Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

DIPLOMOVÁ PRÁCE

Adaptivní Texas Hold'em limit pokerbot pro více hráčů

2016

Bc. Jan Vytřísal

Vedoucí práce: RNDr. Jan Konečný, Ph.D.

Studijní obor: Informatika, prezenční forma

Bibliografické údaje

Autor:	Bc. Jan Vytřísal
Název práce:	Adaptivní Texas Hold'em limit pokerbot pro více hráčů
Typ práce:	diplomová práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2016
Studijní obor:	Informatika, prezenční forma
Vedoucí práce:	RNDr. Jan Konečný, Ph.D.
Počet stran:	47
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliographic info

Author:	Bc. Jan Vytřísal
Title:	Adaptive multiplayer Texas Hold'em limit pokerbot
Thesis type:	master thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2016
Study field:	Computer Science, full-time form
Supervisor:	RNDr. Jan Konečný, Ph.D.
Page count:	47
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

V práci se zabývám vytvořením adaptivního Texas Hold'em limit pokerbota pro více hráčů. Vycházím z pokerbotů pro dva hráče, Vexbot a BRPlayer. Zobecňuji jejich algoritmy Miximax a Miximix. Algoritmus Miximax používám pro hru tří hráčů. V úvahu beru rozsáhlost herního stromu pokeru a zavádím abstrakce, které ho zjednodušují. Výsledkem je pokerbot LAIN. Testování ukazuje, že se LAIN dokáže přizpůsobit více soupeřům. S vyšším počtem odehraných ruk se zlepšuje schopnost LAIN využívat strategii soupeřů ve svůj prospěch. LAIN dokáže hrát proti více než dvěma soupeřům, avšak její abstrakce pro to nebyly navrženy. S přibývajícím počtem soupeřů se LAIN přizpůsobuje čím dál pomaleji. Techniky pro rychlejší učení mohou LAIN výrazně pomoci.

Synopsis

The focus of the thesis is to create an adaptive multiplayer Texas Hold'em limit pokerbot. My work is based on two player pokerbots, Vexbot and BRPlayer. I'm generalizing their algorithms Miximax and Miximix. I'm using Miximax for three player game. I'm introducing abstractions to reduce extensiveness of a game tree. The result is the LAIN pokerbot. Testing shows that LAIN is able to adapt to multiple opponents. The more hands LAIN plays the better is her ability to exploit strategies of her opponents. LAIN is able to play a game with more than two opponents, but her abstractions weren't designed to do this. The more opponent is here to face, the slower will be her ability to adapt. The methods for faster learning could help LAIN greatly.

Klíčová slova: pokerbot; poker; texas hold'em limit; více hráčů; modelování soupeře; adaptace;

Keywords: pokerbot; poker; texas hold'em limit; multiplayer; opponent modelling, adaptation;

Děkuji vedoucímu práce Janu Konečnému za rady a volnou ruku. Moji matce Ivetě za neustálou podporu. Markétě za to, že na mě dohlížela.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Poker je doménou *neúplných informací* – hráč nezná karty soupeřů a nemá jistotu, jaké se otočí další společné karty. Aby byl hráč úspěšný, musí zaujmou lepší strategii, než jakou používají jeho soupeři. Dobrý lidský hráč čerpá ze zkušeností z minulých her, kalkuluje pravděpodobnost výskytu konkrétních karet a přizpůsobuje se tomu, co si o něm myslí ostatní. Používá mentální zkratky pro vyřešení komplikované hry.

Dva hlavní směry pokerové strategie v oblasti umělé inteligence jsou *nalezení optimální strategie* a *modelování soupeřů*. Optimální strategie zajišťuje výplatní neutralitu – hráč s dlouhodobého hlediska neprohraje. Nevyužívá ale soupeřových chyb.

V práci se zabývám modelováním soupeřů, které slouží k zvýšení výplaty. Počítačový hráč pokeru (pokerbot) si ukládá soupeřem odhalené informace. Získává tak možnost využít soupeřovy strategie ve svůj prospěch. V důsledku ovšem podstupuje riziko, že bude sám modelován.

Modelování rozlišujeme *adaptivní* a *statické*. Adaptivní modeluje soupeře v reálném čase. Statické vytváří offline model soupeře, který poté použije v reálné hře.

Mnou zvolená technika adaptivního modelování stojí na průchodu herního stromu s neúplnou informací, ve kterém soupeře zastupuje vytvořený model. Potíž je velikost herního stromu. Pro variantu pokeru Texas Hold'em limit dvou hráčů má 10^{18} uzlů – není možné ho prohledat v reálném čase v průběhu hry. S přibívajícím počtem hráčů strom hry exponenciálně roste[1, str. 72]. Je nutností zavést *abstrakce*, které sníží počet uzlů stromu.

Původní algoritmy průchodu herního stromu s neúplnou informací nazvané *Miximax* a *Miximix*[1, 2] byly navrženy a otestovány ve hře dvou hráčů. Autoři v závěru nastínili možnost rozšíření algoritmů pro hru více hráčů. V práci provádím zobecnění obou algoritmů, které otestuji na vlastní implementaci algoritmu Miximax ve hře tří hráčů.

Z množství dostupných technik modelování soupeřů (kap. 2) jsem pro svoji práci zvolil algoritmy Miximax a Miximix, protože v testování dosahovaly dobrých výsledků a zároveň zde autoři vznesli výzvu prozkoumat, jak si technika povede ve hře více než dvou hráčů.

V kapitole 2 procházím techniky vytvoření modelujících pokerbotů. V první části kapitoly popisuji adaptivní pokerboty, v druhé části statické pokerboty. V podkapitole 2.1.3 se podrobně zabývám pokerboty vytvořenými nad algoritmy Miximax a Miximix.

V kapitole 3 popisuji vytvoření pokerbota LAIN – zobecnění výchozích algoritmů na hru více hráčů, zavedení abstrakcí pro hru tří hráčů.

Na začátku kapitoly 4 zkoumám, jak silné abstrakce potřebuji. Následuje tes-

tování LAIN v prostředí `opentestbed`[1] a `Poker Academy Pro 2.5`[2] proti různým pokerbotům.

V závěru vyhodnotím výsledky testování a nastíním další možný vývoj pokerbota LAIN.

1.1 Pravidla pokeru

Pravidla varianty pokeru Texas Hold'em limit. Každý hráč na začátku obdrží dvě karty, které zná pouze on – tzv. *držené karty*. Hráči se střídají v sázkách a otáčí se *společné karty*. Na konci posledního sázkového kola hráči odhalí držené karty. Hráč s nejsilnější výherní kombinací vyhrává všechny vsazené žetony. Dále v podkapitole popisuji pravidla podrobně.

Pojem *ruka* značí část hry od rozdání karet všem hráčům po určení vítěze.

Hrací balíček
: Obsahuje 52 karet. Každá karta je unikátní, specifikuje ji *hodnota* a *barva*. Hodnot je třináct: 2, 3, 4, 5, 6, 7, 8, 9, T (deset), J (kluk), Q (královna), K (král), A (eso). Barvy jsou čtyři: ♢♡♠♣. Eso může být použito i jako karta hodnoty 1.

Nastavení hry
: Hru hraje až deset hráčů. Z hráčů je náhodně zvolen *dealer*. Hráč po levici dealera je *small blind* a hráč po levici small blinda je *big blind*. Hráči disponují stejnou sumou hracích žetonů, tzv. *stack*.

Sázkové kolo
: Hra má čtyři sázkové kola: *preflop*, *flop*, *turn* a *river*. Hráči se střídají v sázkách dokud nemají všichni hráči srovnané sázky. Následuje další sázkové kolo. Maximální povolený počet sázek na hráče v sázkovém kole je čtyři.

Preflop
: Jedná se o začátek hry. Každý hráč obdrží dvě náhodné karty, tvz. držené karty. Hráč big blind vsadí předem stanovenou povinnou sázku, ozn. *malá sázka*. Hráč small blind polovinu malé sázky. První na tahu bude hráč po levici hráče big blind.

Hráč na tahu
: Hráč na tahu provede jednu ze tří akcí: *zahodí karty*, *dorovná sázku*, *navýší sázku*. Po provedení akce bude na tahu první hráč po jeho levici, který je stále ve hře (nezahodil karty). Jakmile mají všichni pokračující hráči srovnané sázky, začíná další sázkové kolo.

Zahození karet
: Hráč přichází o všechny vsazené ženoty a neúčastní se dalších sázkových kol.

[1] https://code.google.com/archive/p/opentestbed/

[2] http://www.poker-academy.com/, http://www.poker-genius.com/

Dorovnání sázek
Hráč dorovnává sázku prvního hráče po své pravici, který je stále ve hře.

Navýšení sázek
Hráč dorovná sázku a navýší ji o velikost malé sázky v kolech preflop a flop a nebo o dvojnásobek malé sázky v kolech turn a river.

Flop
Z balíčku se otočí 3 společné karty. Sázet začíná první hráč po levici dealera, který je stále ve hře.

Turn
Z balíčku se otočí 1 společná karta. Sázet začíná začíná první hráč po levici dealera, který je stále ve hře.

River
Sázkové kolo probíhá totožně jako sázkové kolo Turn. Mají-li hráči srovnané sázky, nastává *showdown*.

Showdown
Hráči odhalí držené karty. Porovná se jejich *výherní kombinace*. Hráč s nejsilnější výherní kombinací vyhrává všechny vsazené žetony, tzv. *bank*.

Výherní kombinace
Je kombinace pěti nejlepších karet vybraná z držených karet hráče a společných karet 1. Pokud má výherní kombinace méně než pět karet, přidává se k ní *kicker*.

Kicker
Jsou doplněkové karty vybírané z dostupných společných a držených karet, které se přidávají k výherní kombinaci. Karty jsou vybírány od nejvyšší. Pokud je více vítězů se stejně silnou výherní kombinací, vyhraje hráč s vyšším kickerem. Při rovnosti kickerů je mezi výherce bank rovnoměrně rozdělen.

U živého pokeru je často přítomno pravidlo *skrytí držených karet*. Hráči na konci odehrané ruky ukazují držené karty postupně. Pokud hráč, který má ukázat karty ví, že již nemůže vyhrát, nemusí ukazovat držené karty. Situace nastane, když hráč dříve na řade ukázal silnější výherní kombinaci.

Přítomnost pravidla skrytí držených karet v tomto textu nepředpokládám. Zmíním se však o způsobu, jak ho řešit v pokerbotovi LAIN (kap. 3.3.3) a řešení otestuji (kap. 4.2.4).

Tabulka 1: Výherní kombinace. Řazeny od nejslabší po nejsilnější. Podtržené karty značí výherní kombinaci, ostatní karty jsou kickery.

Nejvyšší karta	K♠ 9♢ 7♣ 3♡ 2♣
Pár	K♠ K♢ 7♣ 3♡ 2♣
Dva páry	K♠ K♢ 7♣ 7♡ 2♣
Trojice	K♠ K♢ K♣ 3♡ 2♣
Postupka	K♠ Q♢ J♣ T♡ 9♣
Barva	K♠ 9♠ 7♠ 3♠ 2♠
Full House	K♠ K♢ K♣ 3♡ 3♣
Čtveřice	K♠ K♢ K♣ K♡ 2♣
Čistá postupka	K♠ Q♠ J♠ T♠ 9♠

2 Přehled technik modelujících pokerbotů

V kapitole shrnu známé modelující pokerboty. V první části popíši *adaptivní* pokerboty, kteří modelují soupeře v reálném čase. V druhé části se zabývám *statickými* pokerboty, kteří vytvoří offline model soupeře a použijí ho v reálné hře.

Modelování soupeřů obecně má dvě části:

- sbírání dat o soupeři,
- využití nasbíraných dat proti soupeři.

2.1 Adaptivní pokerboti

Výhodou adaptivních pokerbotů je, že není nutné dělat obsáhlé výpočty dopředu a připravovat se na konkrétního soupeře. Adaptivní pokrbot se dokáže přizpůsobit soupeři, proti kterému právě hraje. Není podmínkou, aby o soupeři cokoliv věděl dopředu.

Nevýhodou je začátek hry, kdy adaptivní pokerbot nemá žádné informace a spoléhá se pouze na svůj výchozí model. Podobně to může být i u statických pokerbotů, ale vypočítaná offline strategie jim dává podstatnou výhodu.

2.1.1 Rovnicový systém

Jednim z prvních adaptivních pokerbotů je **Poki**[1]. Jeho rozhodovací strategie je založená na *rovnicích*, na základě kterých Poki vyhodnotí aktuální herní situaci. Výsledkem je vygenerování pravděpodobnostní trojice akcí zahození, dorovnání a navýšení, která udává pravděpodobnost provedení akcí. Například pro vygenerovanou trojici $\{0, 0.8, 0.2\}$ v 0 % případů zvolí zahození, 80 % případů zvolí dorovnání, ve 20 % navýšení. Uvedu hlavní rovnice:

Síla výherní kombinace (*HR*) Je vyčíslení udávající kolik procent ze všech možných soupeřových výherních kombinací Pokiho výherní kombinace porazí. Výherní kombinaci Poki tvoří z držených karet a společných karet. Soupeřovy výherní kombinace tvoří všechny možné zbývající držené karty a společné karty. Předpokladem je, že soupeřovy držené karty jsou všechny zastoupeny se stejnou pravděpodobností. Výsledná síla výherní kombinace je $HR = \frac{vyhra + \frac{remiza}{2}}{vyhra + prohra + remiza}$, kde *vyhra*, *remiza* a *prohra* jsou počty výher, remiz a proher Pokiho výherní kombinace proti soupeři.

Potenciál výherní kombinace Udává pravděpodobnost, že se Pokiho výherní kombinace zlepší s budoucími společnými kartami.

Návratnost investice Kolik nejméně musí mít Poki pravděpodobnost na výhru, aby mělo smysl dorovnat sázku. Například pokud v banku je 9 žetonů a Poki musí dorovnat 1 žeton, musí mít Poki pravděpodobnost na výhru větší než $\left(\frac{1}{9+1}\right) \cdot 100 = 10\,\%$, aby mohl očekávat kladnou návratnost investice.

Pro každého soupeře si Poki udržuje *tabulku vah*, což je pole o 1326 hodnotách. Každá hodnota je v intervalu $\langle 0, 1\rangle$ a odpovídá pravděpodobnosti s jakou Poki věří, že soupeř drží danou kombinaci karet. Tabulka vah je aktualizována po každé zpozorované akci soupeře. Vstupem pro aktualizaci je model soupeře, provedená akce a aktuální stav hry. Pro každou drženou kombinaci karet e z tabulky vah WT je prediktorem vypočítána pravděpodobnost p, že soupeř v aktuálním stavu hry provede zpozorovanou akci. Aktualizace je provedena jako $WT[e] = WT[e] \cdot p$.

Když Poki musí provést akci, vypočítá sílu své výherní kombinace (*HR*) proti každému soupeři zvlášť. Tabulka vah ovlivní, které kombinace držených karet jsou konkrétnímu soupeři přiřazeny. Výpočtem Poki získá pravděpodobnost na výhru své výherní kombinace proti jednotlivým soupeřům. Vynásobením jednotlivých pravděpodobností dostane celkovou pravděpodobnost, že porazí všechny soupeře zároveň. Spolu s faktory jako je potenciál výherní kombinace a návratnost investice, Poki vygeneruje pravděpodobnostní trojci akcí zahození, dorovnání a navýšení na základě které provede akci.

Model soupeře je realizovaný *prediktorem* akcí. Pro herní stav vrací pravděpodobnostní trojici akcí soupeře. Poki zpravidla využívá tři metody predikce, které kombinuje a vybírá nejpřesnější z nich:

Expertní systém Vypočítá pravděpodobnostní trojici pro soupeře z Pokiho vlastní rovnicové strategie. Předpokládá, že soupeř hraje racionálně.

Statistika Poki si pro každého soupeře uchovává tabulku kontextů, do které ukládá zpozorované akce. Za kontext je považováno sázkové kolo a počet sázek, kterým soupeř čelí. Příklad dotazu pro kontextovou tabulku je: $P\left(AkceDorovnani \mid SazkoveKoloTurn \,\&\, JednaSazkaKDorovnani\right)$, což

vrátí pravděpodobnost akce dorovnání vzhledem k sázkovému kolu turn a situaci, kdy soupeř musí dorovnat jednu sázku.

Neuronovou síť Tréningovou množinou jsou akce konkrétního soupeře a herní kontext, ve kterém akci provedl. Pro zvolený kontext neuronová síť vrací akci, která má největší pravděpodobnost na provedení.

2.1.2 Monte Carlo simulace

Simbot[3][1] je reimplementace pokerbota Poki, kde je místo rovnic použita *Monte Carlo simulace*[4] (MCS). Když Simbot musí provést akci, zavolá MCS pro akce dorovnání a navýšení, aby vypočítal jejich očekávanou hodnotu (EV). MCS projde herní strom do hloubky, v listových uzlech získá EV a propaguje ho zpět nahoru. MCS se pro každou akci spustí několiksetkrát. Získaná EV se zprůměrují a Simbot provede akci, která je nejpříznivější.

Model soupeře je použit ve dvou situacích při průchodu stromem MCS algoritmem:

Přiřazení držených karet soupeři Na začátku každého průchodu jsou všem soupeřům přiřazeny držené karty podle jejich tabulky vah – přiřazuje se podle nejpravděpodobnější kombinace. Držené karty jsou přiřazeny i soupeřům, kteří ve hře provedli akci zahození, aby v hracím balíčku zůstaly co nejpravděpodobnější karty. Z hracího balíčku jsou náhodně zvoleny společné karty. Každý průchod je se stejně zvolenými drženými kartami proveden dvakrát – jednou pro Pokiho akci dorovnání a jednou pro akci navýšení.

Volba akce za soupeře V průběhu průchodu stromem MCS algoritmus musí volit akce za každého soupeře. Volba je provedena preferovaným prediktorem akcí.

Monte Carlo simulace dokáže nalézt strategie pro Simbota jako je *čekání-navýšení*, kdy Simbot čeká nebo dorovná sázku s předpokladem, že očekává soupeřovu akci navýšení, aby rázem sám navýšil. Také umí strategii *maskování* držených karet pasivní hrou, ve které pouze dorovnává soupeře a zakrývá tak skutečnou sílu své výherní kombinace. Soupeř je spíše ochoten sázet, než kdyby ho Simbot donutil zahodit karty agresivní hrou. V neposlední řadě je přítomna i strategie *blafování* – Simbot navyšuje s velmi slabou výherní kombinací, protože předpokládá, že soupeř často zahodí karty.

Přes potenciál a větší adaptovatelnost MCS oproti Rovnicovému systému Simbot nedosahoval v testování lepších vysledků než Poki. Simbot se umí více

[3]V původním článku označen jako Poki se Selective Sampling simulací. V komerčním pokerovém testovacím prostředí Poker Academy 2.5 je nazýván Simbot. Označení Simbot používám kvůli lepšímu rozlišení mezi pokerboty.

[4]Označena jako Selective Sampling, protože průchod herním stromem není náhodný, ale řízený.

přizpůsobit soupeři, kvůli čemuž je ale více náchylný na chyby v modelu. Pokud model předpokládá, že soupeř zahodí karty ve 30 % případu a ve skutečné strategii soupeř zahazuje jen ve 20 % případů, Simbot bude hrát více agresivněji, než by měl. V důsledku bude hrát slabší výherní kombinace, protože předpokládá časté soupeřovo zahazování, což se negativně projeví na jeho výplatě.

2.1.3 Prohledávání herního stromu s neúplnou informací

Stěžejní pro moji práci jsou pokerboti **Vexbot**[2], jeho vylepšená verze **BRPlayer**[3] a algoritmy *Miximax* a *Miximix*[1, 2], na kterých jsou postaveni. Pokerboti jsou určeni pro hru dvou hráčů a líší se pouze v aplikaci abstrakcí při získávání informací. Vzhledem k důležitosti pokerbotů se o nich rozepíši podrobněji.

Jádro pokerbotů má dvě části:

- prohledávání herního stromu,
- modelování soupeře.

V následujícím textu souhrnně označuji pokerboty Vexbot a BRPlayer jako pokerbot.

Pokerbot na tahu spustí algoritmus Miximax nebo Miximix postupně pro akce zahození, dorovnání a navýšení, aby vypočítal jejich očekávanou hodnotu (EV). Algoritmus z aktuální herní situace začne prohledávat herní strom do hloubky. V listech vypočítá EV a propaguje ho zpět směrem nahoru. Po získání EV všech tří kořenových akcí dojde k jejich porovnání. Byl-li spuštěn Miximax, zvolí akci s nejvyšší EV. Byl-li spuštěn Miximix, zvolí akci na základě smíšené strategie[5]. Stejně tak když volí akce za sebe sama v průchodu herním stromem, provede buď akci s nejvyšší EV při Miximaxu nebo akci na základě vlastní smíšené strategie při Miximixu.

Pokerbot modeluje soupeře při dvou příležitostech:

- soupeř odhalí držené karty,
- soupeř provede akci.

Když dojde k odhalení držených karet soupeře, pokerbot vypočítá, sílu jeho výherní kombinace (HR) a uloží ji do histogramu v listu *kontextového stromu*. Kontextový strom je datová struktura typu *trie*, ve které uzly odpovídají rozhodovacím uzlům herního stromu. Čím větší vzorek bude mít pokerbot uložený, tím přesněji bude vědět, jaké HR má od soupeře očekávat v konkrétním kontextu hry.

[5]Smíšená stragie zde odpovídá pravděpodobnostní trojici akcí zahození, dorovnání, navýšení. Například trojice $(0.1, 0.7, 0.2)$ značí, že se akce zahození provede s pravděpodobností 10 %, akce dorovnání s pravděpodobností 70 % a akce navýšení s pravděpodobností 20 %.

Kdykoliv provede soupeř akci, pokerbot ji uloží do odpovídajícího uzlu kontextového stromu. Pokerbot tímto způsobem modeluje tendence soupeře – jeho pasivitu/agresivitu při sázení.

Procházení herního stromu i modelování soupeře vyžaduje zavedení *abstrakcí*. Při procházení herního stromu abstrakce redukují jeho rozvětvení a jsou dvojího typu: úprava otáčení společných karet a vynechávání větví stromu. Abstrakce prvního typu jsou potřeba pouze v sázkových kolech preflop a flop:

Preflop Abstrakce seskupí všechny budoucí společné karty v uzlech šance do několika *škatulek* a přiřadí jim pravděpodobnosti s jakou nastanou.

Flop Změna v prohledávání nastane až v sázkovém kole river, kde se otočí pouze malá podmnožina ze 46 možných karet.

Abstrakce druhého typu vynechávají větve stromu s nulovou pravděpodobností na provedení.

Abstrakce při modelování soupeřů sjednocuje histogramy na základě podobnosti kontextu. Slouží k urychlení učení. Vexbot rozlišuje tři úrovně abstrakcí: kontextový strom, s1 a s2. BRPlayer rozlišuje pět úrovní abstrakcí. Úrovně jsou rozděleny podle stupně vynechání informací o hře. Každá úroveň má dvě varianty: v první začíná pokerbot, v druhé soupeř. Úrovně jsou seřazeny od kontextů nejvíce podobných původní hře po nejméně podobné:

Kontextový strom Abstrakce na úrovní kontextového stromu odpovídá přesné sekvenci akcí – maximum informací na úkor řídkosti dat.

s4 Abstrakce seskupuje všechny histogramy kontextového stromu, kde je stejný počet sázek v sázkových kolech preflop plus flop a turn plus river. Například všechny histogramy s dvěma sázkami v sázkových kolech preflop a flop a třemi sázkami v sázkových kolech turn a river. Sčítají se sázky obou hráčů.

s3 Seskupení histogramů podle počtu sázek v prvních třech sázkových kolech a posledním sázkovém kole.

s2 Seskupuje histogramy, které mají počet sázek ve všech čtyřech sázkových kolech stejný.

s1 Seskupuje histogramy pouze podle počtu sázek ve všech čtyřech sázkových kolech provedených soupeřem. Zachová minimum informací na úkor potřebě mít co nejvíce dat.

Kontextový strom je také formou abstrakce, protože se do něho neukládají karty pokerbota ani otočené společné karty. Předpokládá se otočení splečných karet se stejnou pravděpodobností. Pokerboti Vexbot a BRPlayer využívají kontextový strom při ukládání soupeřových akcí stejným způsobem.

Histogram v listovém uzlu je pole reprezentující interval $\langle 0.0, 1.0\rangle$ rovnoměrně rozdělený na 20 podintervalů. První prvek odpovídá všem *HR* v intervalu $\langle 0.0, 0.05)$. Druhý prvek všem *HR* v intervalu $\langle 0.05, 0.1)$. Poslední prvek $\langle 0.95, 1.0\rangle$.

Pokerbot rozlišuje čtyři druhy uzlů při prohledávání herního stromu:

Uzel šance

Odpovídá rozdání společných karet. Z důvodu zjednodušení výpočtu se předpokládá stejná pravděpodobnost pro každou kartu. Celkový počet kombinací rozdání karet je n. Očekávaná hodnota uzlu šance u je $EV(u)$. $P(u_i)$ je pravděpodobnost, že nastane uzel u_i. $EV(u_i)$ je očekávaná hodnota uzlu u_i.

$$EV(u) = \sum_{i=1}^{n} P(u_i) \cdot EV(u_i)$$

Soupeřův rozhodovací uzel

Uzly, kde se pokerbot rozhoduje za soupeře na základě jeho předchozího zpozorovaného chování ve stejném kotextu. Má uloženou soupeřovu pravděpodobnostní trojici akcí zahození (f), dorovnání (c), navýšení (r), které jsou smíšenou strategií uzlu u.

$$EV(u) = \sum_{i \in \{f,c,r\}} P(u_i) \cdot EV(u_i)$$

Vlastní rozhodovací uzel

Pokerbot volí akci na základě smíšené strategie podobně stejně jako u soupeřova rozhodovacího uzlu (miximix) nebo volí akci s nejvyšším EV (miximax).

$$EV(u) = \max\{EV(f), EV(c), EV(r)\}$$

Listový uzel

Výplata z pohledu pokerbota – zisk nebo ztráta. $P(vyhra)$ značí pravděpodobnost výhry pokerbota v uzlu u. Pravděpodobnost je spočítána z histogramu ruk, které soupeř pokerbotovi dříve ukázal ve stejném herním kontextu. *bank* je celkový počet vsazených žetonů. *cena* je kolik žetonů v banku má pokerbot.

$$EV(u) = (P(vyhra) \cdot bank) - cena$$

Příklad výpočtu očekávaných hodnot akcí (EV). Pro jednoduchost uvažujme histogramy s 10 podintervaly. Je sázkové kolo river, doposud každý hráč vložil do banku 1 žeton. Velikost sázky je 2 žetony. Vexbot se nachází na vrcholu herního stromu (obr. 1) a potřebuje vypočítat EV akce navýšení sázky (r).

Vexbot provede akci navýšení (r) a přesune se do uzlu soupeře. Zde musí vypočítat EV soupeřových akcí zahození (F), dorovnání (C), navýšení (R) a provést jejich váženou sumu vzhledem k pravděpodobnostní trojici $\{0.2, 0.7, 0.1\}$.

Zahození

Akce zahození má očekávanou hodnotu $EV(rF) = +1$. Pokerbot získá 1 žeton.

Dorovnání

Akce dorovnání závisí na síle výherní kombinace (HR) Vexbota. Histogram [rC] znázorňuje, že soupeř v 20 % případů blafuje – má HR v intervalu $\langle 0.0, 0.2)$. V 80 % případů hraje se silnou rukou – HR v intervalu $\langle 0.8, 1.0\rangle$. Mohou nastat čtyři situace vzhledem k HR Vexbota:

Vexbot blafuje, má $HR = 0.1$. $EV(rC) = (0.1 \cdot 6) - 3 = -2.4$.
Porazí pouze blaf, $HR = 0.2 - 0.7$. $EV(rC) = (0.2 \cdot 6) - 3 = -1.8$.
Porazí polovinu silných ruk, $HR = 0.9$. $EV(rC) = (0.6 \cdot 12) - 6 = 1.2$.
Porazí všechny ruky, $HR = 1$. $EV(rC) = (1.0 \cdot 12) - 6 = 6$.

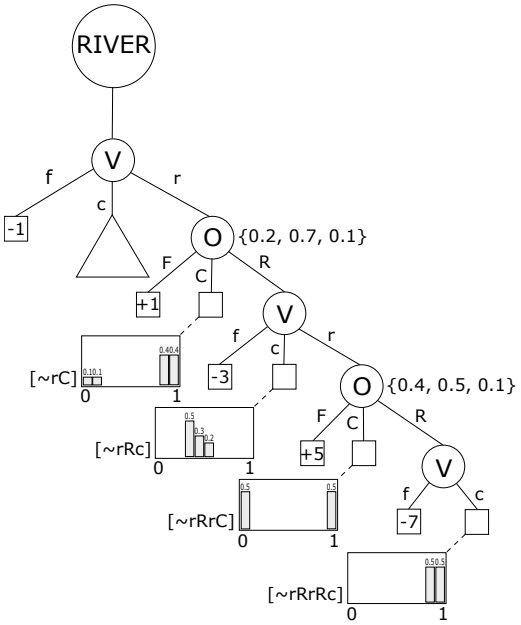
Navýšení

Pro výpočet $EV(rR)$ je potřeba spočítat $max\,(EV(rRf), EV(rRc), EV(rRr))$, používá-li Vexbot Miximax. V případě použití algoritmu Miximix by aplikoval vlastní smíšenou strategii a provedl váženou sumu pro $EV(rRf)$, $EV(rRc)$, $EV(rRr)$. EV sekvence rRf je $EV(rRf) = -3$. EV sekvence rRc bude opět záležet na HR Vexbota, výpočet viz výše. EV sekvence rRr bude výsledkem vážené sumy EV sekvencí rRrF, rRrC, rRrR. Nakonec $EV(rRrR)$ se spočítá jako $\max\,(EV(rRrRf), EV(rRrRc))$.

Má-li Vexbot průměrně silnou výherní kombinaci $HR = 0.5$, $EV(r)$ za použití algoritmu Miximax bude $EV(r) = -0.76$. Připomeňme, že $EV(f) = -1$. Pokud tedy $EV(c) > -0.76$, Vexbot zvolit akci dorovnání. V opačném případě akci navýšení.

Příklad modelování soupeře. Uvažujme poslední sázkové kolo. Pokerbot začne sázkou (r). Soupeř sázku navýší (R) – pokerbot v kontextovém stromu vyhledá kontext „~rR“ a uloží zpozorovanou akci. Pokerbot pokračuje dorovnáním sázky (c). Dojde k odhalení karet obou hráčů. Pokerbot vypočítá HR soupeře a uloží ho do histogramu v listovém uzlu kontextového stromu, který najde pod klíčem „~bRc“. Znak ~ značí libovolnou sekvenci akcí, které proběhly v dřívějších sázkových kolech. Pro zjednodušení příkladu nebyly uvedeny. V praxi probíhá ukládání v obou případech zvýšením počítadla o 1.

V testování proti různým pokerbotům Vexbot i BRPlayer vykazují podobný trend – čím více odehrají ruk proti konkrétnímu soupeři, tím lépe se mu dokáží přizpůsobit. S dostatkem času jsou schopni opravit chybný model soupeře. Oprava však může trvat tisíce až desítky tisíc odehraných ruk v závislosti na složitosti soupeřovy strategie a chyby modelu. Vytvoření chybného modelu soupeře nastane z důvodu karetní variance. Soupeř ukazuje stále silné výherní kombinace v konkrétní situaci hry, v důsledku čehož se pokerbot chybně začne vyhýbat akcím, které k herní situaci vedou.



Obrázek 1: Ukázka stromu hry, který prohledává Vexbot při výpočtu očekávaných hodnot. Hra se nachází v sázkovém kole river. Doposud každý hráč vsadil 1 žeton. *V* značí rozhodovací uzel Vexbota, *O* rozhodovací uzel soupeře. V listech jsou vyznačeny očekávané hodnoty (EV). Obdelníky jsou histogramy síly výherních kombinací, které soupeř ukázal v konkrétním uzlu herního stromu.

Autoři poukazují na důležitost dobrého výchozího modelu, aby pokerbot ze začátku hry neprohrával přiliš mnoho. Potřeba rychlejšího učení se pro hru proti lidským hráčům.

2.2 Statičtí pokerboti

V kapitole uvedu přehled pokerbtů, kteří vytváří model soupeře offline – ne v reálném čase. Výhoda oproti adaptivním pokerbotum spočívá v silné strategii již od začátku hry. Nemusí spoléhat pouze informace, které jim o sobě poskytne soupeř při hře.

2.2.1 Statisticky Nejlepší Odpověď

Algoritmus Statisticky Nejlepší Odpověď (SNO)[4, 5] z tréningových dat vytvoří model soupeře a vypočítá strategii, která model nejvíce využívá. Vypočítanou strategii použije v reálné hře proti cílovému soupeři. V kapitole rozepíši SNO podrobněji.

V první řadě je potřeba získat velké množství her odehraných cílovým soupeřem, kde jsou odhalené držené karty všech hráčů. Autoři uvádí alespoň 100000 až 1 milion odehraných her. S větším množstvím dat se zvyšuje přesnost modelu soupeře. Aby byla vypočítaná strategie co nejlepší, je potřeba cílového soupeře

pozorovat v co nejvíce různých situacích. Proto je důležité proti jakým soupeřům náš cílový soupeř hraje. Soupeř, který by v každé fázi hry volil akci dorovnání, neposkytne příliš mnoho informací o strategii cílového soupeře.

Z nasbíraných dat je vytvořen model soupeře. Nejdříve algoritmus zvolí abstrakci původní hry Texas Hold'em limit poker pro dva hráče, kterou redukuje její herní strom z přibližně 10^{18} uzlů na přibližně 6.45^9 uzlů. Postupně prochází odehrané hry a pro každou akci cílového supeře zvýší počítadlo v odpovídajícím uzlu abstraktního stromu. Touto technikou algoritmus získá pravděpodobnostní trojice akcí zahození, dorovnání a navýšení pro soupeřovy uzly v abstraktním stromu. Odhalené karty jsou mapovány do 5 až 8 škatulek v abstraktním stromu podle síly výherní kombinace, kterou reprezentují.

V případě, že nebyly zaplněny všechny uzly cílového soupeře v abstraktním stromu, se použije výchozí strategie. Cílový soupeř vždy dorovná nebo volí akce dorovnání a navýšení s různou pravděpodobnostní.

Na vytvořený model soupeře je aplikován algoritmus nejlepší odpovědi, který vrací strategii nejvíce využívající cílového supeře. Algoritmus projde abstraktní strom modelu soupeře k listovým uzlům, získá jejich očekávanou hodnotu (EV) a propaguje ji zpět nahoru. Pro každý uzel si zapamatuje, která akce vedla k největší očekávané hodnotě. Zapamatovaná data jsou strategií algoritmu SNO.

V testování SNO dokáže velmi efektivně využívat cílového supeře. Problém nastává v situaci, kdy by SNO měl hrát proti hráči, proti kterému nebyl vytvořen. Nehraje-li nový soupeř podobnou strategii jako cílový soupeř, SNO bude značně prohrávat.

2.2.2 Omezená Nashova Odpověď

Omezená Nashova Odpověď (ONO)[4, 5] řeší problém algoritmu SNO, který je efektivní pouze proti soupeři, proti kterému byl natrénován. Algoritmus ONO místo nejlepší odpovědi na soupeřovu strategii nachází robůstní odpověď – snaží se soupeře využívat a zároveň minimalizovat šanci, že sám bude využíván. Hraje-li algoritmus ONO proti soupeři proti kterému nebyl natrénován, bude prohrávat jen do určité očekávané míry. Kompromisem je, že nebude cílového soupeře schopný využít tolik, jako SNO algoritmus.

Nashova rovnováha pro hru pokeru dvou hráču je strategie, která každemu hráči zajistí očekávanou hodnotu (EV) hry rovnou nule. Čím více spolu dva hráči odehrají ruk, tím více se výplata každého z nich bude blížit nule. Žádný z hráčů nemůže změnou strategie zlepšit EV své hry, pokud druhý hráč stále bude hrát podle Nashovy rovnováhy. Vypočítání Nashovy rovnováhy je považováno za vyřešení hry.

Vypočítání Nashovy rovnováhy pro Texas Hold'em limit dvou hráčů je velmi obtížné[6]. ε-Nashova rovnováha je Nashova rovnováha vypočítaná vzhledem k abs-

[6]Cluster čítající 200 počítačů, kde každý operoval s 24 jádry, potřeboval na výpočet Nashovy rovnováhy 68 dní[6].

traktní hře pokeru. Proměnná ε udává, o kolik nejvíce může jeden z hráčů zlepšit své EV hry, pokud nebude taktéž hrát podle ε-Nashovy rovnováhy.

Stejně jako v algoritmu SNO jsou nasbírány stovky tisíc odehraných her proti cílovému soupeři a je z nich vytvořen model.

Algoritmus ONO nalezne robůstní strategii výpočtem ε-Nashovy rovnováhy v abstraktní hře s ohledem na konstantu p. Předpokládá, že cílový soupeř bude hrát stejně jako ve vytvořeném modelu v p procentech případů, kde $p \in \langle 0, 1\rangle$ a bude se nás snažit využívat v $1 - p$ procentech případů. Pro $p = 0$ algortimus ONO vyprodukuje ε-Nashovu rovnováhu, protože model soupeře není použit. Pro $p = 1$ algoritmus ONO vyprodukuje stejnou strategii jako algoritmus SNO, protože Nashova rovnováha není použita.

Volba konstanty p zásadně ovlivní chování ONO pokerbota. S p blížícím se 1 se bude ONO pokerbot snažit cílového hráče více využívat, ale sám bude proti využívání více zranitelný. Při testování v [4, 5] byla p zvoleno přibližně 0.1, což odpovídá $\varepsilon = 0.1$ malé sázky na odehranou ruku. Algoritmus ONO dosahuje výrazně lepších výsledků než algoritmus SNO proti jinému než cílovému soupeři.

2.2.3 Tým expertů

V jádře algoritmu Týmu expertů[4, 5] je manažer, volící, který expert (pokerbot) se má použít, když je potřeba odehrát ruku. Expert je volen, aby co nejvíce využil soupeře. Algoritmus Týmu expertů řeší situaci hry proti neznámému soupeři, u něhož není k dispozici historie odehraných her, ze kterých by bylo možné vytvořit model.

Na začátku manažer vyzkouší každého experta jednou – expert odehraje ruku proti soupeři. Výplaty expertů jsou uloženy a pro odehrání další ruky je volen expert s nejvyšší výplatou. V průběhu hry se výplaty expertů průměrují podle toho, kolikrát byli zvoleni. Pro odehrání ruky se volí expert s nejlepší průměrnou výplatou, přičemž je brán zřetel, jak často byl který expert volen. Expert, který dlouho nedostal šanci odehrát ruku, ji spíše dostane než expert, který odehrál velké množství ruk v řade. Zajišťuje se tak prozkoumání možností všech expertů. Zamezí se situaci, kdy expert po vyzkoušení na začátku algoritmu má negativní výplatu, kvůli které by nebyl znovu zvolen.

Povšimněte si, že i když je algoritmus Týmu expertů z principu statický – tým expertů je pevně zvolen před začátkem hry – obsahuje adaptivní část, kdy se v průběhu hry volí nejvhodnější expert.

Pro testování byly sestaveny dva týmy expertů – tým expertů založených na algoritmu Statisticky Nejlepší Odpovědi (SNO) a tým expertů založených na algoritmu Omezené Nashově Odpovědi (ONO). Při použití týmu SNO expertů manažer rychle pozná, když najde správného experta proti neznámému soupeři. Výplata konkrétního SNO experta bude vysoká. SNO experti jsou ale velmi využivatelní, takže zkoušení různých expertů v počáteční fázi se negativně projeví na celkové výplatě algoritmu Týmu expertů. Nenajde-li manažer mezi SNO experty vhodného kandidáta, bude algoritmus pouze ztrácet na výplatě.

V takovém případě by mezi SNO experty bylo vhodné zařadit experta hrajícího podle ε-Nashovy rovnováhy.

Zkoušení ONO expertů proti neznámému soupeři má méně negativní dopad na celkovou výplatu algoritmu než u týmu SNO expertů. Každý ONO expert minimalizuje svoji vlastní využivatelnost. Nevyhoví-li žádný ONO expert, algoritmus neprohraje o příliš mnoho.

Lepších výsledů proti neznámým soupeřům dosahoval tým ONO expertů. Průměrnou výplatou předčil ε-Nashovu rovnováhu i tým SNO expertů.

2.2.4 Datově Orientovaná Odpověď

Algoritmus Datově Orientované Odpovědi (DOO)[7] vygeneruje robůstní strategii, která dokáže využívat soupeře. Je zároveň více odolna proti využívání než strategie algoritmu Omezené Nashovy Odpovědi (ONO) a potřebuje méně tréningových dat. Algoritmus DOO nachází kompromis mezi schopností využívat a vlastní zranitelností proti využívání. Podobně jako algoritmus ONO algoritmus DOO vypočítá strategii ε-Nashovy rovnováhy, kde nutí soupeře hrát s pravděpodobností p podle vytvořeného modelu. Oproti ONO algoritmu se však p vztahuje na uzly stromu se stejnou herní informací, namísto na celou strategii – je možné nastavit p ke každému uzlu zvlášť. Připomeňme, že p vyjadřuje na kolik procent pokerbot věří, že má správný model soupeře.

Rozdíly oproti algoritmu ONO ve dvou příkladech:

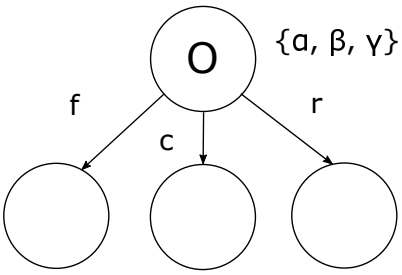
Uzel bez informace V modelu soupeře jsou uzly, které nebyly navštíveny při konstrukci modelu z tréningové množiny odehraných her soupeřem. V těchto případech algoritmus ONO volil akci dorovnání jako výchozí, což velmi pravděpodobně neodpovídalo reálné strategii soupeře. Algoritmus DOO pro tento uzel volí hodnotu $p = 0$, která odpovídá použití Nashovy rovnováhy na celý podstrom tohoto uzlu.

Uzel s málo informacemi Čím je uzel v modelu vzdálenější od kořene, tím spíše je nepřesnější pravděpodobnostní trojice akcí zahození, dorovnání a navýšení vzhledem k reálné strategii soupeře. Alogoritmus DOO určí přesnost pravděpodobnostní trojice podle celkového počtu pozorování v uzlu a přizpůsobí tomu hodnotu p. S málo informacemi se p bude blížit k 0.

Výsledky testování ukázaly použitelnost algoritmu již pro tréningovou množinu o 100 odehraných hrách. Strategie algoritmu DOO se velmi blížila ε-Nashově rovnováze. Se zvyšujícím se počtem dat tréningové množiny algoritmus DOO více využíval soupeře podobně jako u algoritmu ONO.

2.2.5 Implicitní Modelování

Algoritmus Implicitního Modelování (IM)[8] nejdříve vypočítá portfolio odpovědí a následně určí očekávanou výplatu každé z nich vzhledem k doposud nasbíraným informacím o strategii soupeře. IM tedy postupuje opačně než standardní



Portfolio			
	m1	*m2*	*m3*
o	7	2	-3

Obrázek 2: Rozdíl mezi modelováním soupeře běžným způsobem a algoritmem Implicitního Modelování. Levá část obrázku ukazuje uzel soupeře O, který volí akce s pravděpodobnostmi α pro akci zahození, β pro akci dorovnání a γ pro akci navýšení. Platí, že $\alpha+\beta+\gamma=1$. Pravá část je portfolio očekávaných výplat odpovědí $m1$, $m2$, $m3$ na soupeřovu strategii o.

způsoby modelování soupeře. Jakkoliv je složitá strategie soupeře, IM ji redukuje na portfolio, které má k dispozici.

Běžný postup pro modelování soupeře spočívá v pozorování jeho akcí a pamatování si pravděpodobností, s jakými akce provádí v uzlech herního stromu. IM místo toho pravděpodobnosti z uzlů sumarizuje jako očekávané hodnoty v portfoliu odpovědí (obr. 2). Odpověď v portfoliu reprezentuje strategii IM pokerbota.

Portfolio odpovědí je vygenerováno offline. Má-li pokerbot k dispozici informace o kompletní strategii soupeře, použije pro výpočet odpovědi Omezenou Nashovu Odpověď (ONO). Při malém množství dat je použita Datově Orientovaná Odpověď (DOO). ONO i DOO vygenerují robůstní odpovědi – kompromis mezi schopností využívat a limitování vlastní využivatelností.

Portfolio by mohlo obsahovat odpověď na každý dataset informací o soupeřově strategii. Autoři však poukazují na nutnost omezit velikost portfolia, protože čím větší velikost portfolia, tím je náročnější vybrat správnou odpověď.

Po výpočtu portfolia je pokerbot schopný adaptovat se soupeři v reálném čase. Potřebuje znát pouze očekávanou výplatu každé strategie v portfoliu. Zvolí strategii s nejvyšší očekávanou výplatou. Očekávanou výplatu strategií počítá v průběhu hry ze zpozorovaných akcí soupeře. K výpočtu dochází podobně jako v algoritmu Týmu expertů, zkoušením odpovědí z portfolia a zjišťováním, která má nejvyšší očekávanou výplatu.

Autoři otestovali algoritmus vytvořením pokerbotů Big-Portfolio a Small-Portfolio. Z odehraných her ze soutěže pokerbotů Annual Computer Poker Competition (ACPC) 2010 offline vygenerovali Big-Portfolio, obsahující odpověď pro každého z 12 soutěžících. Small-Portfolio obsahovalo 4 vybrané odpovědi. Oba pokerboti byli otestování proti soutěžícím z ACPC 2011, kde se Small-Portfolio umístil první a Big-Portfolio druhý. Výsledky ukazují, že algoritmus Implicitního Modelování je schopný efektivně hrát proti soupeřům, které nezná.

3 LAIN – Adaptivní pokerbot pro více hráčů

LAIN (limit Hold'em Artificial Intelligence) je adaptivní pokerbot varianty Texas Hold'em limit poker. Je navržen pro hru tří hráčů, dokáže však hrát hru $n \geq 2$ hráčů.

V kapitole popíši zobecnění algoritmů Miximax a Miximix pro hru více hráčů. Zavedu abstrakce pro procházení herního stromu a ukládání informací o soupeřích. Pokerboty, ze kterých vycházím a jejich principy jsem shrnul v kapitole 2.1.3.

3.1 Zobecnění algoritmů Miximax a Miximix

Hlavním rozdílem v pokeru, kde jsou více než dva hráči, je nutnost vyhrát nad větším počtem soupeřů zároveň. To se odrazí v množství držených karet, které je hráč ochoten hrát. Například držená karetní kombinace 5♠5♡ je hratelná ve dvou hráčích, ale je velmi diskutabilní ve více hráčích. Karetní kombinace má malou pravděpodobnost na zlepšení s dalšími společnými kartami – pomůže pouze 5♣ a 5♢ z balíčku a je velká pravděpodobnost, že některý z mnoha soupeřů bude mít vyšší pár nebo složí silnější výherní kombinaci v průběhu hry.

S tím souvisí *hra z pozice*. Hráč, který je mezi prvními na řadě, bude hrát spíše ty silnější karetní kombinace, než hráč, který hraje mezi posledními. Jinak řečeno, pokud vím, že po mně bude hrát velké množství hráčů, musím počítat s tím, že můžu čelit velkému počtu sázek. Oproti tomu hráč na posledních pozicích ví, kolik hráčů před ním již vsadilo a kolik hráčů po něm ještě může sázet. Nevsadil-li před nim nikdo a po něm hraje již jen jeden hráč, z hry více hráčů se stává hra dvou hráčů a neatraktivní držené karty získají na síle, protože hráč musí porazit pouze jednoho soupeře.

V algoritmech Miximax a Miximix je potřeba zahrnout pravděpodobnost pokerbota na výhru proti více soupeřům. Připomeňme význam histogramu, kterým se rozumí pole hodnot uložené v listovém uzlu modelu soupeře – kontextového stromu trie. Stejně jako v původním algoritmu pole reprezentuje interval $\langle 0, 1\rangle$ a je rozdělené na 20 stejně velkých podintervalů. Do prvního intervalu spadají všechny výherní kombinace se sílou (HR) v rozmezí $\langle 0, 0.05)$, do druhého HR v rozmezí $\langle 0.05, 0.1)$, do posledního HR v rozmezí $\langle 0.95, 1\rangle$.

Původní vzorec pro vypočítání očekávané hodnoty (EV) listového uzlu je:

$$EV(u) = (P(vyhra) \cdot bank) - cena.$$

Platí, že ruka s vyšším HR porazí ruku s nižším HR. Pravděpodobnost výhry proti jednomu soupeři je:

$$P(vyhra) = \frac{HR_{mensi}}{HR_{celkem}}.$$

HR_{mensi} je počet HR v histogramu, které jsou menší než HR pokerbota. HR_{celkem} je celkový počet HR v histogramu.

Pravděpodobnost výhry proti n soupeřům vypočítám jako produkt pravděpodobností na výhru proti každému soupeři zvlášť. Předpokladem je uložení histogramu ruk pro každého soupeře v odpovídajícím listovém uzlu modelu soupeře.

$$P(vyhra) = \prod_{i=1}^{n} \frac{HR^i_{mensi}}{HR^i_{celkem}} = \prod_{i=1}^{n} P_i(vyhra) \tag{1}$$

Podobný způsob vypočítání pravděpodobnosti na výhru proti n soupeřům použil Davidson v [1] pro pokerbota Poki, který byl zmíněn v kapitole 2.1.1. Místo histogramů měl pro každého soupeře uloženou tabulku vah o 1326 hodnotách, která odpovídala všem možným drženým karetním kombinacím. Položka v poli reprezentovala pravděpodobnost, s jakou pokerbot věří, že soupeř drží konkrétní kombinaci karet. Davidson tabulky použil pro výpočet *HR* pokerbota proti každému soupeři. Vynásobením *HR* proti jednotlivým soupeřům získal šanci na výhru proti n soupeřům.

Výpočet očekávaných hodnot u uzlu šance, soupeřova rozhodovacího uzlu a vlastního uzlu pokerbota je stejný, jako v původních algoritmech Miximax a Miximix. Nezávisí na počtu hráčů.

Povšimněte si, že v důsledku ponechání výpočtu očekávané hodnoty vlastního uzlu, se při akci zahození karet pokerbotem výpočet zastaví a je vrácena očekávaná hodnota rovna zápornému počtu doposud vsazených žetonů. Výpočet po akci zahození je zastaven i přesto, že ve hře může být více než jeden aktivní soupeř. Jejich hra nemá vliv na očekávanou hodnotu z pohledu pokerbota.

3.2 Zvolení abstrakcí pro Miximax a Miximix

Počet uzlů herního stromu roste exponenciálně s přibývajícím počtem hráčů. Zabývám se proto pouze hrou tří hráčů, pro kterou jsem zvolil abstrakce. Standardně může hrát u pokerového stolu až deset hráčů. Musel bych ale použít velmi silné abstrakce, které by původní hru znehodnotily. V kapitole popíši použité abstrakce pro algoritmus Miximax hry tří hráčů. Abstrakce lze použít i pro algoritmus Miximix.

Stejně jako v původním algoritmu Miximax, jsou při průchodu vynechány větve, které mají pravděpodobnost na provedení 0 %.

3.2.1 Vynechání uzlů šance

První abstrakcí je vynechání uzlů šance, které korespondují s úplným vynecháním otáčení budoucích společných karet. Uzly šance jsem nahradil výpočtem síly výherní kombinace (*HR*). Na začátku každého sázkového kola je vypočítána *HR* pokerbota proti jednomu obecnému soupeři. Výsledné *HR* reprezentuje sílu výherní kombinace pokerbota proti soupeři, který hraje všechny možné startovní kombinace karet se stejnou pravděpodobností. Důsledkem tohoto přístupu je, že se pokerbotova *HR* nezlepší při průchodu herním stromem algoritmem Miximax.

Jediná hodnota *HR* se použije při výpočtu pravděpodobnosti na výhru pokerbota v listových uzlech. Pseudokód upraveného algoritmu Miximax (obr. 3).

V závislosti na sázkovém kole spuštění algoritmu Miximax, rozlišuji způsob výpočtu *HR*:

Preflop *HR* pro každou ze 1326 držených kombinací byl předvypočítán offline. Pokerbot v tabulce vyhledá svoji startovní kombinaci karet a získá *HR*. Offline výpočet je založený na datech převzatých z [9]. Data obsahují počty výher, proher a remíz pro každý abstrahovaný pár držených karet. Abstrahovaných kombinací je 169. Data tedy obsahují $169 \cdot 169$ položek. Jsou trojího typu: pár, karty v barvě a karty mimo barvu. Stejné kombinace jsem sečetl a zprůměroval. Zůstalo 169 unikátních abstrahovaných kombinací. U každé jsem standardně vypočítal $HR = \frac{vyhra + \frac{remiza}{2}}{vyhra + prohra + remiza}$. Nakonec jsem zpětně převedl 169 abstrahovaných kombinací na 1326 startovních kombinací.

Flop *HR* je určen v reálném čase z držených karet pokerbota, právě otočených společných karet a postupným otočením 47 karet sázkového kola turn. Stejně jako u sázkového kola preflop spočítá počet výher, proher, remíz proti všem možným drženým karetním kombinacím soupeře a dosadí do vzorce.

Turn Z držených karet, společných karet a postupným otáčením 46 karet sázkového kola river, se vypočítájí výhry, prohry, remízy proti všem možným drženým kartám soupeře.

River Zde se *HR* vypočítá z držených karet pokerbota, společných karet proti všem možným drženým kombinacím soupeře.

3.2.2 Redukce počtu povolených sázek

Druhá abstrakce redukuje počet povolených sázek na kolo ze čtyř na dvě. Musel jsem zavést mapování mezi reálnou hrou a abstraktní hrou. V reálné hře si pokerbot ukládá zpozorované akce a tvoří z nich sekvenci. Například pokud v sázkovém kole flop hráči provedou po sobě akce navýšení (r, hráč 1), navýšení (r, hráč 2), navýšení (r, hráč 3), dorovnání (c, hráč 1) a dorovnání (c, hráč 2), pokerbot bude mít uloženou sekvenci `rrrcc`. Taková sekvence v abstrahovaném prohledávání stromu neexistuje, protože obsahuje tři sázky. Mapuji ji na sekvenci `rrcc`. Všimněte si, že se z reálné hry mapuje více sekvencí na jednu sekvenci abstraktní hry. Jde tedy o jistou újmu na obecnosti – dochází ke ztrátě informace.

Abstrahovaná sekvence akcí se použije při volání algoritmu Miximax. Vytvoří se z ní část abstraktního stromu hry. Nad tímto stromem se spustí algoritmus Miximax, který projde strom až k listům a zpětnou propagací vrátí očekávané hodnoty kořenových akcí (prvních akcí nad kterými se spustil algoritmus). Akci s maximální hodnotou provede v reálné hře.

```
Action miximax(GameState gs){
  foreach action in gs.getActions(){
    gsClone = gs.clone();
    gsClone.doAction(action);
    EV[action] = miximaxRec(gsClone);
  }
  return maxAction(EV);
}
double miximaxRec(GameState gs){
  if(gs.isMyNode()){
    foreach action in gs.getActions(){
        gsClone = gs.clone();
        gsClone.doAction(action);
        EV[action] = miximaxRec(gsClone);
    }
      return maxEV(EV);
  }
  else if(gs.isOpponentNode()){
    foreach action in gs.getActions(){
      gsClone = gs.clone();
      gsClone.doAction(action);
      fr[action] = trie.getActionFrequency(gsClone);
      EV[action] = miximaxRec(gsClone);
    }
    return weightedSum(fr, EV);
  }
  else{
    if(gs.isShowdown()){
      pWin = trie.getWinPercentage(gsClone);
    }
    else if(gs.myFold){
      pWin = 0;
    }
    else{
      pWin = 1;
    }
    return (pWin * gs.bank) - gs.myBetsTotal;
  }
}
```

Obrázek 3: Pseudokód upraveného rekurzivního algoritmu Miximax. Oproti původnímu algoritmu zde chybí rozlišení uzlů šance, které odstranila abstrakce kvůli příliš velké rozvětvenosti stromu. Změněna je také funkce *getWinPercentage*, která vypočítá šanci na výhru proti všem soupeřům zároveň.

Mapování sekvence reálné hry na abstraktní se skládá ze čtyřech kroků:

Rozdělení na podsekvence Reálná sekvence `rrcc_rrrcc_crcrcrfc_cc` je rozdělena na podsekvence reprezentující sázkové kola: preflop je `rrcc`, flop `rrrcc`, turn `crcrcrfc`, river `cc`.

Mapování podsekvencí na abstraktní Každá podsekvence se prochází zleva doprava a kontroluje se počet provedených akcí navýšení (r). Pokud je provedeno více než dvě navýšení, akce se přepíše na dorovnání (c). Průchod končí dříve, pokud podsekvence dosáhne konce sázkového kola v abstraktní hře. Viz mapování `rrrcc` na `rrcc`: `r` na `r`, `r` na `r`, `r` na `c`, `c` na `c`, konec sázkového kola v abstraktní hře.

Zahrnutí akcí zahození Ve hře n hráčů může až $n-2$ hráčů provést akci zahození a zbylí 2 hrači budou stále pokračovat ve hře. Je potřeba proto upravit abstraktní podsekvence. Mapování podsekvence sázkového kola turn `crcrcrfc` je na abstraktní podsekvenci `crcrcc`. Abstraktní podsekvence nezahrnuje akci zahození hráče 1. V Abstraktní podsekvenci se zkouší dosadit akce zahození (f) na pozice hráče 1 zprava doleva: `crcrcc` má indexy hráčů 123123. První index hráče 1 z prava je na pozici 3 (číslováno od 0). Zkusí se dosadit f: `crcfcc`. Odstraní se část podsekvence zprava, která není platná podle abstraktní hry: `crcf`. Obdobně se bude postupovat pro každou další akci zahození. Jakmile jsou všechny akce zahození v abstraktní podsekvenci, konec.

Zřetězení abstraktních podsekvencí Jednotlivé abstraktní podsekvence se zřetězí. Výsledkem je abstraktní sekvence. V uvedeném příkladu to bude: `rrccrrcccrcfcc`.

Problémem mapování je, že běžně vytvoří abstraktní sekvenci, která nezachovává stejný stav hry jako reálná sekvence. V reálné sekvenci je pokerbot LAIN na tahu. Zavolá algoritmus Miximax s abstraktní sekvencí. V abstraktní sekvenci ale nastal konec kola. Miximax se nemůže provést. Příklad: reálná sekvence `rrrc` na abstraktní `rrcc`. Předpoklad: LAIN je na pozici 1. Nyní je na tahu v reálné hře, avšak v abstraktní hře je konec sázkového kola.

Nastane-li takový problém, LAIN volí akci dorovnání, protože nemůže zavolat algoritmus Miximax.

3.3 Modelování více soupeřů

Princip pokerbota, který modeluje soupeře pozorováním jejich akcí a odhalených karet, zůstává stejný. Změnit se musí způsob ukládání informací a podoba datové struktury trie, která je *kontextovým stromem* hry. V původním článku kontextový strom odpovídal hernímu stromu hry bez uzlů šance. Každý uzel reprezentoval jednu akci, v listech byl histogram soupeře.

3.3.1 Úprava kontextového stromu

V kontextovém stromu jsem zredukoval celkový počet povolených sázek ze čtyř na dvě. Upravený kontextový strom odpovídá hernímu stromu, který prohledává algoritmus Miximax.

Zpozoruje-li pokerbot akci některého ze soupeřů, provede mapování sekvence akcí z reálné hry do abstraktní hry. Výslednou abstraktní sekvenci po odehrání ruky uloží do kontextového stromu – zvýší počítadlo frekvencí na cestě kontextovým stromem. V případě příkladu z předchozí kapitoly, kde se sekvence `rrrcc` mapovala na sekvenci `rrcc`, se kontextový strom rekurzivně projde od kořene a s každým uzlem na cestě se zleva doprava konzumuje abstraktní sekvence. V navštívených uzlech je zvýšeno počítadlo frekvencí o 1. Pokud při průchodu kontextovým stromem dojde pokerbot do bodu, kde nejsou potomci vyhovující aktuální akci v sekvenci, začne rekurzivně vytvářet nové uzly.

Dojde-li k odhalení karet ve hře tří hráčů, mohou zde být až dva soupeři. Musím aktualizovat až dva histogramy v kontextovém stromu, které jsou ve stejném uzlu (odpovídá jim stejná sekvence). Pro každého soupeře, který odhalil držené karty, projdu rekurzivně kontextový strom dokud nenarazím na list. V listu je histogram pro každého hráče. Vypočítám sílu výherní kombinace (*HR*) soupeře, převedu ji na index do histogramu konkrétního hráče a zvýším počítadlo o 1.

Ze zpozorovaných informací o soupeřích si na konci odehrané ruky ukládám:

- *HR* všech soupeřů, kteří ukázali držené karty. Musí platit, že jsem neprovedl akci zahození.
- Abstraktní sekvenci, pokud jsem neprovedl akci zahození.
- Část abstraktní sekvence, provedl jsem-li akci zahození. Příklad ve hře tří hráčů, moje pozice je 0: `crrf`. Uložím si abstraktní sekvenci `crr`.

Neukládám si *HR* soupeřů, pokud jsem v průběhu ruky zahodil karty. Při průchodu herním stromem algoritmem Miximax neprohledávám větve, ve kterých jsem zahodil karty, protože mají $EV = -cena$, kde *cena* je kolik jsem měl v banku vsazeno celkem žetonů. Na EV akce zahození nemají histogramy soupeřů vliv.

V paměti si udržuji pro každého hráče jeden kontextový strom, pro hru tří hráčů to jsou tři stromy. Kontextové stromy jsou rozlišeny podle začínajícího hráče. Sedím-li na pozici číslo 0, kontextový strom číslo 0 bude strom, kde začínám já. Při prohledávání Miximaxem se využívá kontextový strom hráče, který začal ruku. Ve hře tří hráčů je to dealer.

V kontrastu s autory původních pokerbotů Vexbot a BRPlayer (kap. 2.1.3) jsem nezavedl žádné metody agregace zpozorovaných informací pro rychlejší učení. Jejich motivace byla především, aby pokerbot dokázal hrát proti lidskému soupeři, který často odehraje jen několik set her. Ve hře proti počítačovým hráčům, kde je provedeno testování LAIN, nejsem vystaven tomuto omezení.

3.3.2 Výchozí model

Volba výchozího modelu je důležitá, aby pokerbot LAIN příliš neprohrával na začátku hry. Se špatným počátečním modelem budou *EV* akcí LAIN chybně vypočítána vzhledem k zatím neznámé strategii soupeře. LAIN může sázet příliš agresivně nebo příliš pasivně. Problém jsem vyřešil řídkým modelem obsahujícím pouze nezbytné minimum dat. Důvodem je, aby nově naučená data o soupeřích měla velký vliv na podobu modelu co nejdříve.

Na začátku hry je kontextový strom prázdný – je potřeba ho naplnit výchozími daty. Zvolil jsem velmi jednoduché řešení, které plní kontextový strom při spuštění Miximaxu. Pokud potřebuje Miximax prohledat uzly, které nejsou v kontextovém stromu, vytvoří se s výchozími daty.

Výchozí data nastavují frekvenci každého uzlu akce dorovnání na 1. Ostatní uzly budou mít frekvenci 0. V důsledku na začátku hry mají uzly akce dorovnání pravděpodobnost na provedení 100 %. Předpokládám tedy, že mě soupeři vždy dorovnají.

Do listových uzlů jsou uloženy výchozí histogramy pro každého hráče. Naplnění histogramu je zvoleno následovně: $[0 \ \ldots \ 0\,1\,1\,0 \ \ldots \ 0]$. Jednou je zastoupena $HR \in \langle 0.45, 0.5)$ a jednou $HR \in \langle 0.5, 0.55)$. Předpokládejme, že LAIN má sílu výherní kombinace $HR = 0.5$. V nově vytvořeném uzlu, kde jsou přítomni oba soupeři, vypočítá svoji pravděpodobnost na výhru jako $P(win) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$, protože porazí polovinu *HR* z každého histogramu.

Výchozí model je velmi hrubý. Předpokladem však je hra proti počítačovým hráčům, která bude trvat desítky tisíc ruk. Výchozí model bude nakonec naplněn relevantními daty.

3.3.3 Pravidlo skrytí držených karet

V některých pokerových pravidlech je povoleno skrýt držené karty na konci hry, pokud je jisté, že hráč prohrál. LAIN dokáže hrát poker i s tímto pravidlem. V kapitole uvedu příklady situací se skrytím držených karet a popíši své řešení.

Příklad 1. Uvažujme konec hry – showdown. Hráči si mají porovnat výherní kombinace, nejsilnější vyhrává bank. Ve hře jsou hráč A, hráč B, hráč C. V abecedním pořadí ukazují držené karty. Hráč A ukáže výherní kombinaci s $HR = 0.9$. Hráč B má výherní kombinaci $HR = 0.8$. Hráč C má $HR = 0.5$. Protože platí pravidlo skrytí držených karet, hráči B a C bez ukázání zahodí držené karty.

Mohou nastat dva případy:

1. LAIN je hráč A. Do kontextového stromu si nemůže uložit žádné *HR* soupeře.

2. LAIN je hráč B nebo C. Uloží si *HR* hráče A.

```
void increaseHRUniformly(double winnerHR, int loserSeat){
    winnerIndex = histograms.getIndex(winnerHR);
    fraction = 1 / winnerIndex;
    for(int i = 0; i < winnerIndex; i++){
      histograms[loserSeat][i] += fraction;
    }
}
```

Obrázek 4: Řešení neznámé síly výherní kombinace (HR) poraženého hráče (loserSeat). Dostupná je pouze síla výherní kombinace vítěze (winnerHR). Poražený hráč má HR v intervalu $I = \langle 0, winnerHR)$. Do intevalu I histogramu hráče loserSeat je rovnoměrně rozloženo číslo 1 reprezentující navýšení neznámého HR.

Příklad 2. Stejná herní situace. Hráč A ukazuje $HR = 0.8$. Hráč B $HR = 0.9$. Hráč C $HR = 0.5$. Hráč C bez ukázání zahazuje karty.

Tři možné případy:

1. LAIN je hráč A. Do kontextového stromu uloží HR hráče B.
2. LAIN je hráč B. Uloží si HR hráče A.
3. LAIN je hráč C. Uloží si HR hráčů A a B.

Jediná jistota je, že LAIN uvidí sílu výherní kombinace vítěze (HRv) a bude vědět, kteří hráči prohráli s menší sílou výherní kombinace (HRp). Vhodné je ukládat HRp, jinak hrozí zaplnění histogramů soupeřů převážně vysokými HR. V důsledku by pokerbot LAIN čím dál více zahazoval držené karty, protože by nebyly dost silné.

Řešení. Pro každého soupeře, který skryl držené karty vypočítám interval v histogramu $I = \langle 0, HRv)$ do jakého jeho HRp patří. Neznáme HRp reprezentuji číslem 1 a rozložím ho mezi podintervaly intervalu I. Pokud v I je méně než 10 uložených HR, provedu rovnoměrné rozložení (obr. 4).

Pro počet uložených HR v I větší nebo rovno 10 provedu proporcionální rozložení vzhledem k již uloženým HR (obr. 5).

```
void increaseHRProportionally(double winnerHR, int loserSeat){
    winnerIndex = histograms.getIndex(winnerHR);
    HRcount = histograms[loserSeat].lowerHRCount(winnerIndex);
    double fraction = 1 / HRcount;
    for(int i = 0; i < winnerIndex; i++){
      update = (histograms[loserSeat][i] * fraction);
      histograms[loserSeat][i] += update;
    }
}
```

Obrázek 5: Řešení neznámé síly výherní kombinace (*HR*) poraženého hráče (loserSeat). Dostupná je pouze síla výherní kombinace vítěze (winnerHR). Poražený hráč má *HR* v intervalu $I = \langle 0, winnerHR)$. Do intevalu I histogramu hráče loserSeat je proporcionálně rozloženo číslo 1 reprezentující navýšení neznámého *HR* vzhledem k počtu již uložených *HR*.

4 Testování a výsledky

V první části kapitoly se věnují tomu, jaké jsem použil abstrakce na redukci počtu uzlů herního stromu. Hledal jsem abstrakci vhodnout pro hru tří hráčů. V druhé části popíši testy pokerbota LAIN v prostředích `opentestbed` a `Poker Academy Pro 2.5`.

4.1 Abstrakce v praxi

Spočítal jsem čtyři varianty abstrakcí redukující počet uzlů kontextového stromu trie, abych zvolil vhodnou pro pokerbota LAIN. Všechny tabulky zobrazují kolik listových uzlů s histogramy (dále jen listy) je v kontextovém stromu pro 2 až 10 hráčů. Cílem je najít kompromis mezi co nejmenším počtem listů a zachováním vlastností původní hry. Čím méně listů, tím rychleji se LAIN učí. S příliš silnými abstrakcemi ale nebude LAIN schopná využívat soupeře. Například pokud bych snížil počet sázek na 0, LAIN by znala pouze akce zahození a dorovnání.

Pokerboti Vexbot a BRPlayer, na nichž je LAIN založená, využívají kontextový strom s přibližně $6.5 \cdot 10^3$ listy. Pro každého hráče potřebují jeden. Musí tedy uvažovat přibližně $1.3 \cdot 10^4$ listů. Poznamenejme, že jejich kontextový strom nemá implicitně zahrnuté povinné sázky hráčů small blind a big blind v sázkovém kole preflop. V tabulkách v této kapitole jsem povinné sázky zahrnul.

Celkový počet listů, které LAIN musí uvažovat, je počet listů v kontextovém stromu krát počet hráčů.

V kapitole jsou počty listů zaokrouhlovány.

Kontextový strom bez abstrakcí má rozmezí 10^3 listů pro 2 hráče a až 10^{21} lístů pro 10 hráčů (tab. 2). Pro tři hráče je to 10^7 listů. Jsou započítány i listy, ve kterých jsou pouze soupeři – pokerbot zahodil karty.

Tabulka 2: Počet listů s histogramy v kontextovém stromu bez použití abstrakcí. Započítány jsou i listy, kde není pokerbot.

Hráči	**Listy s histogramy**
2	$5 \cdot 10^3$
3	$10 \cdot 10^6$
4	$5.5 \cdot 10^9$
5	$1.3 \cdot 10^{12}$
6	$1.8 \cdot 10^{14}$
7	$1.7 \cdot 10^{16}$
8	$1.3 \cdot 10^{18}$
9	$8 \cdot 10^{19}$
10	$4 \cdot 10^{21}$

Tabulka 3: Počet listů s histogramy v kontextovém stromu. Abstrakce vynechala listy, kde není pokerbot.

Hráči	**Listy s histogramy**
2	$5 \cdot 10^3$
3	$7.6 \cdot 10^6$
4	$3.4 \cdot 10^9$
5	$7 \cdot 10^{11}$
6	$8.7 \cdot 10^{13}$
7	$7.6 \cdot 10^{15}$
8	$5 \cdot 10^{17}$
9	$2.9 \cdot 10^{19}$
10	$1.4 \cdot 10^{21}$

Ve všech ostatních testech vynechávám listy, ve kterých není přítomen pokerbot LAIN. Jde o formu abstrakce. Oproti předchozímu výsledku bez abstrakce je počet listových uzlů snížen přibližně o třetinu (tab. 3).

O poznání větší úbytek listových uzlů s histogramy jsem zaznamenal po zavedení abstrakce redukující počet povolených sázek v každém sázkovém kole ze 4 na 3 (tab. 4). Počet listů pro tři hráče je $4.3 \cdot 10^5$.

Další abstrakce redukuje počet sázek na 2. Počet listů pro tři hráče je $2 \cdot 10^4$ (tab. 5). Pro čtyři je to $6.7 \cdot 10^5$.

Poslední abstrakcí, kterou jsem testoval, je snížení počtu sázek na 1 (tab. 6). Pro tři hráče bude mít kontextový strom 450 listů s histogramy. Pro čtyři hráče je to ~5500 a pět hráčů ~59000.

Tabulka 4: Počet listů s histogramy v kontextovém stromu po použití abstrakce snižující počet povolených sázek na kolo na 3.

Hráči	**Listy s histogramy**
2	$1.7 \cdot 10^3$
3	$4.3 \cdot 10^5$
4	$4.9 \cdot 10^7$
5	$3.4 \cdot 10^9$
6	$1.8 \cdot 10^{11}$
7	$7.4 \cdot 10^{12}$
8	$2.6 \cdot 10^{14}$
9	$8.3 \cdot 10^{15}$
10	$2.3 \cdot 10^{17}$

Tabulka 5: Počet listů s histogramy v kontextovém stromu po použití abstrakce snižující počet povolených sázek na kolo na 2.

Hráči	**Listy s histogramy**
2	$3.7 \cdot 10^2$
3	$2 \cdot 10^4$
4	$6.7 \cdot 10^5$
5	$1.7 \cdot 10^7$
6	$3.7 \cdot 10^8$
7	$7.3 \cdot 10^9$
8	$1.3 \cdot 10^{11}$
9	$2.2 \cdot 10^{12}$
10	$3.5 \cdot 10^{13}$

Tabulka 6: Počet listů s histogramy v kontextovém stromu po použití abstrakce snižující počet povolených sázek na kolo na 1.

Hráči	**Listy s histogramy**
2	$2.7 \cdot 10^1$
3	$4.5 \cdot 10^2$
4	$5.5 \cdot 10^3$
5	$5.9 \cdot 10^4$
6	$5.8 \cdot 10^5$
7	$5.5 \cdot 10^6$
8	$5 \cdot 10^7$
9	$4.4 \cdot 10^8$
10	$3.9 \cdot 10^9$

4.2 Testování pokerbota

Pokerbota LAIN jsem otestoval v prostředích `opentestbed` a `Poker Academy Pro 2.5` proti různým pokerbotům. LAIN zvítězila ve všech partiích. Grafy v kapitole jsou výstupy z `opentestbed` a `Poker Academy Pro 2.5`. V závěru kapitoly otestuji řešení pravidla skrytí držených karet (kap. 3.3.3) a ukáži, že je LAIN schopná hrát hru čtyřech hráčů, přestože pro to nejsou její abstrakce navrženy.

Výkon LAIN hodnotím jako počet vyhraných malých sázek (sb) na odehranou ruku (h). Malá sázka je nastavena na 1$.

LAIN byla napsána nad rozhraním `meerkat-api`, které definuje, jak má vypadat pokerbot. `Opentestbed` i `Poker Academy Pro 2.5` používají `meerkat-api` pro definici pokerbotů.

4.2.1 Opentestbed

`Opentestbed` je freeware testovací prostředí pro vzájemnou hru pokerbotů. Umožňuje nastavit mimo jiné kolik se má odehrát ruk, výši sázek a jací pokerboti mají být ve hře. V každém testování pokerboti odehráli 250000 ruk. `Opentestbed` nemá grafické rozhraní, výstupem programu je graf a historie odehraných ruk.

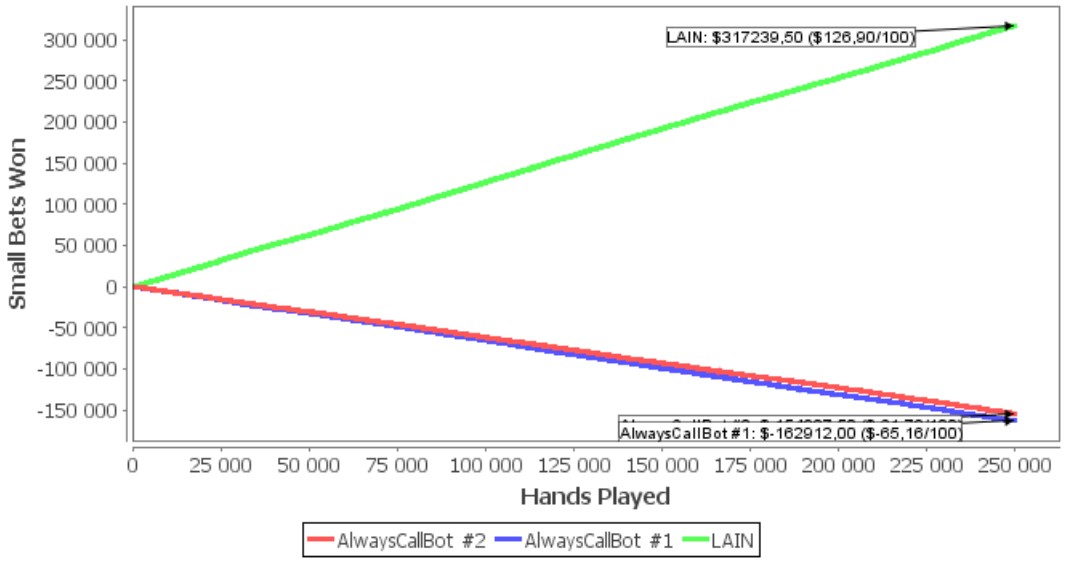
Pokerboti z `opentestbed`, proti kterým byla LAIN postupně testována:

AlwaysCallBot Vždy dorovná sázku, pokud může. Pokud nemůže, tak čeká. Hraje tedy všechny ruky, které jsou mu rozdány.

AlwaysRaiseBot Vždy navýší sázku. Nemůže-li, dorovná sázku nebo čeká. Hraje všechny ruky.

SimpleBot Rozlišuje dva druhy strategie: jednu pro preflop a druhou pro pozdější sázkové kola (tzv. postflop).

- **preflop** Používá seznam pravidel, kdy hrát jaké držené karty. Se silnými drženými kartami navyšuje, se středními dorovnává, se slabými zahazuje. Pokerbot blafuje s párem 22 a kombinací $A2$ ve stejné barvě – navyšuje s nimi jako se silnou kombinací. Nevyhovělo-li žádné dřívější pravidlo, v 5 % případů dorovná s libovolnou drženou kombinací – taktéž jde o blaf.
- **postflop** Řídí se rovnicemi, kterými vypočítá: sílu své výherní kombinace (HR), potenciál na zlepšení výherní kombinace ($ppot$) a návratnost investice (po). Na začátku vygeneruje náhodné číslo x, které zajišťuje, že se pokerbot nebude s vysokou pravděpodobností chovat stejně ve stejných herních situacích. Pokud nečelí sázce, při $x < HR^2$ navýší. Při $x < ppot$ také navýší. V ostatní případech čeká. Čelí-li sázce, navýší s $x < HR^{1+pocetSazek}$. Dorovná když $\Big(\Big(HR^2 \cdot bank\Big) > d\Big) \vee (ppot > po)$.



Obrázek 6: Graf pokerbota LAIN proti dvěma AlwaysCallBot.

SimpleBotNoBluff Deterministická varianta pokerbota Simplebot. Pokerbot neblafuje a x je konstanta nastavená na 0.5.

V první partii hrál proti dvěma AlwaysCallBot (obr. 6). Zvítězil s výplatou 1.269 sb/h.

Ve druhé partii hrál proti dvěma AlwaysRaiseBot (obr. 7). Zvítězil s výplatou 1.9349 sb/h.

Ve třetí partii hrál proti dvěma Simplebot (obr. 8). Zvítězil s výplatou 0.3874 sb/h.

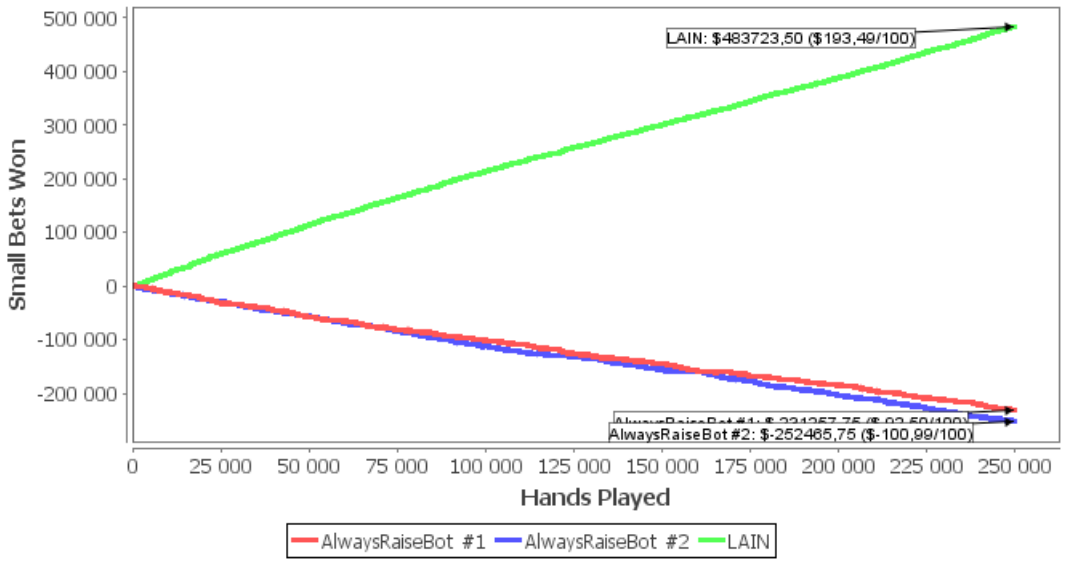
Ve čtvrté partii hrál proti dvěma SimpleBotNoBluff (obr. 9). Zvítězil s výplatou 0.5527 sb/h. Ve zkráceném úseku čítajícím 30000 odehraných ruk, je vidět učení se pokerbota LAIN (obr. 10). Přibližně 1500 odehraných ruk ji trvalo vytvořit stabilní model obou soupeřů. Po tomto bodě hry se její výplata zvyšovala pouze mírně.

V páté partii hrál proti jednomu SimpleBot a jednomu SimpleBotNoBluff (obr. 11). Zvítězil s výplatou 0.4491 sb/h.

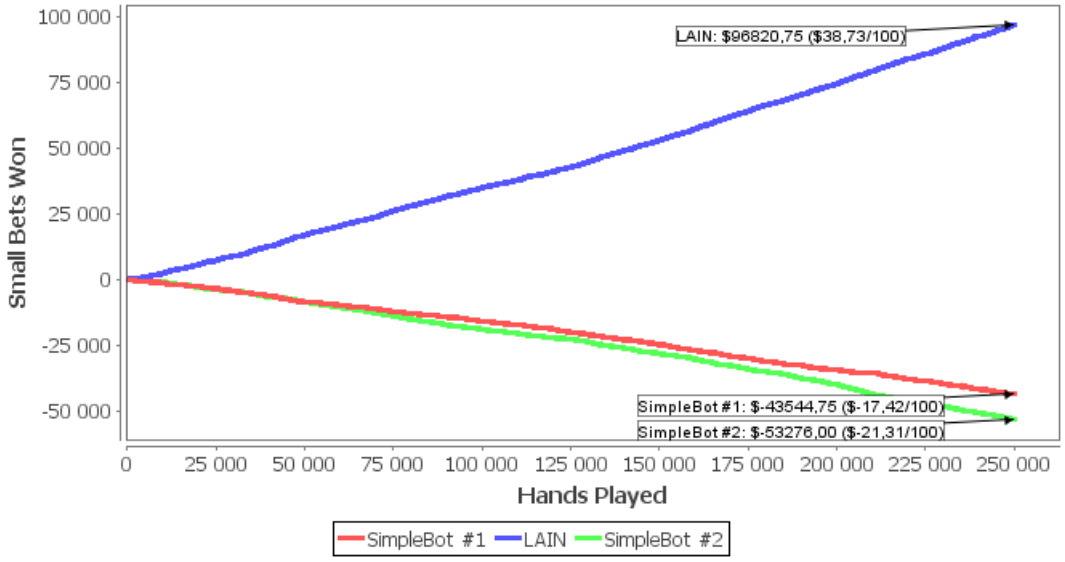
4.2.2 Poker Academy Pro 2.5

`Poker Academy Pro 2.5` je komerční program pro hru lidských a počítačových hráčů. Jeho součástí jsou pokerboti hry více hráčů Poki (kap. 2.1.1), Simbot (kap. 2.1.2 a Jagbot. Z trojice pokerbotů je Poki považován za nejsilnějšího. Protože Poker Academy používá uživatelské rozhraní, hry pokerbotů trvají podstatně déle než u prostředí `opentestbed`.

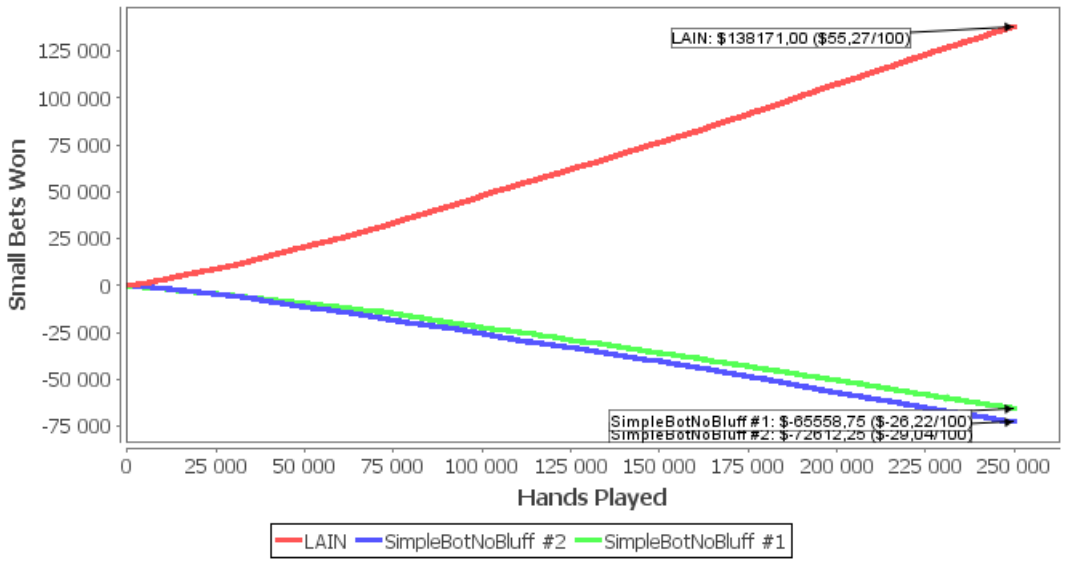
LAIN byla otestována ve dvou partiích. V každé bylo odehráno 30000 ruk.



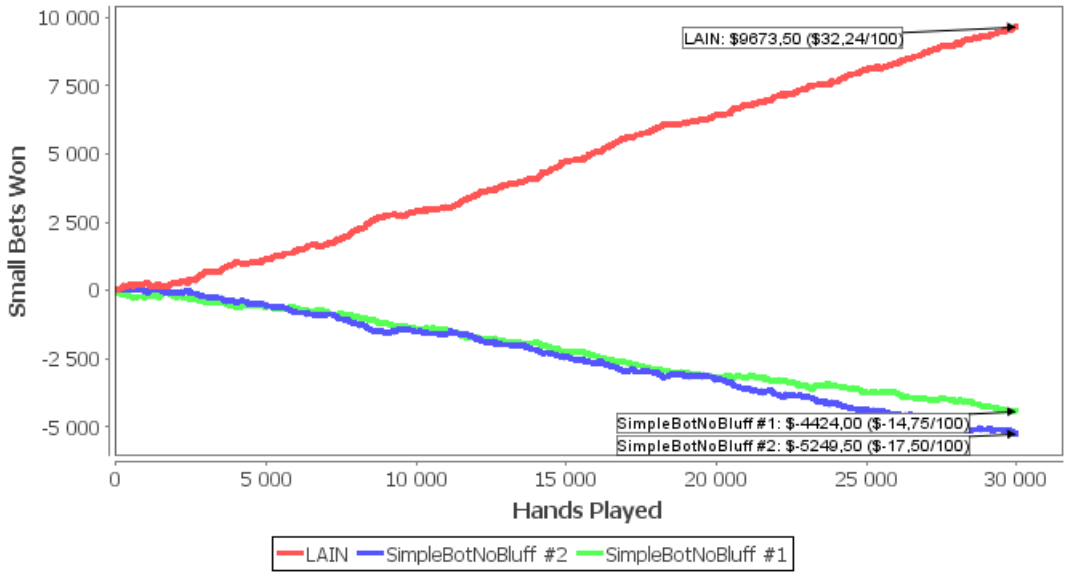
Obrázek 7: Graf pokerbota LAIN proti dvěma AlwaysRaiseBot.



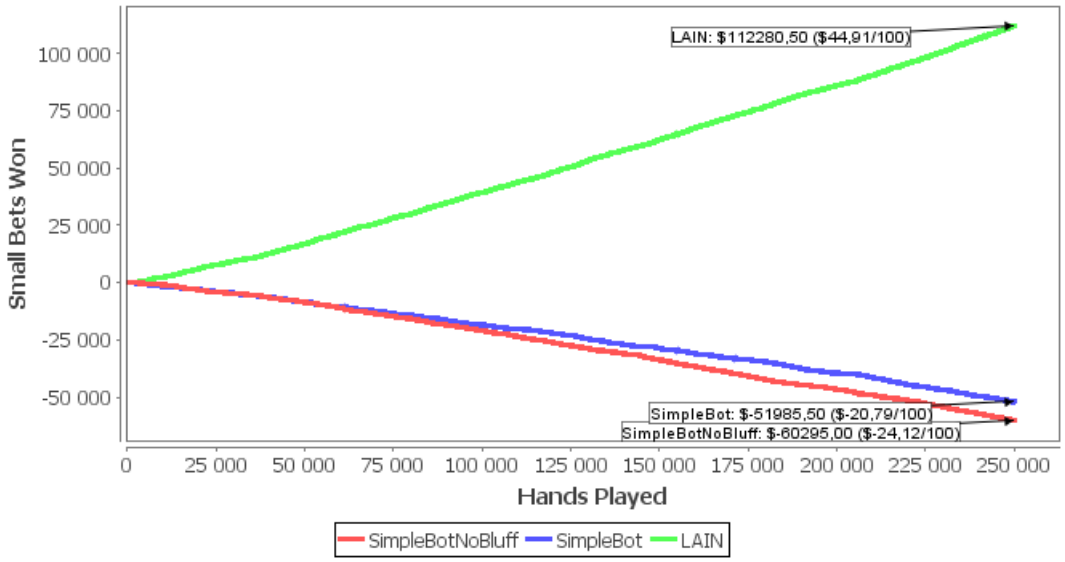
Obrázek 8: Graf pokerbota LAIN proti dvěma Simplebot.



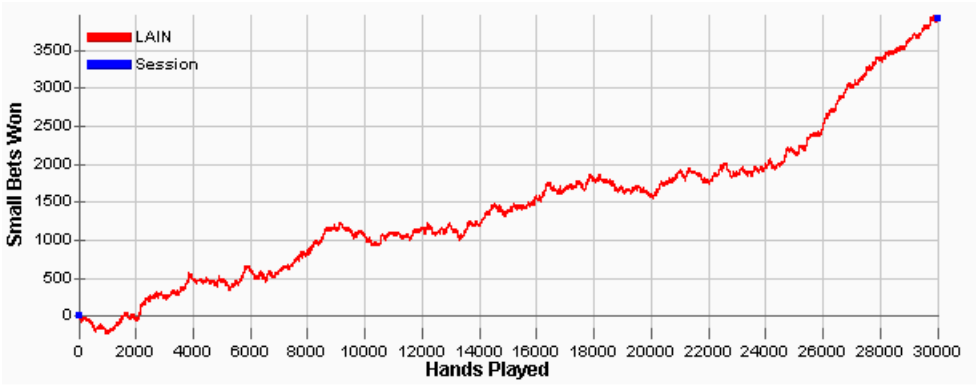
Obrázek 9: Graf pokerbota LAIN proti dvěma SimpleBotNoBluff.



Obrázek 10: Graf pokerbota LAIN proti dvěma SimpleBotNoBluff – úsek 30000 odehraných ruk. LAIN trvalo přibližně 1500 ruk, než vytvořila stabilní model obou soupeřů.



Obrázek 11: Graf pokerbota LAIN proti jednomu SimpleBot a jednomu SimpleBotNoBluff.



Obrázek 12: Graf LAIN ve hře proti dvěma pokerbotům Poki.

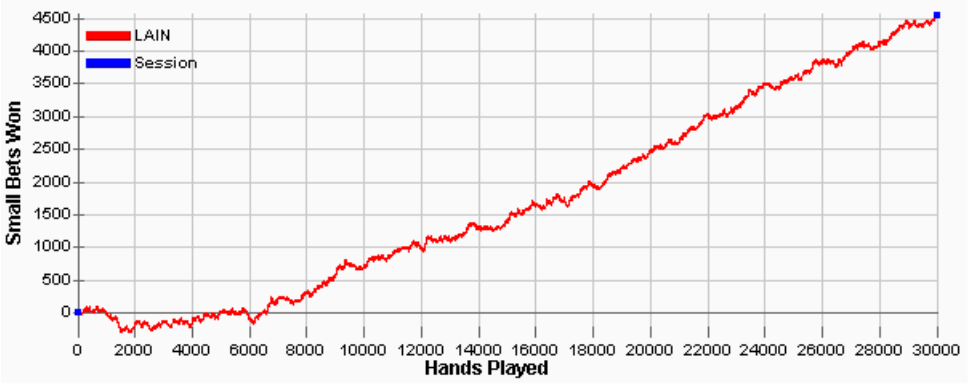
Poki Pokerbot založený na rovnicích. Využívající modelování soupeře.

Simbot Reimplementace pokerbota Poki postavená na Monte Carlo simulaci. Modeluje soupeře.

Jagbot Založený na pravidlech, které určují, jak hrát s konkrétními drženými kartami. Nepoužívá model soupeře.

První partii LAIN odehrála proti dvěma pokerbotům Poki (obr. 12). Zvítězila s výplatou 0.131 sb/h.

Ve hře proti pokerbotům Simbot a Jagbot LAIN zvítězila s výplatou 0.15 sb/h (obr. 13). Přes 6000 odehraných ruk trvalo, než LAIN začala konzistentně vítězit.



Obrázek 13: Graf LAIN ve hře proti pokerbotům Simbot a Jagbot.

4.2.3 Hra čtyř hráčů

V kapitole testuji LAIN ve hře čtyř hráčů. Abstrakce pro LAIN byly navrženy pro hru tří hráčů. V důsledku to znamená, že vytvoření modelu soupeřů bude LAIN trvat tím déle, čím více hráčů ve hře bude.

Ve hře tří hráčů LAIN rozlišuje $6 \cdot 10^4$ listových uzlů s histogramy soupeřů. Ve hře čtyř hráčů to je $2.68 \cdot 10^6$ listových uzlů.

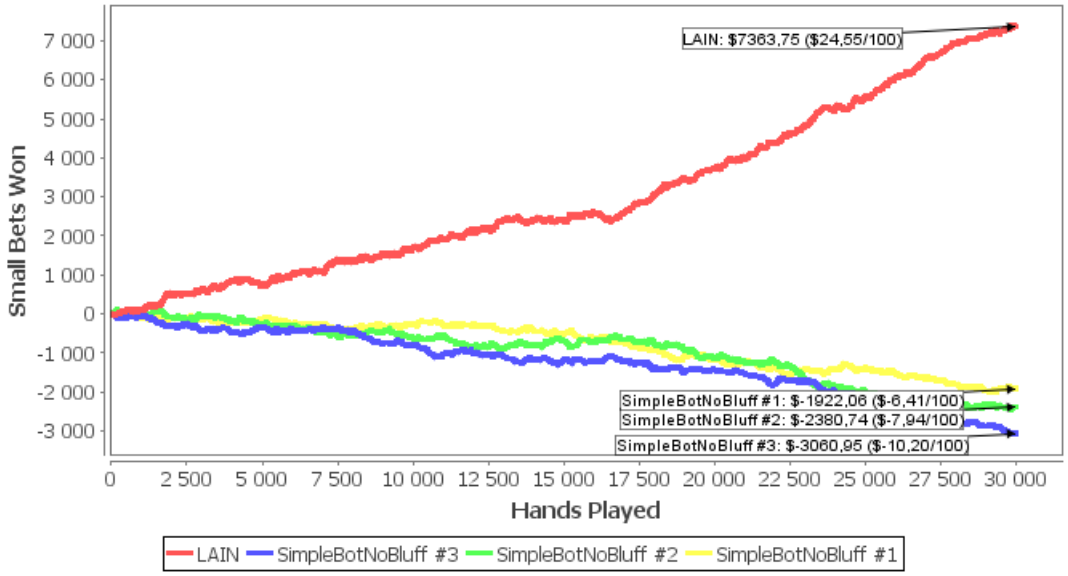
LAIN ve hře s třemi SimpleBotNoBluff pokerboty zvítězila po 30000 odehraných rukách s výplatou 0.2455 sb/h (obr. 14).

4.2.4 Pravidlo skrytí držených karet

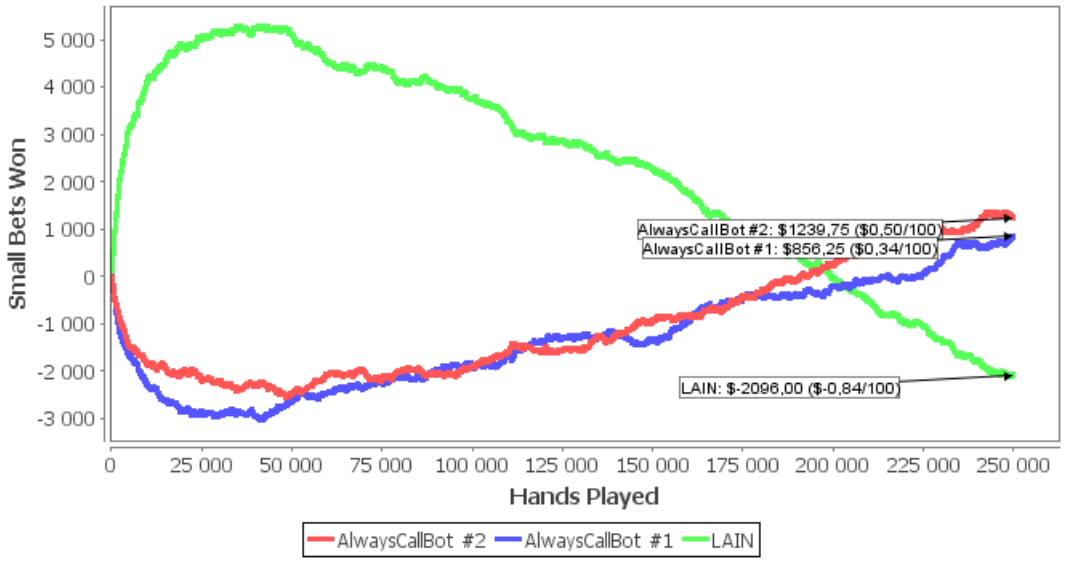
V kapitole testuji navrhované řešení (kap. 3.3.3) pro pravidlo skrytí držených karet. Připomeňme, že pravidlo skrytí držených karet umožňuje hráči na konci ruky neukázat držené karty pokud prohrál.

Test je proveden proti dvojici jednoduchých pokerbotů AlwaysCallBot. Pokud LAIN nemá ošetřeno pravidlo skrytí držených karet, její strategie zdegeneruje (obr. 15). Přibližně po odehrání 45000 ruk výplata LAIN přestala růst a začala klesat. Po odehrání 250000 ruk LAIN prohrála s výplatou -0.0084 sb/h.

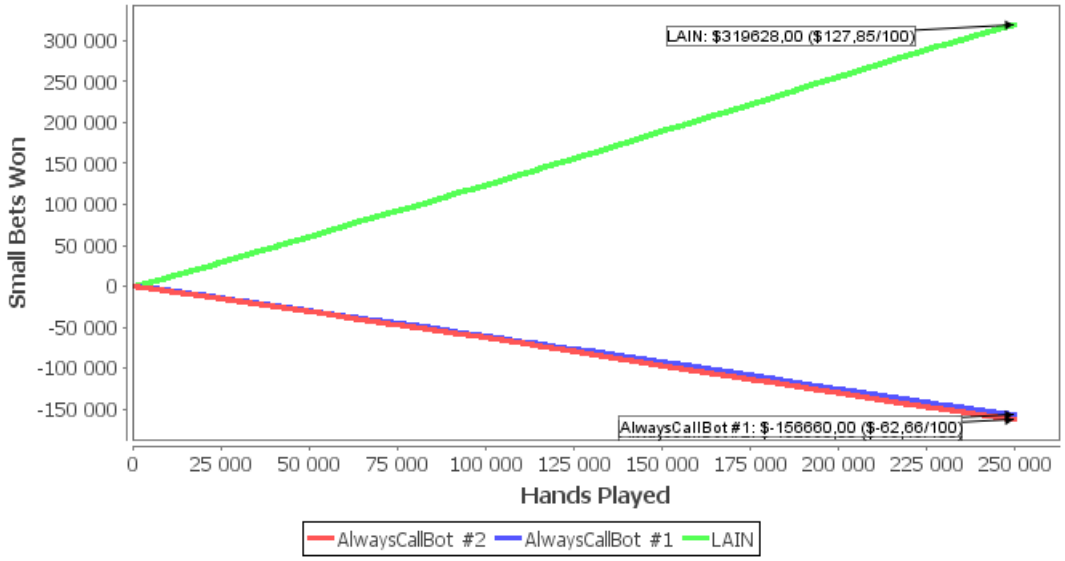
Se zavedením techniky rozložení neznámé síly soupeřovy výherní kombinace do jeho histogramu strategie LAIN nezdegeneruje (obr. 16). LAIN zvítězila s výplatou 1.2785 sb/h.



Obrázek 14: Graf pokerbota LAIN proti třem SimpleBotNoBluff – úsek 30000 odehraných ruk. LAIN trvalo přibližně 2000 ruk, než vytvořila stabilní model obou soupeřů. Až po 16000 odehraných rukách ale zaznamenala podstatně vyšší výplatu.



Obrázek 15: Graf LAIN proti dvěma AlwaysCallBot. Ve hře bylo povoleno pravidlo skrytí držených karet na konci hry. Strategie LAIN zdegenerovala.



Obrázek 16: Graf LAIN proti dvěma pokerbotům AlwaysCallBot. Ve hře bylo povoleno pravidlo skrytí držených karet na konci hry. LAIN používá metodu rozložení neznámé síly výherní kombinace soupeře do jeho histogramu.

Závěr

Hlavním kritériem pro výběr abstrakcí pokerbota LAIN byl kompromis mezi rychlostí přizpůsobení se soupeřům a zachování vlastností původní hry. Z uvedených abstrakcí nejlépe vycházela abstrakce redukující počet sázek v sázkovém kole ze čtyř na dvě. Po použití abstrakce má kontextový strom přibližně 20000 listových uzlů s histogramy soupeřů. Pro hru tří hráčů tedy LAIN rozlišuje 60000 herních kontextů. Pokerboti Vexbot[2] a BRPlayer[3] pro hru dvou hráčů rozlišují přibližně 13000 kontextů, neuvažujeme-li metody pro rychlejší učení, které sdružují podobné herní kontexty. Ve hře tří hráčů bude LAIN trvat téměř pětkrát delší dobu, než se plně přizpůsobí soupeřům, kteří by svou strategií využívali všechny její kontexty.

Testování proti jednoduchým pokerbotům z `opentestbed` ukazuje, že LAIN je schopna rychle vytvořit model soupeře. Po vytvoření uvodního modelu se s každou další odehranou rukou výplata LAIN zlepšuje pouze mírně. Tento trend ukazuje na jednoduchost strategie soupeřů. LAIN nemá možnost modelování příliš zdokonalit.

Například proti dvojici deterministických pokerbotů SimpleBotNoBluff je jejich model vytvořen za 1500 odehraných ruk (obr. 10).

Z grafu hry LAIN proti dvojici pokerbotů Poki z `Poker Academy Pro 2.5` je jasně patrné, že jde o silnější soupeře. Přibližně 1000 odehraných ruk trvalo, než LAIN přestala prohrávat. Poté graf mírně roste nebo stagnuje až do hranice 24000 odehraných ruk. Stagnace může naznačovat pouze částečně vytvořený model soupeřů a nebo schopnost pokerbotů Poki modelovat LAIN. Připomeňme, že Poki využívá modelu soupeře, aby lépe určil, jaké držené karty soupeř hraje.

V úseku od 24000 do 30000 odehraných ruk výplata LAIN výrazně roste, což ukazuje na vylepšení jejího modelu soupeřů.

Poker je hra vysoké variance, která značně ovlivňuje výsledek hry. Pokud ale výplata LAIN v delších odehraných usecích roste nebo nanejvýš stagnuje, potvrzuje to schopnost LAIN přizpůsobit se soupeřům a vytvoři platný model.

Hra proti pokerbotům Simbot a Jagbot se ukázala být ze začátku náročnější, než jsem předpokládal. LAIN začala konzistentně vítězit až za 6000 odehraných ruk. Přisuzuji to vzájemné strategické odlišnosti pokerbotů Simbot a Jagbot.

Krátké testování ve hře čtyř hráčů (kap. 4.2.3) ukazuje schopnost LAIN úspěšně hrát hru více než tří hráčů. Protože ale použité abstrakce nebyly navrženy pro hru více než tří hráčů, je nutné očekávat tím delší dobu učení, čím více soupeřů je přítomno a čím složitější hrají strategii.

Ve hře proti třem pokerbotům SimpleBotNoBluff, LAIN vytvoření dobrého modelu soupeřů trvalo přes 16000 ruk. Dobrým modelem rozumíme takový, který již nejde příliš zlepšit.

Po zavedení metody rozložení neznámé síly výherní kombinace soupeře do jeho histogramu (kap. 3.3.3), je LAIN schopná vytvořit model soupeřů i ve hře

s pravidlem skrytí držených karet. Pravidlo přidává další neúplné informace do hry pokeru, protože hráč najednou nemá jistotu, že na konci odehrané ruky uvidí držené karty soupeře. V živém pokeru je pravidlo často ve hře přítomné, což byla hlavní motivace pro prozkoumání tohoto problému. Autor přiznává, že je v tomto směru potřeba provést více testování, zejména proti složitým pokerbotům.

Jak LAIN dále vylepšit:

- Zavedení abstrakcí, které se automaticky přizpůsobí počtu hráčů a zároveň zachovají relevantní informace o hře.
- Abstrakce, které zachovají původní počet 4 sázek na sázkového kolo, aby se zdokonalila schopnost LAIN využívat strategie soupeře.
- Ukládání si síly výherní kombinace soupeřů (HR) i přesto, že LAIN provedla akci zahození. Aktuální LAIN této informace nedokáže využít, protože neprochází větvě herního stromu, ve kterých provedl akci zahození. S vyšším počtem hráčů ve hře je přitom zahazování čím dál běžnější.
- Použití technik seskupování histogramů, za účelem rychlejšího učení, které zavedli autoři pokerbotů Vexbot a BRPlayer.

A Výkladový slovník pojmů

Akce Jsou *čekání*, *dorovnání*, *zahození*, *vsazení* a *navýšení*.

Barva Rozlišovací znak na kartě: ♢♡♠♣ nebo *výherní kombinace* s pěti kartami se stejným znakem.

Balíček Sada 52 karet určených ke hře.

Bank Všechny žetony, které byly vsazeny v průběhu *ruky*.

Big blind Hráč, který na začátku *ruky* vloží do *banku* povinnou sázku velikosti *malé sázky*.

Blaf Hráč sází s *výherní kombinací*, která nemá šanci na výhru.

Čekání Hráč nic nedělá. Volba čekání je aktivní pouze pokud hráč nečelí sázce.

Čekání-navýšení Strategie ve které hráč očekává po své akci *čekání* nebo *dorovnání* soupeřovo *navýšení*, aby sám také navýšil.

Dealer Hráč po jehož levici je *small-blind*. V *postflopu* je posledním hráčem na tahu.

Dorovnání Dorovnání sázky, které hráč čelí.

Držené karty Dvě karty, které hráč dostal na začátku *ruky*. Zná je pouze on.

Flop Druhé *sázkové kolo*. Jsou otočeny tři *společné karty*. Hráči mohou sázet.

Kicker Karty doplňující *výherní kombinaci*, je-li ve výherní kombinaci méně než pět karet. Vybírány ze společných a držených karet hráče.

Malá sázka Určuje velikost sázky v *sázkových kolech*.

Maskování Zakrývání skutečné síly *výherní kombinace* pasivní hrou – hráč převážně jen *dorovnává*.

Návratnost investice Kolik nejméně musí mít hráč pravděpodobnost na výhru, aby se mu vyplatilo *dorovnat sázku*.

Navýšení Hráč dorovná sázku, které čelí a zvýší ji o *malou sázku* v sázkových kolech *preflop* a *flop*, o dvojnásobek malé sázky v sázkových kolech *turn* a *river*.

Očekávaná hodnota Kolik očekávám, že získám žetonů po provedení konkrétní *akce*.

Postflop Trojice sázkových kol *flop*, *turn*, *river*.

Pravidlo skrytí držených karet Hráč má možnost neukázat *držené karty* když nastane *showdown*. Pravidlo není v práci použito, pokud to explicitně neuvedu.

Preflop První *sázkové kolo*. Hráči obdrží *držené karty* a mohou sázet.

River Čtvrté *sázkové kolo*. Je otočena jedna *společná karta*. Hráči zahájí sázení.

Ruka Označuje část hry od rozdání karet všem hráčům, po určení vítěze.

Sázkové kolo Jedno z kol hry: *preflop*, *flop*, *turn*, *river*. Hráči se střídají v sázkách.

Showdown Všichni hráči, kteří *nezahodili* karty mají srovnané sázky. Hráči odhalí *držené karty*. Porovnají se jeji *výherní kombinace* a určí vítěž.

Small blind Hráč, který má po levici hráče *big blind*, po pravici hráče *dealer*. Na začátku ruky vloží povinnou sázku velikosti poloviny *malé sázky*.

Společné karty Karty otočené z *balíčku*, společné pro všechny hráče.

Turn Třetí *sázkové kolo*. Je otočena jedna *společná karta*. Hráči mohou sázet.

Vsazení V sázka v *sázkovém kole*.

Výherní kombinace Kombinace pěti nejlepších karet hráče. Má-li kombinace méně než pět karet, doplní se *kickerem*.

Zahození Volba nepokračovat v sázení v aktuální *ruce*. Hráč přijde o vsazené žetony.

B Obsah přiloženého DVD

bin/
: Pokerbot LAIN ve formátu .jar, identifikační soubor LAIN ve formátu .pd a soubor StartingHandRanks.txt. Soubory jsou určeny pouze pro komerční program `Poker Academy Pro 2.5`.

doc/
: Soubor VytrisalDiplomovaPrace.pdf a archiv VytrisalDiplomovaPrace.zip se zdrojovými soubory pro sestavení PDF.

src/
: Ve složce opentestbed je testovací prostředí `opentestbed` a zdrojové soubory pokerbota LAIN. Pokerbot LAIN je již nastavený na hru tří hráčů v prostředí `opentestbed`. Ve složce LAINProject jsou zdrojové soubory LAIN přednastavené na hru v `Poker Academy Pro 2.5`.

readme.txt
: Informace pro sestavení prostředí pro testování pokerbotů `opentestbed`. Popis, jak ho používat a kde najít zdrojové soubory pokerbota LAIN. Návod na exportování LAIN do formátu .jar pro komerční program `Poker Academy Pro 2.5`.

Navíc DVD obsahuje:

data/
: Složka opentestbed obsahuje data testování pokerbota LAIN v programu `opentestbed`. Složka PokerAcademyPro2.5 obsahuje data testování LAIN v komerčním programu `Poker Academy Pro 2.5`. Data jsou ve formě grafů a histogrií odehraných her.

install/
: Složka Eclipse obsahuje s instalátory vývojového prostředí Eclipse pro 32 bit a 64 bit operační systém Windows. Ve složce JDK jsou instalátory Java Development Kit pro běh prostředí Eclipse. Verze pro 32 bit a 64 bit Windows.

Literatura

[1] DAVIDSON, Aaron. Opponent modeling in poker: Learning and acting in a hostile and uncertain environment. University of Alberta, 2002.

[2] BILLINGS, Darse; DAVIDSON, Aaron; SCHAUENBERG, Terence; BURCH, Neil; BOWLING, Michael; HOLTE, Robert; SCHAEFFER, Jonathan; SZAFRON, Duane. Game Tree Search with Adaption in Stochastic Imperfect Information. University of Alberta, 2004.

[3] SCHAUENBERG, Terence. Opponent Modelling and Search in Poker. University of Alberta, 2006.

[4] JOHANSON, Michael; ZINKEVICH, Martin; BOWLING, Michael. Computing robust counter-strategies. University of Alberta, 2007.

[5] JOHANSON, Michael Bradley. Robust strategies and counter-strategies: Building a champion level computer poker player. University of Alberta, 2007.

[6] BOWLING, Michael; BURCH Neil; JOHANSON Michael; TAMMELIN Oskari. Heads-up limit hold'em poker is solved. University of Alberta, http://jeskola.net/, 2015.

[7] JOHANSON, Michael; BOWLING, Michael. Data biased robust counter strategies. University of Alberta, 2009.

[8] BARD, Nolan; JOHANSON, Michael; BURCH, Neil; BOWLING, Michael. Online Implicit Agent Modelling. University of Alberta, 2013.

[9] VYTŘÍSAL, Jan. Texas hold'em limit pokerbot. Univerzita Palackého, 2013.