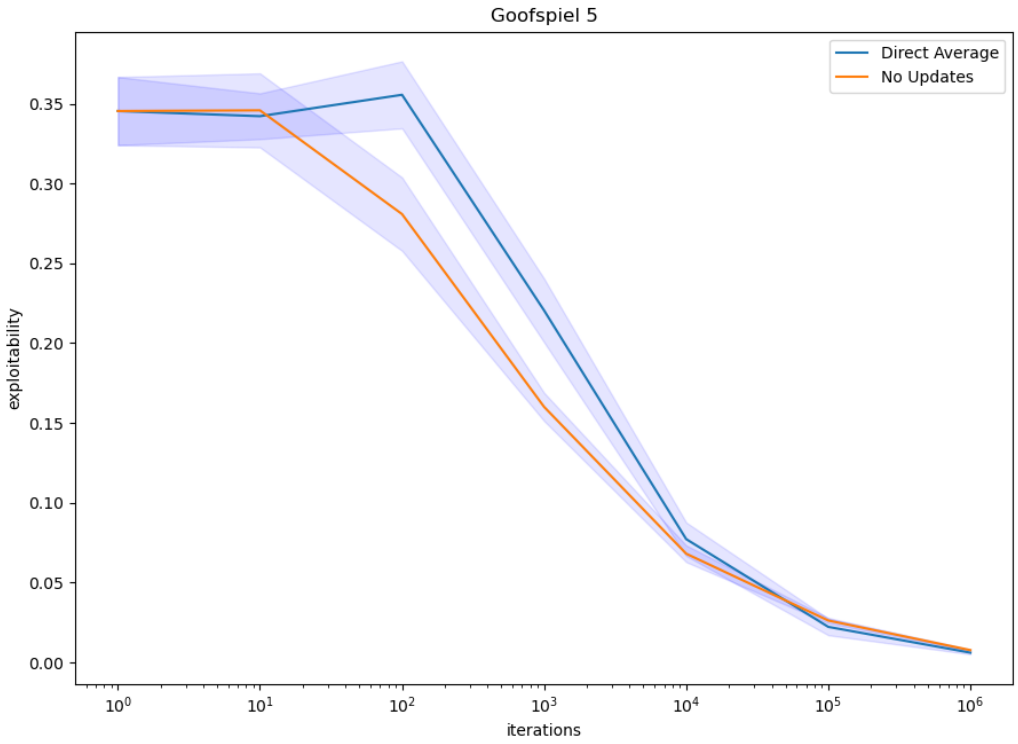

Figure 5.3. Exploitability comparison of MCCFR (No Updates) with outcome sampling and MCCFRb with outcome sampling and direct and exponentially decaying baseline averaging with confidence interval over 20 different Goofspiel(5) games.

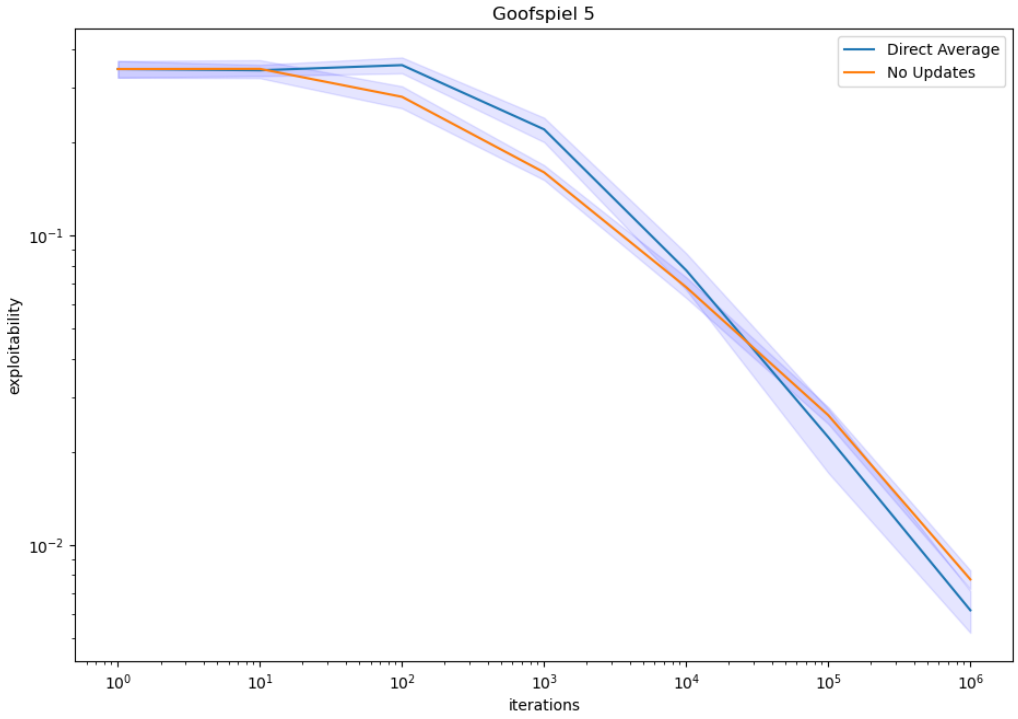

Figure 5.4. Logaritmically scaled exploitability comparison of MCCFR (No Updates) with outcome sampling and MCCFRb with outcome sampling and direct and exponentially decaying baseline averaging with confidence interval over 20 different Goofspiel(5) games.

Variant	Iteration	Mean	Standard deviation
`Direct average`	1	0.35	0.05
	10	0.34	0.03
	100	0.36	0.05
	1000	0.22	0.05
	10000	0.08	0.02
	100000	0.02	0.01
	1000000	0.006	0.002
`No updates`	1	0.35	0.05
	10	0.35	0.05
	100	0.28	0.05
	1000	0.16	0.02
	10000	0.07	0.01
	100000	0.026	0.004
	1000000	0.008	0.001

Table 5.2. Goofspiel(5) experimental results from 20 runs on different games.

5.3.3 Leduc Holdem Direct average, No Updates

The comparison of exploitability MCCFRb with baselines updated via direct average with MCCFR (no updates) algorithm on Leduc Holdem domain computed every one order of magnitude of the number of iterations is depicted below. Experimental results with computed mean and standard deviation from 20 runs on different games are contained in following table.

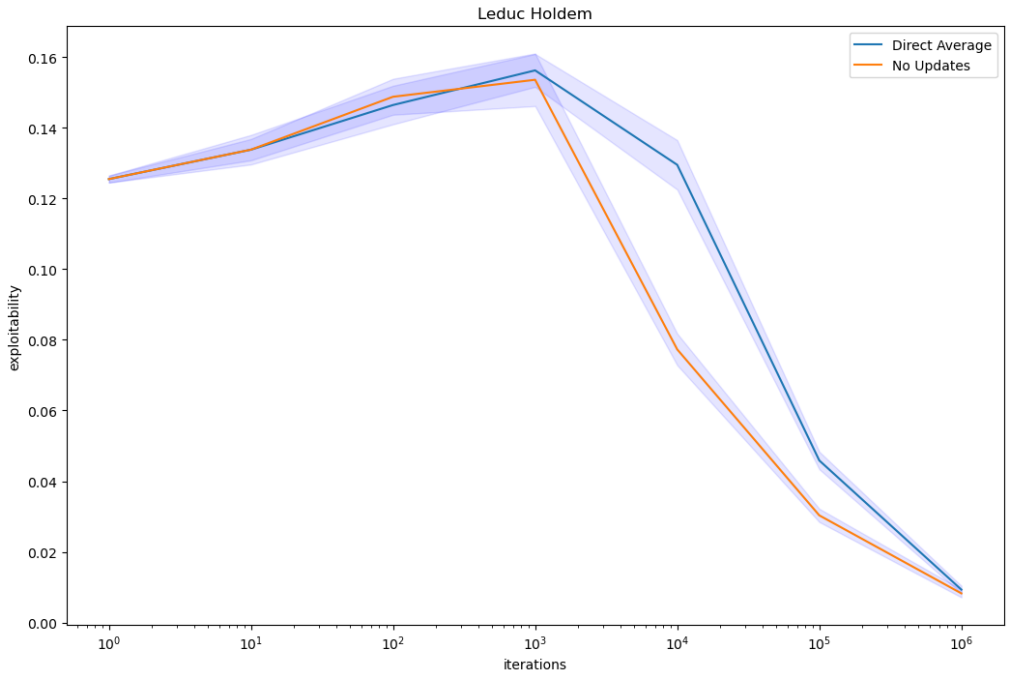

Figure 5.5. Exploitability comparison of MCCFR (No Updates) with outcome sampling and MCCFRb with outcome sampling and direct and exponentially decaying baseline averaging with confidence interval over 20 different Leduc Holdem games.

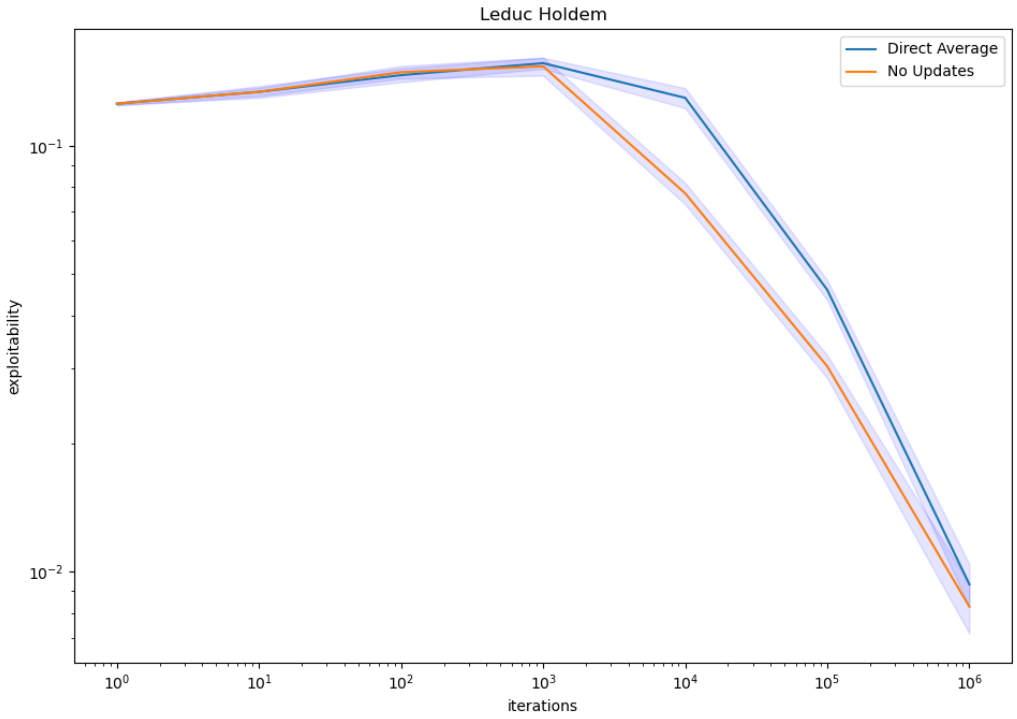

Figure 5.6. Logaritmically scaled exploitability comparison of MCCFR (No Updates) with outcome sampling and MCCFRb with outcome sampling and direct and exponentially decaying baseline averaging with confidence interval over 20 different Leduc Holdem games.

Variant	Iteration	Mean	Standard deviation
`Direct average`	1	0.125	0.002
	10	0.13	0.01
	100	0.14	0.01
	1000	0.16	0.01
	10000	0.13	0.02
	100000	0.046	0.006
	1000000	0.009	0.002
`No updates`	1	0.125	0.002
	10	0.134	0.007
	100	0.14	0.01
	1000	0.16	0.02
	10000	0.08	0.01
	100000	0.030	0.004
	1000000	0.008	0.002

Table 5.3. Leduc Holdem experimental results from 20 runs on different games.

Chapter 6

Discussion and Conclusion

The main goal of this theses was to introduce Monte Carlo counterfactual regret minimization algorithm using baselines as control variates for playing general two-player zero-sum games and achieve variance reduction over Monte Carlo counterfactual regret minimization algorithm without baselines.

The proposed MCCFR algorithm with baselines was evaluated on multiple instances of game and achieved variance reduction over reference MCCFR algorithm with outcome sampling without baselines. MCCFRb with with directly averaged baselines takes roughly half time (in sense of iterations) to achieve similar exploitability as MCCFR in most significant iterations when the measured exploitability is radically reducing with each iteration.

From experiments it is clear that in the beginning of computation when the number of iterations is low and the strategy is not realiably converging in sense of exploitability the MCCFRb is not significantly worse than MCCFR. When the exploitability is minimal the number of iterations to reduce it is quickly ramping up and the MCCFRb advantage over MCCFR is almost mitigated due to the magnitude of exploitability differences.

MCCFRb with exponentially decaying average was inferior to the MCCFRb with direct average and even to the MCCFR.

Fresh new article [24] covering MCCFR algorithms with baselines was published right before submission of the theses and therefore have to be mentioned here. The article is introducing a general framework for baseline construction in extensive form games. In scope of this article our proposed algorithm utilizes learned infoset baseline function. Experimental results presented in the article [24] correspond with the results presented in this theses.

References

[1] J. Von Neumann, and O. Morgenstern. *Theory of games and economic behavior*. Princeton University Press, 1953.

[2] A. W. Tucker H. W. Kunh. John von Neumann's work in the theory of games and mathematical economics. *Bulletin (New Series) of the American Mathematical Society*. 1958, 64 100-122.

[3] Sergiu Hart, and Andreu Mas-Colell. A Simple Adaptive Procedure Leading to Correlated Equilibrium. *Econometrica*. 68 (5), 1127-1150. DOI 10.1111/1468-0262.00153.

[4] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. *Regret Minimization in Games with Incomplete Information*. In: *Proceedings of the 20th International Conference on Neural Information Processing Systems*. USA: Curran Associates Inc., 2007. 1729–1736. ISBN 978-1-60560-352-0.

[5] Oskari Tammelin. Solving Large Imperfect Information Games Using CFR+. 2014,

[6] Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Computer science. Heads-up limit hold'em poker is solved. *Science (New York, N.Y.)*. 2015, 347 145-9. DOI 10.1126/science.1259433.

[7] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. DeepStack: Expert-Level Artificial Intelligence in No-Limit Poker. *Science*. 2017, 356 DOI 10.1126/science.aam6960.

[8] Viliam Lisý. *Monte Carlo Tree Search in Imperfect-Information Games*. Ph.D. Thesis, Czech Technical University in Prague. 2014.
`http://cyber.felk.cvut.cz/teaching/radaUIB/disertace_Lisy%20Viliam.pdf`.

[9] Marc Lanctot. *Monte Carlo Sampling and Regret Minimization for Equilibrium Computation and Decision-Making in Large Extensive Form Games*. Ph.D. Thesis, University of Alberta. 2013.
`http://mlanctot.info/files/papers/PhD_Thesis_MarcLanctot.pdf`.

[10] Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. *Monte Carlo Sampling for Regret Minimization in Extensive Games*. 2009.
`http://papers.nips.cc/paper/3713-monte-carlo-sampling-for-regret-minimization-in-extensive-games.pdf`.

[11] Noam Brown, and Tuomas Sandholm. Superhuman AI for multiplayer poker. *Science*. 2019, eaay2400. DOI 10.1126/science.aay2400.

[12] Neil Burch. *Time and Space: Why Imperfect Information Games are Hard*. Ph.D. Thesis, Department of Computing Science, University of Alberta. 2017. https://poker.cs.ualberta.ca/publications/Burch_Neil_E_201712_PhD.pdf.

[13] Richard Gibson, Marc Lanctot, Neil Burch, Duane Szafron, and Michael Bowling. *Generalized Sampling and Variance in Counterfactual Regret Minimization.* In: *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*. AAAI Press, 2012. 1355–1361.

[14] G. Peter Lepage. A New Algorithm for Adaptive Multidimensional Integration. *J. Comput. Phys.*. 1978, 27 192. DOI 10.1016/0021-9991(78)90004-9.

[15] J. Goodman. *Monte Carlo methods.* In: *Monte Carlo methods, Goodman, Fall 2007*. 2007. https://www.math.nyu.edu/faculty/goodman/teaching/MonteCarlo07/.

[16] Michael H. Bowling, Michael Johanson, Neil Burch, and Duane Szafron. *Strategy evaluation in extensive games with importance sampling.* In: 2008. 72-79.

[17] Neil Burch, Martin Schmid, Matej Moravcik, and Michael H. Bowling. AIVAT: A New Variance Reduction Technique for Agent Evaluation in Imperfect Information Games. *CoRR*. 2016, abs/1612.06915

[18] Martin Schmid, Neil Burch, Marc Lanctot, Matej Moravcik, Rudolf Kadlec, and Michael Bowling. Variance Reduction in Monte Carlo Counterfactual Regret Minimization (VR-MCCFR) for Extensive Form Games using Baselines. *CoRR*. 2018, abs/1809.03057

[19] Martha White, and Michael Bowling. *Learning a Value Analysis Tool For Agent Evaluation.* In: *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI)*. 2009. 1976–1981.

[20] AI Center. *gtlibrary-java*. https://github.com/aicenter/gtlibrary-java. 2014.

[21] Finnegan Southey, Michael Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, and D. Chris Rayner. Bayes' Bluff: Opponent Modelling in Poker. *CoRR*. 2012, abs/1207.1411

[22] Mads Haahr. *RANDOM.ORG: True Random Number Service*. https://www.random.org. 1998–2018. Accessed: 2018-06-01.

[23] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, .Ilhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*. 2020, 17 261–272. DOI https://doi.org/10.1038/s41592-019-0686-2.

[24] Trevor Davis, Martin Schmid, and Michael Bowling. Low-Variance and Zero-Variance Baselines for Extensive-Form Games. *CoRR*. 2019, abs/1907.09633

Appendix A
Enclosed CD

- `experiments` - folder containing experimental data
- `text` - folder containing Tex sources of the thesis
- `GTlib2` - folder containing source codes of GTlib2
  - `algorithms\mccfr.cpp` - source file containig implementation of MCCFR and MCCFRb algorithms
  - `experiments\dp` - folder containing experimental data allowing the rerun of experiments