VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

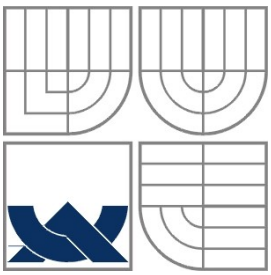
UMĚLÁ INTELIGENCE VE HŘE BANG!

DIPLOMOVÁ PRÁCE
MASTERS'S THESIS

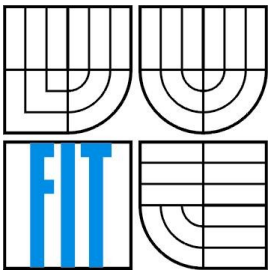
AUTOR PRÁCE
AUTHOR

Bc. VÍT KOLÁŘ

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

UMĚLÁ INTELIGENCE VE HŘE BANG!

ARTIFICIAL INTELLIGENCE IN THE BANG! GAME

DIPLOMOVÁ PRÁCE
MASTERS'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. VÍT KOLÁŘ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. FILIP ORSÁG, Ph.D.

BRNO 2010

Abstrakt

Cílem této diplomové práce je vytvoření umělé inteligence do karetní hry Bang!. Obsahem této práce je kompletní popis hry Bang!, její pravidla, strategické principy používané při hraní a rozbor hry pohledu UI. Dále práce podává přehled metod umělé inteligence a základní informace o disciplíně teorie her. Následuje popis implementace v jazyce C++ a způsobu vytvoření umělé inteligence za pomoci Bayesovské klasifikace a rozhodovacích stromů založených na expertních systémech. Poslední část obsahuje zhodnocení vesměs pozitivních výsledků a závěr s možnými dalšími rozšířeními.

Abstract

The goal of this master's thesis is to create an artificial intelligence for the Bang! game. There is a full description of the Bang! game, it's entire rules, player's using strategy principles and game analysis from UI point of view included. The thesis also resumes methods of the artificial intelligence and summarizes basic information about the domain of game theory. Next part describes way of the implementation in C++ language and it's proceeding with use of Bayes classification and decision trees based on expert systems. Last part represent analysis of altogether positive results and the conclusion with possible further extensions.

Klíčová slova

Bang!, hra, pravidla, strategie, rozhodování, umělá inteligence, UI, teorie her, KBang, client-server, C++, Bayes, rozhodovací strom, implementace, estimace, klasifikace

Keywords

Bang!, game, rules, strategy, decision-making, artificial intelligence, AI, game theory, KBang, client-server, C++, Bayes, decision tree, implementation, estimation, classification

Citace

Vít Kolář: Umělá inteligence ve hře Bang!. Diplomová práce. Brno. FIT VUT v Brně, 2010

Umělá inteligence ve hře Bang!

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Filipa Orsága, Ph.D. a uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Vít Kolář
26. května 2010

Poděkování

Na tomto místě bych chtěl poděkovat vedoucímu mé práce Ing. Filipu Orságovi, Ph.D. za laskavé odsouhlasení mého zadání a za věnovaný čas této práci.

Obsah

1 Úvod

Karetní hra Bang! si od svého vzniku našla mnoho příznivců po celém světě. Tradičním místem, kde se hraje, jsou především společenské prostory a hráče tvoří lidé rozmanitého věku. Dnešní doba si však žádá moderní přístupy a to i v oblasti jako jsou karetní hry. Co dnes není tzv. On-line, přichází o spoustu potenciálu a především příznivců. Ve skutečnosti je tomu tak, že je mnoho lidí, kteří by si určité hry rádi zahráli, ale investovat do papírových kartiček či hracích desek s velmi omezenou životností nechtějí. Tito lidé vyhledávají možnosti jak hrát tyto hry on-line. Hraní je obvykle nestojí žádné výdaje a mohou hrát z pohodlí svého domova či odkudkoli s přístupem na internet. Tak i hry jako je karetní hra Bang! se dočkaly své on-line podoby.

Stejně tak, jako možnost hrát hry on-line, vyžadují dnešní hráči mít možnost hrát proti počítači. Ať už je to z důvodu momentálního nedostatku spoluhráčů, či v tom si něco dokázat, není důležité. Podstatné je, že dnešní doba má velmi rozmanité prostředky nejen ke splnění přání těchto lidí. Značné navyšování výpočetního výkonu a obrovský rozvoj grafických adaptérů byl jistě jedním ze základních faktorů, které odstartovaly rozmanitý vývoj oboru umělé inteligence. Mnoho věcí kolem nás je dnes řízeno počítačem a nyní přišel čas i na oblíbenou karetní hru Bang!

Tato práce navazuje na můj Semestrální projekt, ve kterém jsem vytvořil kompletní a přehledná pravidla hry Bang! včetně strategických postupů a taktických principů používaných hráči v této hře. Na základě této analýzy, uvedené v kapitole 2, jsem později implementoval jednotlivé algoritmy. V rámci Semestrálního projektu jsem si také rozšířil přehled o metodách umělé inteligence a osvěžil jsem si již nabyté znalosti z této oblasti během předchozího studia. Dále jsem své znalosti rozšířil o problematiku disciplíny teorie her. Přehled problematiky těchto disciplín je obsažen v kapitole 3.

Cílem této diplomové práce je implementace umělé inteligence do karetní hry Bang!. Vzhledem k tomu, že cílem hry Bang! je především pobavit hráče, tato umělá inteligence by měla být rovným soupeřem dobrým hráčům, ale na druhou stranu by nemělo být nemožné ji porazit.

Druhá kapitola popisuje právě karetní hru Bang!, její pravidla a strategické a taktické principy používané při jejím hraní. V této kapitole jsem čerpal z [1] a [2] a především vlastních znalostí. Vytvořil jsem ucelená a kompletní pravidla hry včetně strategických principů nabytých zkušenostmi více hráčů. Takovýto soupis strategických principů však nikdy nemůže být dokonalý ani kompletní. Hrát hru Bang! opravdu dobře vyžaduje rozsáhlé strategické a logické myšlení. Jednotlivých strategických principů a taktik v této hře může být, troufám si říci, až nekonečno. Právě strategická rozmanitost této hry jí podle mého názoru přinesla tolik fanoušků po celém světě, přiměla kluby vytvářet turnaje a byla mi výzvou pro vytvoření této práce.

V třetí kapitole se čtenář může seznámit s oborem umělé inteligence, metodami, které používá a disciplínou teorie her. V této kapitole jsem čerpal z [3], [4] a [7]. Podrobný popis všech metod a algoritmů zahrnutý v tomto dokumentu jsem shledal přesahující rozsah a především zaměření tohoto dokumentu. Popis většiny vyjmenovaných metod je podrobně popsán v [3].

Čtvrtá kapitola popisuje první fázi práce na umělé inteligenci. Jde o rozbor hry Bang! z pohledu jednotlivých metod a přístupů UI a způsob vyřešení nejdůležitějšího podproblému, kterým je odhadování rolí ostatních hráčů. V této kapitole jsem čerpal z [5] a [8].

Pátá kapitola popisuje veškeré změny, které jsem provedl v původním zdrojovém kódu projektu KBang (viz níže) a způsob implementace nové umělé inteligence – třídy `AwareAI`. Tato

kapitola by měla sloužit především autorovi původního zdrojového kódu pro usnadnění procesu převzetí mé umělé inteligence, ale i všem ostatním, kteří by chtěli v projektu pokračovat.

Dosažené výsledky jsou popsány a rozebrány v kapitole šesté.

Zhodnocení výsledků a přínosů práce včetně popisu dalšího možného vývoje, předkládá kapitola poslední, Závěr.

V textu se vyskytuje řada výrazů a termínů spojených s hraním hry Bang! a karetních her obecně, které nemusí být čtenáři vždy známy. Vysvětlení těchto termínů je uvedeno v Příloze 1. na konci tohoto dokumentu.

Pro úplnost bych rád uvedl, že všechny názvy a obrázky karet, včetně pravidel a názvu hry Bang! jsou majetkem společnosti: *daVinci Editrice S.r.l.*

Via T.Tittoni, 3
I-06131 Perugia, Itálie
URL: <http://www.davincigames.com/>

a vztahují se na ně všechna autorská práva!

1.1 Stávající řešení UI ve hře Bang!

Po velmi rozsáhlém pátrání jsem zjistil, co jsem opravdu nečekal. K dnešnímu dni (počátek května 2010) na světě není jediná dostupná elektronická verze hry Bang!. Existuje však, zatím nejspíše neveřejný, projekt pokračování projektu KBang, viz níže.

Jednou z motivací k vytvoření této práce – umělé inteligence do hry Bang! pro mne byl projekt **KBang** [6]. Tento projekt vznikl v létě roku 2009 jako bakalářská práce studenta Karlovy University v Praze Michala Čevory. Projekt KBang byl implementován jako client-server aplikace v jazyce C++ za použití knihovny *Qt4* jako „open source“, měl veřejný SVN repositář a byl pravděpodobně první světovou dostupnou on-line verzí této hry. Obsahoval kompletní rozhraní pro hraní základního balíčku hry Bang! a měl cílevědomé plány dalšího vývoje. Projekt obsahoval i počítačem ovládané hráče, ale bez jakékoliv rozumné umělé inteligence - počítač hrál dle pravidel, ale své tahy volil vesměs náhodně a často se dopouštěl i naprosto nesmyslných akcí v rozporu s cílem své role. Na serveru, kde byl projekt spuštěn, byl vidět velký zájem veřejnosti o možnost hrát hru Bang! on-line. Všechna tato fakta i má oblíbenost této hry byly pro mne velkou motivací stát se součástí tohoto projektu. Autor navíc vlastnil uloženou historii všech her, hraných na jeho serveru, což jsem zprvu považoval za (po jisté úpravě) výborná trénovací data pro metody strojového učení.

Naneštěstí však v říjnu roku 2010 kontaktovala autora projektu KBang společnost vlastnící autorská práva na hru a její grafiku – *daVinci Editrice S.r.l.* se stížností ohledně přítomnosti jejich autorského materiálu v tomto „open source“ díle. Žádost autora o udělení práv k použití těchto autorských materiálů v nekomerčním projektu KBang byla zamítnuta. Společnost totiž na konci října roku 2009 veřejně oznámila ve spolupráci se společnostmi *Palzoun Entertainment* a *SpinVector* vydání hry *Bang! The videogame* na únor roku 2010. K dnešnímu dni (počátek května 2010) však tato hra z neznámého důvodu nebyla vydána.

Autor tudíž musel přerušit projekt KBang a zrušit možnost jeho dalšího stahování. V projektu se však rozhodl pokračovat a to zprvu odstraněním veškerého autorského materiálu.

2 Karetní hra Bang!

Bang! je karetní hra na motivy divokého západu. Celosvětově je hra známa pod názvem Bang! krom Francie, kde ji pojmenovali Wanted!. Vroce 2002 ji navrhl Emiliano Sciarra. Ve stejném roce byla vydána italským vydavatelstvím daVinci Editrice. V roce 2004 hra Bang! vyhrála cenu *Origins Award* pro nejlepší hru a nejlepší grafiku. Ve stejném roce byla v České republice nominována na *Hru roku*. Od doby vzniku si hra našla mnoho příznivců po celém světě a vzniklo mnoho oficiálních i neoficiálních rozšíření.

Obrázek 2.1: Logo hry Bang! na obalu, ve kterém se prodává základní balíček hry, převzato z [1]

2.1 Pravidla hry Bang!

Hru hraje čtyři až sedm hráčů (s rozšířeními tři až osm). Každý hráč si vylosuje jednu z následujících rolí: Šerif, Pomocník šerifa, Odpadlík a Bandita. Cíl hry je pro každou postavu jiný:

- Bandité musí zabít Šerifa.
- Šerif a jeho Pomocníci musí zabít všechny Bandity a Odpadlíka. Pomocníci vyhrají, pokud vyhraje Šerif.
- Odpadlík musí zůstat poslední naživu, přičemž jako posledního musí zabít Šerifa.

Každý hráč si také vylosuje jednu postavu, která má speciální schopnosti a daný počet životů (znázorněný počtem nábojnic na kartě postavy). Více o postavách v Příloze 2.

Hra je zajímavým příkladem oboru teorie her. Jedná se o hru s neúplnými informacemi. Známa je pouze role Šerifa, každý hráč tudíž ví pouze, kdo je Šerif a jakou roli hraje on sám. Role ostatních hráčů jsou tajné. Známý je však fakt, kolik jakých rolí, podle celkového počtu hráčů, je ve hře. S postupem hry se postupně odkrývají role mrtvých hráčů a každý se snaží odhadovat podle tahů ostatních, kdo hraje za jakou roli. Toto je však velmi složité, jelikož hráči mohou blafovat (chovat se jako by hráli za jinou roli) a navíc každý se snaží svou roli utajit. Výhoda utajení vlastní role však nesmí být upřednostňována před splněním herního cíle (např. pro Banditu zabití Šerifa).

Hráči mohou hrát mnoho různých strategií a žádný tah nikdy nemá totální vliv na prozrazení role. Strategická variantnost této hry je jistě jedním z důvodů její oblíbenosti. Problematice strategií se budu věnovat v dalších kapitolách.

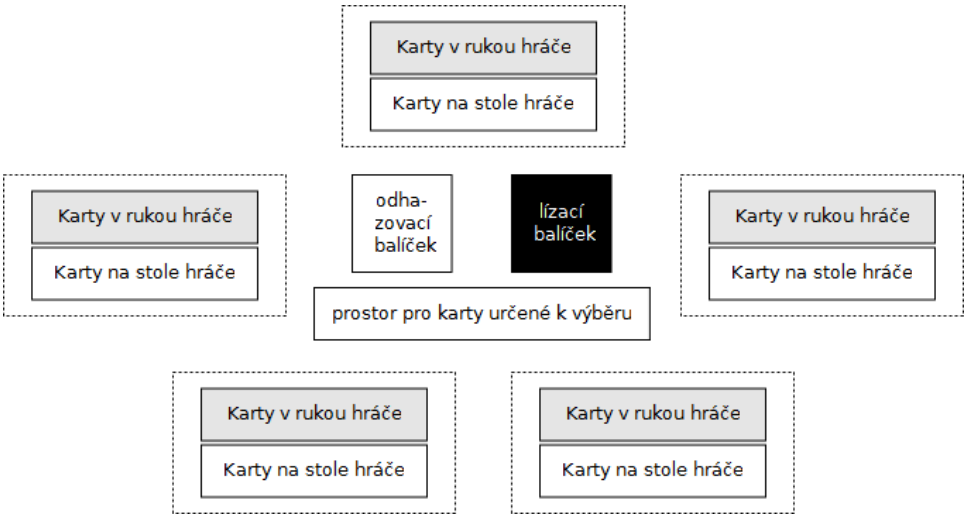
2.1.1 Začátek hry

Každému hráči se náhodně vylosuje role a postava, za kterou bude hrát. Ve hře je vždy Šerif, Bandita a Odpadlík s tím, že Šerif je vždy právě jeden a počet Banditů a Pomocníků záleží na celkovém počtu hráčů a to tak, že Pomocníků spolu s Šerifem by mělo být stejný počet jako Banditů a další hráč je Odpadlík. Při sudém počtu hráčů je Banditů o jednoho více a při maximálním počtu hráčů, tj. osm, hrají dva Odpadlíci.

Šerif odhalí svou roli, ostatní role jsou tajné – známé pouze hráčům, kteří za roli hrají. Schopnosti postav všech hráčů jsou veřejné, platí celou hru a nelze je změnit. U každé postavy je uvedeno, kdy lze schopnost využít a žádná ze schopností hráče nenutí, aby ji použil vždy, kdy k tomu je příležitost. Schopnosti postav jsou vždy nadřazené pravidlům napsaným v pravidlech a navzájem se nijak nevylučují. Přehled schopností postav je v Příloze 2.

Každý hráč má právě tolik životů, kolik životů (nábojnic) má jeho postava, pouze Šerif má o jeden život více. Tento počet životů je po celou hru a pro všechny hráče maximální, jakého mohou dosáhnout. Všichni hráči dostanou před prvním tahem náhodně vybrané karty a to právě tolik, kolik životů (nábojnic) má jejich postava. Šerif má o život více, ale karet dostává podle počtu nábojnic jeho postavy. Vzdálenost dvou sousedních hráčů je na začátku hry vždy 1 (krom postav se speciální vlastností ovlivňující vzdálenost).

Následující obrázek zobrazuje popis herního stolu. Části s bílým pozadím jsou pro všechny hráče veřejné. Části s šedým pozadím jsou skryté všem ostatním hráčům a nakonec lízací balíček s černým pozadím je samozřejmě tajný všem.



Obrázek 2.2: Popis herního stolu

2.1.2 Hra

Hru začíná Šerif a hra pokračuje neustále po směru hodinových ručiček. Kolo každého hráče je rozděleno do následujících tří fází:

1. **Vzití dvou karet**
   Hráč, který je na tahu, si z lízacího balíčku (zbytek karet po rozdání karet na začátku) vezme do ruky dvě karty. Dojdou-li karty v lízacím balíčku, zamíchá se odhazovací balíček a vytvoří se tak nový lízací balíček.
2. **Zahrání herních karet**
   Hráč může zahrát kolik chce karet ku svému prospěchu nebo proti jiným hráčům pouze s následujícími omezeními:
   - Lze zahrát pouze jednu kartu *BANG!* za své kolo (pokud nemá hráč vyloženu kartu zbraně *Volcanic* a nebo nemá schopnost zahrát libovolný počet karet *BANG!* jako např. postava *Willy the Kid*).
   - Žádný hráč nesmí mít před sebou na stole vyložené dvě karty se stejným názvem (vykládat na stůl lze ve hře bez rozšíření pouze karty s modrým okrajem).
   - Každý hráč může mít vyloženou pouze jednu zbraň. Pokud chce hráč vyložit zbraň a už jednu zbraň vyloženu má, tak tu vyloženou musí nejdříve odhodit na odhazovací balíček, a až poté může vyložit zbraň novou.

   Efekt zahrání karty určují symboly, jež jsou na kartě znázorněné a barva okraje. Toto je podrobněji popsané v kapitole 2.1.3 Efekty karet. Běžně lze karty hrát pouze během vlastního tahu (s výjimkou karet *Pivo* a *Vedle!*) a po vyzvání efektem karty nějakého jiného hráče.
3. **Odhození přebývajících karet**
   Poté co hráč již nechce nebo nemůže zahrát další karty, nastává fáze 3 jeho tahu - hráč musí odhodit všechny přebytečné karty, aby mu na konci kola zbývalo maximálně pouze tolik karet, kolik je aktuální počet jeho životů. Konec fáze 3 je konec kola hráče a je na tahu další hráč (po směru hodinových ručiček).

2.1.3 Efekty karet

Na každé kartě jsou znázorněny **symboly, které určují efekt karty**. V balení hry Bang! je *přehledová karta*, která obsahuje všechny tyto symboly včetně jejich vysvětlení, viz následující obrázek karty:

Obrázek 2.3: Přehledová karta symbolů efektů karet, převzato z [1]

Hrací (efektové) karty v základním balíčku hry Bang! jsou dvojího druhu. Liší se barvou okraje. **Karty s hnědým okrajem** se zahrají odhozením na odhazovací balíček a jejich efekt se řeší okamžitě. Druhým typem jsou **karty s modrým okrajem**, které se zahrají (vloží do hry) tím, že si je hráč vyloží před sebe (krom karty *Vězení* viz níže) a mají svůj efekt stále, dokud jsou ve hře, tj. dokud nejsou odstraněny ze hry např. kartou *CatBalou* nebo u *Dynamitu* jeho vybuchnutím (splnění speciální podmínky).

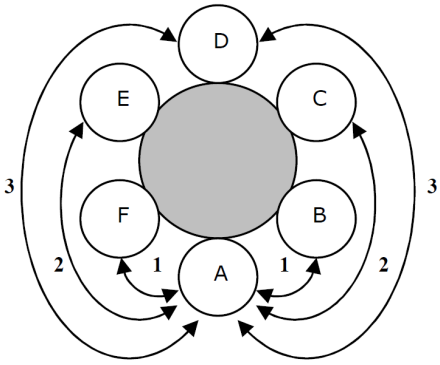
Karty *BANG!* jsou základním prostředkem na střelbu a ubírání životů ostatním hráčům. Chce-li hráč zahrát kartu *BANG!* a tím vystřelit na jiného hráče, musí na daného hráče mít dostřel – viz odstavec Vzdálenost mezi dvěma hráči. Hráč na kterého byla karta *BANG!* Zahrána, může okamžitě zahrát **kartu *Vedle!*** čímž se vyhne této střele a neztratí žádný život. Nezahraje-li hráč kartu *Vedle!* musí si okamžitě ubrat jeden život. Pakliže ztrácí poslední život, hra pro hráče končí, nezahraje-li při této ztrátě posledního života kartu *Pivo*.

Karta *Pivo* může být zahrána dvěma způsoby: buď běžně, během vlastního tahu a nebo v cizím tahu, ale pouze v případě ztráty posledního života a vždy vyléčí hráči, který ji zahrál, jeden život. Hráči nemohou vyléčit životy, které neztratily – nemohou získat více životů než měla jejich postava na začátku hry. Karta *Pivo* nemůže být použita k vyléčení jiného hráče a nesmí se zahrát v situaci, kdy ve hře zbývají poslední dva hráči.

Karta *Salón*, když je zahrána, má efekt takový, že vyléčí 1 život všem hráčům včetně hráče, který kartu zahrál. Symboly na kartě jsou ve dvou řádcích a vždy platí současně, nejedná se o něco jako o možnost volby.

Vzdálenost mezi dvěma hráči je minimum počtu míst (aktuálně hrajících hráčů) mezi nimi, počítaje po směru nebo proti směru hodinových ručiček. Vzdálenost je velmi důležitá, jelikož běžně hráč dostřelí/dosáhne pouze na hráče na vzdálenost 1, což jsou pouze hráči sedící bezprostředně vedle. Místa vyřazených hráčů ze hry se do vzdálenosti nezapočítávají, takto se vzdálenosti mezi hráči postupně zmenšují. Vzdálenost mezi hráči upravují karty koní. V základním balíčku karet (hra Bang! bez rozšíření) jsou to karty s modrým okrajem *Mustang* a *Appaloosa*.

Hráč, který má vyloženou kartu *Mustang*, je pro všechny ostatní hráče vzdálen na vzdálenost o 1 vyšší. On však ostatní hráče vidí pořád na normální vzdálenost. Dá se říci, že karta *Mustang* je obranná, jelikož má efekt pouze v tazích ostatních hráčů, kdy majitele karty od nich vzdaluje.



Obrázek 2.4: Vysvětlující diagram vzdáleností hráčů, převzatý z [1]

Hráč, který má vyloženou kartu *Appaloosa,* má všechny hráče na vzdálenost o jedna menší. Ostatní hráči ho však stále vidí na normální vzdálenost. Karta *Appaloosa* je útočná, má efekt pouze v tahu hráče, který ji má vyloženou a přibližuje k němu ostatní hráče.

Další karty s modrým okrajem jsou **zbraně**. Každý hráč může na začátku střílet pouze na hráče na vzdálenost 1 (bez karet koní a speciálních schopností postav to je pouze hráč bezprostředně po levici a pravici). Chce-li hráč střílet na vyšší vzdálenost, musí mít vyloženou kartu zbraně. Jedná se o karty s modrým okrajem, ilustrací zbraně a symbolem s číslem v „hledáčku“, které znázorňuje maximální dostřel, který karta hráči poskytuje. Jak již bylo řečeno, hráč může mít vyloženou pouze jednu kartu zbraně, pokud chce vyložit jinou, musí nejdříve stávající odhodit.

Na některých kartách (*Barel*, *Vězení* a *Dynamit*) je symbol, určující efekt karty, znázorněn jako karetní symbol. Hráč, který zahraje tuto kartu, si musí nejdříve **líznout z balíčku** – otočit vrchní kartu balíčku a odhodit ji. Jestliže znak na otočené kartě v pravém dolním rohu odpovídá znaku (popřípadě ještě hodnotě u karty *Dynamit*), bude mít karta efekt. V opačném případě se nic nestane.

V základním balíčku hry je 6 karet, které na sobě mají symbol knihy, a jejich efekty jsou trochu složitější viz následující odstavce.

Karta *Dynamit*: Hráč, který zahraje tuto kartu, ji položí před sebe. *Dynamit* před ním zůstane ležet celé kolo. Když hráč začne své následující kolo (a má *Dynamit* stále ve hře), před svou první fází si musí líznout (otočit a odhodit vrchní kartu balíčku, viz odstavec výše). Otočí-li pikovou kartu mezi 2 až 9, *Dynamit* exploduje, odhodí se a hráč ztrácí 3 životy. V opačném případě pošle *Dynamit* hráči po levici, který musí to samé provést na začátku svého kola. Hráči si takto předávají *Dynamit* dokud neexploduje (s efektem výše popsaným) nebo je odhozen pomocí karet *CatBalou* nebo *Panika!* Je-li *Dynamit* u hráče společně s kartou *Vězení*, testuje se nejprve *Dynamit*. Pokud by byl hráč zabit kartou *Dynamit*, jeho zabití není uvažováno jako zabití jiným hráčem (pro pravidla Trestů a odměn v kapitole 2.1.5).

Karta *Vězení*: Hráč, který zahraje tuto kartu, ji položí před jakéhokoliv hráče, který je tím uvězněn. Uvěznit nelze Šerifa ani hráče, který už *Vězení* před sebou má. Uvězněný hráč si musí líznout před začátkem svého tahu. Lízne-li si srdcovou kartu, utekl z vězení a karta *Vězení* je odhozena a on pokračuje normálně ve svém tahu. V opačném případě odhodí *Vězení* a ztrácí fázi 1 a 2 svého tahu, tudíž pouze musí odhazovat přebytečné karty, dle svého počtu životů. Na hráče se může střílet a hráč může stále zahrávat karty *Vedle!* a *Pivo* mimo vlastní tah.

Karta *Hokynářství*: Po zahrání se otočí tolik vrchních karet balíčku, kolik je aktuálně hrajících hráčů. Počínaje hráčem, co zahrál tuto kartu, se postupuje po směru hodinových ručiček a každý hráč si vybere jednu z otočených karet, kterou si vezme do ruky.

Karta *Kulomet*: Každý hráč, vyjma hráče, který tuto kartu zahrál, musí odhodit kartu *Vedle* nebo ztratit 1 život. Lze zkusit se schovat za *Barel*.

Karta *Indiáni!*: Každý hráč, vyjma hráče který tuto kartu zahrál, musí odhodit kartu *BANG!* nebo ztratit 1 život. Nelze zahrávat *Vedle!* ani zkoušet se schovat za *Barel*.

Karta *Duel*: Hráč, který zahraje tuto kartu, vyzývá jiného hráče na duel. Vyzvaný hráč musí odhodit kartu *BANG!*. Odhodí-li vyzvaný hráč kartu *BANG!*, hráč, který zahrál *Duel,* musí také odhodit *BANG!* a tak dále až do té doby, kdy jeden z hráčů *BANG!* neodhodí (buď nechce nebo nemůže), ten ztrácí 1 život a duel končí. *Duel* vždy končí ubráním života právě jednoho z hráčů, toho, který první neodhodí kartu *BANG!*. Na *Duel* se nedá bránit kartou *Vedle!* a tudíž samozřejmě ani *Barelem*.

Karta *Volcanic*: Hráč, který má tuto kartu ve hře (vyloženu před sebou) může zahrát během svého tahu libovolný počet karet *BANG!*. Hráč může střílet na libovolné cíle, avšak zbraň *Volcanic*

má dostřel pouze 1, tudíž pokud hráč nemá rozšířen svůj dostřel (schopností své postavy nebo kartou upravující vzdálenost ostatních hráčů), může takto střílet pouze na bezprostřední sousedy.

2.1.4 Vyřazení postavy ze hry

Ztratí-li hráč (postava) poslední život, hra pro něj končí. Jediná možná záchrana v této situaci je zahrání karty *Pivo*, které se jako jediná karta dá zahrát po ztrátě posledního života, vyléčí 1 život hráči a hráč může pokračovat ve hře. Kartu *Salón* v této situaci zahrát nelze.

Hráč, pro kterého hra skončila, otočí (všem ukáže) svou kartu role a odhodí všechny karty, které mu zbyly v ruce i vyložené karty ze stolu.

2.1.5 Tresty a odměny

- **Vyřadí-li ze hry Šerif svého Pomocníka**, musí Šerif jako trest odhodit všechny karty, které mu zbyly v ruce i své vyložené karty ze stolu.
- **Kdokoli, kdo vyřadí ze hry Banditu** (i kdyby to byl sám také Bandita), vezme si jako odměnu 3 karty z lízacího balíčku.

2.1.6 Konec hry

Hra končí, nastane-li jedna z následujících podmínek:

- **Šerif je vyřazen ze hry.**
  Zbyl-li ve hře v této situaci pouze Odpadlík, je vítěz. Jinak vyhrávají všichni Bandité.
- **Všichni Bandité i Odpadlík jsou vyřazeni ze hry.**
  V této situaci vyhrává Šerif a všichni jeho Pomocníci.

2.2 Strategie a taktiky ve hře Bang!

Základní strategií pro každého hráče jistě je snažit se hru vyhrát.

2.2.1 Základní strategie určená vylosovanou rolí

Šerif by měl střílet pouze na své nepřátele, ale především nesmí dopustit, že ho ostatní hráči vědomě přesvědčí, že jeden z jeho Pomocníků je Bandita a on ho zabije. Pokud si Šerif opravdu není jist, koho se chystá zabít, nesmí tak učinit, ale musí hráči například nechat jen jeden život a nechat samotné zabití na ostatních hráčích. V situaci, kdy Šerif omylem zabije jednoho ze svých pomocníků totiž přichází o všechny své karty a je tím pádem velmi snadný cíl a většinou hru prohrává, jelikož je zabit. Platí, že Šerif může hru vyhrát, i pokud ani jedinkrát nevystřelí. Prioritou role Šerifa je tedy sám přežít a pokud ho okolnosti nutí ke střílení, pouze udržuje rovnováhu mezi ostatními hráči. Pro Šerifa je ze všech nejsložitější odhadnout, kdo hraje za jakou roli, může to pouze odhadovat z tahů ostatních hráčů, kteří však mohou blafovat. Všichni ostatní hráči znají svoji roli, tudíž se jim lépe odhaduje, kdo ze zbylých hráčů je kdo a po smrti pár hráčů mohou mít v tomto jistotu velmi brzo.

Bandité se většinou příliš neukrývají, jelikož nemohou vyhrát, pokud nezabijí Šerifa. Obvykle střílejí na Šerifa hned, jakmile mají dostřel a co nejvíce způsobů jak ho zranit a oslabit (ukradení či zničení karet kartami *CatBalou* a *Panika!*). Vystřelením na Šerifa se Bandité vzájemně poznají a nestřílí na sebe z nevědomosti. Za smrt Bandity je odměna tří karet, proto pokud je nějaký bandita

téměř mrtev (má jeden život a málo karet v ruce), je výhodné pro Banditu takového svého spoluhráče zabít, čímž sami bandité získají tuto odměnu, naproti tomu aby ji získal protivník.

Pomocníci Šerifa začnou střílet na Bandity, jakmile je rozpoznají. Pomocníci by se měli vyvarovat takového blafování vůči Banditům, aby si Šerif mohl myslet, že jsou Bandité, zabil je a tím hru dá se říci sobě i všem Pomocníkům prohrál. Proto Pomocníci by měli tento typ blafování využívat např. pouze v situaci, kdy na ně Šerif nedostřelí a nebo pokud má Šerif jiný, typicky snadnější, cíl.

Odpadlík je role nejtěžší, hraje na obě strany, v tu pravou chvíli na té správné a je to právě tato role, která dělá odhadování rolí hráčů tak těžké a blafování tak efektivní. Prioritou Odpadlíka je, stejně jako u Šerifa, sám přežít, jelikož aby vyhrál, musí zůstat poslední ve hře. Proto by ze začátku hry měl Odpadlík hrát na straně Šerifa, aby jej Pomocníci či Šerif nezabili. V situaci, kdy Bandité zeslábnou (např. úmrtí druhého Bandity, apod.), se Odpadlík staví na stranu Banditů a snaží se co nejrychleji zabít všechny Pomocníky a zároveň bránit Šerifa. Jednoduše řečeno: Odpadlík se vždy staví na stranu slabších, aby vyrovnával síly, čímž dopomáhá vzájemnému zabíjení ostatních hráčů mezi sebou a zároveň brání Šerifa, jehož smrt by pro Odpadlíka znamenala prohru. Role Odpadlíka je velmi složitá (proto také na turnajích velmi vysoce hodnocená) a jeho strategie se vždy především odvíjejí od aktuální situace ve hře.

2.2.2 Strategie na základě dostřelu na Šerifa

Velmi důležitým aspektem hry je vzdálenost od ostatních hráčů, především od Šerifa.

Jsem-li Bandita a sedím na druhé straně stolu od Šerifa, pak jediným způsobem jak ho mohu zranit je, že si získám a udržím ve hře zbraň. Pakliže bych se tímto způsobem jako Bandita prozradil, velmi rychle mi Šerif či jeden z jeho Pomocníků tuto pro mne klíčovou zbraň sebere (karta *CatBalou* a *Panika!*), proto je v takové situaci lepší se neprozradit a zabíjet Pomocníky mezi mnou a Šerifem.

V opačném případě, kdy jsem Bandita a sedím bezprostředně vedle Šerifa, jsou dvě možnosti. Blafovat, že jsem Pomocník a začít střílet na Šerifa v nějakém dobrém okamžiku (např. když má málo životů, mám v ruce *Volcanic* a dost karet *BANG!*, apod.) nebo při dostatku způsobů jak ho oslabit na něj začít ihned střílet a doufat, že ostatní Bandité mají také dostřel.

Úkolem Šerifových Pomocníků i Odpadlíka je držet Bandity a hráče, u nichž si nejsou jisti rolí, mimo dostřel na Šerifa, což dělají pomocí karet *CatBalou* a *Panika!*. Ideální strategie u ničení zbraní hráčům, u kterých si Pomocníci nejsou jisti jejich rolí, je ta, že se jim ničí ty zbraně, kterými dostřelí na Šerifa, ale zároveň se jim ponechá možnost střílet na hráče, o kterých si Pomocníci myslí, že jsou Bandité pro případ, kdyby tito hráči byli také na straně Šerifa.

2.2.3 Plnění cíle vs. odhalení své role

Rozhodnout se, kdy přímočaře plnit cíl své role a kdy zůstat jako role v utajení, je velmi složité a odvíjí se to od aktuální situace ve hře. Ideální je plnit svůj cíl a zároveň zůstat v utajení.

Na začátku hry je klíčový moment, kdy se prozradí Bandité a jaké možnosti mají ostatní Bandité po prozrazení prvního. Pro Bandity je vždy lepší, když se jako první prozradí první Bandita po tahu Šerifa, jelikož než Šerif bude znovu na tahu a bude se moci např. uzdravit, budou na tahu ještě všichni ostatní Bandité, kteří mohou Šerifa zabít, čímž by hra skončila, než se Šerif bude moci zregenerovat a Pomocníci vyřadit Bandity ze hry.

Vždy je výhodné, pokud je hráčova role ostatním stále utajena, aby toto prozrazení udělal až v situaci, kdy tím opravdu něčeho dosáhne (např. úplně vyřadí ze hry, zabije postavu, na kterou se

chystá střílet). Každé zabití nějakého hráče silně ovlivňuje rozložení sil na obou stranách, proto jedna věc je někoho zranit či oslabit, ale druhá věc je někoho úplně vyřadit ze hry.

2.2.4 Taktika podle schopnosti postavy

Některé postavy mají jen jisté zvýhodnění, ale některé vyžadují speciální taktiku a strategii jak za danou postavu hrát.

Typicky např. postava *Bart Cassidy* a *El Gringo* (popis postav viz Příloha 2.), pokud si mohou dovolit ztratit život (mají životů dost, mají kartu *Pivo!* apod.), tak to udělají, jelikož tím získají kartu, která v určitých fázích hry může být klíčová.

Jesse Jones pokud je přesvědčen, kdo je protihráč a neprozradí se tím, tak svojí schopností může každé kolo takto protihráče oslabovat.

2.2.5 Ztratit život vs. ztratit kartu

Někdy je lepší ztratit život než kartu. Typicky např. při reakci na efekt karty *Indiáni!* a *Duel*, na které se reaguje kartou *BANG!*, kterou však mohu plánovat použít k výstřelu.

2.2.6 Role náhody

Jako ve všech karetních hrách je i v této role náhody. Na začátku hry se role náhody vztahuje na rozlosování rolí, kde je jasné, že pokud vedle Šerifa sedí dva Bandité, jsou Šerifovy šance na výhru menší, než pokud vedle Šerifa sedí dva jeho Pomocníci. Všechny situace však může vyhrát jakákoliv strana, rozlosování rolí nikdy nemá totální vliv na výhru jedné strany.

Druhý faktor ovlivněný náhodou je lízání a u každého hráče v každém tahu braní karet z (poctivě zamíchaného) balíčku. Tento faktor ovlivněný náhodou je však pro všechny hráče stejný a jen zřídka dochází k situacím, kdy štěstí či smůla ovlivní výsledek celé hry.

2.2.7 Blafování

Blafování je, jak u všech her, tak i u této hry, náročné. Málo kdy je hráč v situaci, kdy je blafování opravdu výhodné, proto se ve hře Bang! příliš často neblafuje. Jediným způsobem blafování je, dá se říci, vydávat se za jinou roli.

Jediná role, která častěji než velmi zřídka využívá blafování je Odpadlík a hráč v nouzi (např. poslední přeživší bandita, apod.). Blafování je vždy třeba promyslet z pohledu ostatních hráčů a ujistit se, že výsledek bude pro mne a pro cíl mých spoluhráčů výhodný.

Typickým špatným blafem je např. když se Pomocník vydává za Banditu tím, že vystřelí na Šerifa, čímž spustí na Šerifa vlnu střel od všech Banditů a Šerif zemře. Ideálním blafem je něco neškodného, co hráče přesvědčí o roli, za kterou chci, aby si mysleli, že hraji, ale zároveň příliš neublíží mým spoluhráčům. Typicky sebrání karty pomocí karet *CatBalou* a *Panika!* někomu, jehož role je již prozrazena.

2.2.8 Strategie pro dané kolo a držení karet v ruce

Aktuální situace ve hře si může vyžádat specifické tahy hráče. Např.: Po zahrání *Hokynářství* se ví, jaké karty si ostatní hráči vzali, což může ovlivnit hráčovy plány. Po zahrání karty *Indiáni!* je šance, že hráči mají další kartu *BANG!* nižší, což může ovlivnit výsledek zahraného *Duelu*, apod.

S určitou kombinací karet v ruce mohu čekat na nějakou konkrétní kartu, proto klíčový tah odkládám. V ruce mohou hráči na konci svého tahu mít pouze tolik karet, kolik je jejich aktuální počet životů, i proto je lepší si držet dostatečný počet životů, jelikož karty v ruce jsou to, co hráče brání a pomocí čeho plní svůj cíl. Téměř vždy je dobré si v ruce ponechat jednu kartu *BANG!* a jednu kartu *Vedle!*. Tyto karty jsou nejpoužívanější karty ve hře a mohou vám zachránit cenné životy. Pokud si mohu ponechat pouze dvě karty v ruce, což znamená, že mám pouze dva životy a zbudou mi pouhé dvě karty, čímž jsem poměrně snadný cíl pro protihráče, vždy je dobré, mám-li tu možnost, si ponechat právě ony dvě zmíněné nejpoužívanější karty.

Nepotřebuji-li některé karty zahrát a mohu-li si je nechat v ruce, vždy tak činím, jelikož tím zvyšuji míru neúplných informací o mé postavě, což je výhodné. Typicky např. u karet *Pivo*, zbraní, když jednu mám už vyloženou, atd.

3 Umělá inteligence

Umělá inteligence, často označována zkratkou UI či AI (anglicky artificial intelligence), je obor informatiky zabývající se teoreticky i prakticky napodobováním „inteligentního chování“ pomocí strojů, typicky počítače. Obor umělé inteligence je mladý, zároveň však velmi rychle se rozvíjející, především díky obrovskému rozvoji výpočetní techniky v poslední době. Rozsah tohoto oboru je v dnešní době již tak velký, že jej není dost dobře možné charakterizovat několika větami. Oficiální definice umělé inteligence zatím neexistuje, jelikož naráží na problém definice či přesného vymezení „inteligence“ samotné, jako takové. O systému se říká, že je „inteligentní“, pokud za daných okolností „udělá tu správnou věc“ či „se rozhodne správně“.

Ačkoliv se vytvořit obecnou umělou inteligenci, která by byla srovnatelná s lidskou, ukázalo být nesmírně obtížné (v dnešní době spíše zatím nemožné), vědci během posledních padesáti let vyvinuli sadu postupů, které dosahují v jednotlivých problémech dílčích úspěchů. Tyto postupy můžeme nazývat metody. Je jich více druhů a každá se více či méně hodí na řešení konkrétních úloh, konkrétních problémů. Rozdělení (klasifikaci) těchto metod vystihují následující podkapitoly.

K vytvoření této kapitoly jsem čerpal z [3], [4], [7] a [8].

3.1 Metody řešení úloh založené na prohledávání stavového prostoru

Princip těchto metod spočívá ve vhodném procházení stavového prostoru řešené úlohy za účelem nalezení stavu požadovaného, také nazývaného cílového.

I přes dnes dostupný vysoký výpočetní výkon, je pro většinu problémů zcela nemyslitelné, aby stroj hledal řešení postupným testováním všech možností (stavů). Vznikla potřeba hledání nějak efektivně řídit. Pokud si řešenou úlohu rozdělíme do různých stavů a definujeme, že jeden z těchto stavů je počáteční, některé stavy jsou cílové a mezi různými stavy je možné aplikováním určitých akcí (operátorů) přecházet, vznikne nám tak stavový prostor. Stavový prostor je často graficky reprezentován jako orientovaný graf, jehož uzly jsou stavy a přechody udávají akci, jejímž vykonáním se dostaneme z jednoho stavu do druhého. Stavový prostor může být pro řadu úloh příliš rozsáhlý, v některých případech dokonce i nekonečný.

Nejen z těchto důvodů vznikly v uplynulých desítkách let různé metody (algoritmy) prohledávání stavového prostoru s různými výhodami a nevýhodami. Nelze říci, že některá z metod je jednoznačně lepší než jiná. Záleží vždy na povaze řešené úlohy, požadavcích na řešení a dostupných prostředcích.

Metody prohledávání stavového prostoru spočívají v nalezení cesty v grafu mezi počátečním a cílovým uzlem a dělí se na dvě základní skupiny: Neinformované a Informované, viz následující podkapitoly.

3.1.1 Neinformované metody

Neinformované metody [3] [4] prohledávání nemají k dispozici žádné vhodné znalosti o stavovém prostoru ani žádné prostředky k ohodnocení jednotlivých stavů, které by jim umožnily urychlit cestu k cíli. Musí tudíž systematicky procházet uzly, dokud nenaleznou řešení. Jednotlivé algoritmy se od

sebe liší jen způsobem, jakým toto systematické procházení provádějí, zda-li vždy naleznou řešení, když existuje, zda-li nalezené řešení je nejlepší ze všech, apod.

Mezi Neinformované metody prohledávání stavového prostoru patří např.:

- Metoda prohledávání do šířky (BFS - Breadth First Search)
- Metoda stejných cen (UCS - Uniform Cost Search)
- Metoda prohledávání do hloubky (DFS - Depth First Search)
- Metoda omezeného prohledávání do hloubky (DLS - Depth Limited Search)
- Metoda postupného zanořování do hloubky (IDS - Iterative deepening DFS)
- Metoda zpětného navracení (Backtracking)
- Metoda obousměrného prohledávání (BS - Bidirectional BFS search)

3.1.2 Informované metody

Informované metody [3] [4] prohledávání mají navíc znalosti o stavovém prostoru, které jim umožňují odhadnout, jak daleko se nachází řešení od aktuálního stavu. Tento odhad reprezentuje tzv. heuristická funkce. Heuristickou funkci dodává na základě znalostí člověk a informované metody jsou na ní kriticky závislé. Čím lepší heuristika je k dispozici, tím rychleji a s menším zatížením paměti dojde k nalezení řešení.

Informované metody se dělí na dvě kategorie:

Informované Metody založené na výběru nejlépe ohodnoceného stavu (Best First Search)
Tyto metody jsou velmi podobné neinformované metodě UCS. Patří sem např.:

- Metoda „hladového“ prohledávání (Greedy Search)
- Algoritmus A* (A star)

Informované Metody lokálního prohledávání (Local Search)
Řešením těchto metod je pouze nalezení cílového stavu a vlastní cesta je bezvýznamná. Metody lokálního prohledávání neprohledávají stavový prostor systematicky, přesto mají své přednosti: Mají zcela zanedbatelnou paměťovou náročnost a často dospějí k přijatelnému řešení i v rozsáhlých, typicky nekonečných, stavových prostorech, kdy použití výše popsaných systematických algoritmů je nemožné. Patří sem např.:

- Metoda „lezení do kopce“ (Hill-climbing)
- Metoda simulovaného žíhání (Simulated annealing)

3.2 Expertní systémy

Tuto problematiku jsem nastudoval z [3] a [4]. Pojem expertní systém se poprvé objevil zhruba před 30ti lety jako přirozený důsledek poznání, že kvalita systémů s umělou inteligencí závisí daleko více na kvalitě znalostí, než na kvalitě mechanismů pro jejich využívání. Expertní systémy jsou tedy systémy opírající se o špičkové znalosti odborníků – expertů. Naděje, kterou úspěchy prvních expertních systémů vlily do vědeckých týmů, způsobila, že se pojem expertní systém rychle rozšířil a to nejen v odborných kruzích. Vědecké časopisy však možnosti expertních systémů tak nadsadily a vyvolaly až nereálná očekávání, že svým způsobem tento pojem zamlžily a nakonec zdiskreditovaly.

Dnes tyto systémy mají charakter specializovaných, problémově orientovaných subsystémů, avšak do oboru umělé inteligence bezesporu patří.

Oficiální definice zatím neexistuje. Následující popisy se vyskytují v odborné literatuře: „Expertní systém je:

- počítačový systém hledající řešení problému v rozsahu určitého souboru tvrzení nebo jistého seskupení znalostí, které byly formulované experty pro danou specifickou oblast.
- systém založený na reprezentaci poznatků expertů, které využívá při řešení zadaných úloh.
- systém kooperujících programů na řešení vymezené tříd úloh, v jednotlivých problémových oblastech obyčejně řešené experty.
- počítačový systém vybavený znalostmi odborníka (experta) ze specifické oblasti, v jejichž rozsahu je schopný učinit rozhodnutí rychlostí i kvalitou vyrovnávající se nejméně průměrnému specialistovi.“

Jednoduše řečeno: Expertní systém je software, který má za úkol poskytovat expertní rady, rozhodnutí nebo doporučit řešení v konkrétních situacích.

Expertní systémy jsou navrženy tak, aby mohly zpracovávat nenumerické a neurčité informace a řešit tak úlohy, které nejsou řešitelné tradičními algoritmickými postupy a mají dvě základní komponenty, které jsou na sobě relativně nezávislé. Řídící mechanismus pro odvozování závěrů a bázi znalostí. Veškerá inteligence je uložena mimo programový kód. Programový kód řídícího mechanismu má za úkol pouze vyhodnocovat stav, který je ovlivněn expertními znalostmi uloženými v bázi znalostí a informacemi získanými z okolního světa (například odpovědi na otázky kladené uživateli, výsledky měření nějakých senzorů apod.).

Oproti člověku, expertovi, mají expertní systémy mnoho výhod, ale i možných nevýhod.

Expertní systémy používají při řešení úlohy dvě základní strategie procesu usuzování.

1. **Dopředné řetězení**: Jedná se o usuzování řízené daty (dopředné řetězení, forward chaining). Expertní systém postupuje tak, že získává potřebná data a na jejich základě se rozhoduje. Používá se při řešení problému zahrnující syntézu (navrhování, konfigurace, plánování apod.).
2. **Zpětné řetězení**: Jedná se o usuzování řízené cíli (zpětné řetězení, backward chaining). Expertní systém postupuje tak, že vybere možný závěr a pokouší se dokázat jeho platnost hledáním dat, které tento závěr podporují. Tato strategie je vhodná pro diagnostické problémy, které mají malý počet cílových hypotéz.

S expertními systémy je úzce spojena problematika Reprezentace znalostí (viz [3]) a Dolování dat z databází (viz [5]).

3.3 Strojové učení

K vytvoření této kapitoly jsem čerpal z [3] a [4]. Strojové učení je oblastí umělé inteligence, zabývající se algoritmy a technikami, které umožňují počítačovému systému „se učit“ resp. měnit své znalosti a vlastnosti tak, že příště bude vykonávat stejnou nebo podobnou úlohu efektněji.

Strojové učení se rozděluje do tří základních skupin, viz následující podkapitoly.

Učení s učitelem (supervised learning)
Učení se provádí na tzv. trénovací množině, která obsahuje pro každý případ vstupní hodnoty a množinu požadovaných resp. správných hodnot - výstupních. Pro každý krok učení je známá požadovaná hodnota, čili systém je okamžitě informován o aktuálním hodnocení poslední akce na základě které upraví své nastavení. Učení probíhá tak dlouho, dokud pro vstupní hodnoty není výstupní hodnota systému v rámci zvolené odchylky od hodnoty požadované. Hodnota výstupní funkce může být spojitá (při regresi) či diskrétní (při klasifikaci).

Patří sem např.:

- Rozhodovací stromy (Decision tree)
- Metody prohledávání prostoru verzí

Učení bez učitele (unsupervised learning)
Učení bez učitele se také provádí na trénovací množině, avšak v tomto případě ke vstupním hodnotám není dostupná žádná podpůrná informace. Při učení se hledají shluky obrazů představovaných mnohorozměrnými číselnými vstupními vektory z trénovací množiny. Jiný název pro učení bez učitele je „shluková analýza“.

Patří sem např.:

- Metoda k-průměrů (k-means clustering)
- Algoritmus k-nejbližších sousedů (k-nearest neighbor alg.)

Posilované či motivované učení (reinforcement learning)
Jednoduše řečeno se jedná o metodu pokus-omyl. Od učení s učitelem se tato metoda učení liší tím, že neznáme požadovanou hodnotu výstupní funkce, ale každý výsledek algoritmu ohodnotíme. Tato odměna může být pozitivní i negativní a různě veliká. Systém má za úkol maximalizovat součet všech odměn.

Patří sem např.:

- Metoda ADP learning
- Metoda TD learning
- Metoda Q learning

3.4 Metody hraní her – obor teorie her

Metody hraní her jsem nastudoval z [7] a zcela jistě spadají do oboru umělé inteligence. Nechápou však hru v obvyklém slova smyslu: hraní pro zábavu. Jedná se o metody interaktivního rozhodování, tedy rozhodování více osob (subjektů) v situacích, kdy rozhodnutí jednotlivých účastníků ovlivňují dosažený výsledek, jak vlastní, tak výsledek ostatních. Tyto metody se uplatňují v mnoha oblastech lidské činnosti od ekonomie a managementu, přes politologii a dopravu až například po sociologii, biologii a etiku. Vzhledem k rozsahu této problematiky vznikla **samostatná vědní disciplína - teorie her**, spadající do aplikované matematiky, zkoumající racionální lidské chování při rozhodování.

V teorii her je hra přesně stanovený matematický model rozhodovací situace, jejíž výsledek závisí na rozhodnutí alespoň dvou různých subjektů. Herně-teoretické modely se pak snaží tyto konfliktní situace nejen analyzovat, ale především pomocí matematiky se snaží nalézt co nejlepší strategie pro konkrétní účastníky takových konfliktů. Jelikož takové situace můžeme nalézt téměř ve

všech oblastech týkající se našeho života, je, jak již jsem se zmínil, obor aplikací teorie her mimořádně bohatý.

Obor teorie her definuje **mnoho různých pojmů**, jejich rozsáhlý výčet nemá smysl zde uvádět. Zmíním pouze ty důležité v nejbližším vztahu k této práci:

- *Hráč*: účastník hry.
- *Strategie*: varianty možného chování subjektu.
- *Rozhodování*: akt výběru akce z množiny přípustných možností.
- *Strategie hráče*: hráč provádí rozhodování, musí však znát své možnosti – nalezení těchto možností je součást hráčových analytických schopností a také jeho inteligence.
  Pozn.: Ve hře Bang! jsou možnosti hráče všechny kombinace zahrání pro něj dostupných karet v relaci na jaký cíl je zahrát a v jakém pořadí.
- *Užitek*: hráčův výnos ze hry (tahu) je to subjektivní míra na výstupu modelu. Užitek je vztažen ke každé hráčově strategii a na jeho základě může hráč provádět rozhodování.
  Pozn.: V tazích hry Bang! by byl hráčův užitek míra splnění jeho cílů.
- *Preference*: preference hráče je jeden ze vstupů modelu.
  Pozn.: Preference ve hře Bang! mohou být např.: útočit na konkrétního hráče, útočit za každou cenu, zůstat v utajení, apod.

Významné pro teorii her je také **klasifikace her a hráčů** do mnoha různých tříd, na základě kterých se ke hrám poté přistupuje. Uvedu pouze základní výčet bez bližšího popisu:

- hry s *úplnou* a *neúplnou informací*
- hry *statické* a *dynamické*
- hry *s nulovým* a *nenulovým součtem*
- hry *kooperativní* a *nekooperativní*
- hry *opakované* a *neopakované*
- hry *nekonečné* a *konečné*
- *racionalita* hráčů

Z pohledu teorie her je Bang! hra s neúplnou informací, dynamická, nekooperativní, s nenulovým součtem, opakovaná a obecně s racionálními hráči.

3.4.1 Vlastní metody

Mezi nejznámější a nejzákladnější metody umělé inteligence používané ke hraní her patří Minimax a její modifikace Alfa-Beta prořezávání. Tyto algoritmy **prohledávání stavového prostoru** se používají pro hry dvou hráčů s úplnou informací, tzn. když každý hráč má všechny informace o hře a jejím současném stavu.

- Metoda Minimax a její modifikace Alfa-Beta ořezávání

Při řešení obtížných úloh jsou nezbytné **metody rozkladu úlohy na podproblémy**. Tohoto přístupu velmi často využívá i člověk. Nejznámější metodou těchto rozkladů úloh jsou:

- AND-OR grafy

Existuje řada metod umělé inteligence zaměřených na řešení určitých typů úloh. Při řešení složitých problémů technické praxe je však zpravidla nutná kombinace více takových metod,

vytvoření **hybridního systému**, jehož složky řeší konkrétní dílčí úlohy. Velmi důležité jsou také již zmíněné různé heuristické metody a přístupy, které výrazně urychlují proces nalezení optimálního řešení.

3.5 Biologií inspirované metody UI

Tyto metody považuji za jedny z nejmodernějších a nejpropracovanějších metod dnešní doby. Každá z nich by si zasloužila samostatný a velmi rozsáhlý dokument. Takový prostor jim ale v rámci této práce věnovat nemohu. K vytvoření této kapitoly jsem čerpal z [4] a [7], kde se nachází i spousta rozšiřujících informací.

3.5.1 Umělé neuronové sítě

Umělé neuronové sítě jsou jedním z nejmodernějších výpočetních modelů používaných v umělé inteligenci a mají také jednu z nejrozsáhlejších možností aplikace využití. Jejich vzorem je chování odpovídajících biologických struktur v lidském mozku. Skládají se z umělých (nebo také formálních) neuronů, jejichž předobrazem je biologický neuron. Neurony jsou vzájemně propojeny a navzájem si předávají signály a transformují je pomocí určitých přenosových funkcí. Neuron má libovolný počet vstupů, ale pouze jeden výstup a spojují se do vrstev.

Umělé neuronové sítě spadají do metod strojového učení (viz kapitola 3.3), učení je jejich typickou vlastností. Cílem učení neuronové sítě je nastavit síť tak, aby dávala požadované výsledky. V biologických sítích jsou zkušenosti uloženy v dendritech. V umělých neuronových sítích jsou zkušenosti uloženy v jejich matematickém ekvivalentu - váhách. Učení neuronové sítě rozlišujeme na učení s učitelem a učení bez učitele.

Podobně jako v biologických sítích je v učení s učitelem využita zpětná vazba. Neuronové síti je předložen vzor. Na základě aktuálního nastavení je zjištěn aktuální výsledek. Ten se porovná s požadovaným výsledkem a určí se chyba. Poté se spočítá nutná korekce (dle typu neuronové sítě) a upraví se hodnoty vah či prahů, aby se hodnota chyby snížila. Tento proces se opakuje až do dosažení stanovené minimální chyby. Poté je síť naučena a připravena k použití.

Při učení bez učitele se nevyhodnocuje výstup, ten je nám neznámý. Síť dostává na vstup sadu vzorů, které si sama třídí. Buď si vzory třídí do skupin a reaguje na typického zástupce, nebo si přizpůsobí topologii (u sítí s proměnnou topologií) vlastnostem vstupu.

Umělých neuronových sítí existuje řada druhů a jejich celková problematika je velmi obsáhlá.

3.5.2 Genetický algoritmus

Genetický algoritmus, spadající pod genetické programování, je model strojového učení, který odvozuje své chováni od procesu evoluce v přírodě. To se děje vytvořením populace jedinců reprezentovaných chromozomy, v podstatě množinou textových řetězců, které jsou analogií chromozomu v naší DNA. Jedinci v této populaci pak podléhají procesu evoluce.

Genetické algoritmy se používají pro mnoho různých aplikací. Příkladem mohou být mnohorozměrné optimalizační problémy, v nichž řetězec chromozomů může reprezentovat hodnoty různých parametrů, které optimalizujeme. U některých typů úloh však mohou být výpočetně náročnější než specializované algoritmy a metody.

Dnes již existuje celá škála různých variant genetického programování. Následující schéma základního algoritmu nelze vztáhnout na všechny aplikace, ale slouží k pochopení principu metody.

V popisu základního algoritmu používám tyto **termíny**:

- *Jedincem* se rozumí vhodně zakódovaný výstup metody – hledané řešení úlohy. Během výpočtů...
- *Zdatnost* (anglicky fitness) je výsledek *funkce zdatnosti*, která každému jedinci přiřadí hodnotu vyjadřující jeho schopnost řešit původně zadanou úlohu. Čím vyšší zdatnost, tím je jedinec kvalitnější. Hledaný nejlepší jedinec se pozná právě podle toho, že dosáhl nejvyšší zdatnosti. Funkce zdatnosti tak vyjadřuje cíl řešené úlohy.
- *Populace* je soubor jedinců, který se zpracovává v jednom cyklu výpočtu.

Základní algoritmus lze slovy popsat takto:

1. Inicializace: vytvoř nultou populaci (obvykle složenou z náhodně vygenerovaných jedinců).
2. Začátek cyklu: Pomocí určité výběrové metody (zpravidla zčásti náhodně) vyber z populace několik jedinců s vysokou zdatností.
3. Z vybraných jedinců vygeneruj nové vhodným použitím následujících možných operátorů, čímž vznikne další generace:
    - křížení – prohození částí několika jedinců mezi sebou
    - mutace – náhodná změna částí jedinců
    - reprodukce – kopírování jedinců beze změny
4. Vypočti zdatnost těchto nových jedinců.
5. Konec cyklu: pokud není splněna ukončující podmínka, pokračuj od bodu 2.
6. Konec algoritmu: Jedinec s nejvyšší zdatností reprezentuje nejlepší nalezené řešení a je hlavním výstupem algoritmu.

4 Návrh umělé inteligence

V této kapitole popisuji první fázi práce na umělé inteligenci. Jde o rozbor hry Bang! z pohledu jednotlivých metod a přístupů UI a způsob vyřešení nejdůležitějšího podproblému, kterým je odhadování rolí ostatních hráčů.

Metody prohledávání stavového prostoru pro potřeby této práce nemohu využít, jelikož hra Bang! je z velké míry hrou s neúplnou informací. Metody strojového učení vyžadují kvalitní trénovací data, která bohužel nemám a tudíž je také nemohu použít. Biologií inspirované metody UI se k využití ve hře Bang! také příliš nehodí, ať už z důvodu absence trénovacích dat, tak z důvodu problému ohodnocení jednotlivých situací. Hra Bang! je bohužel velmi specifická. Z pohledu algoritmů se jedná pouze o odhadování rolí ostatních hráčů, a v případě rozhodování o svém tahu se jedná pouze o analýzu situace a rozhodnutí na základě vhodného rozhodovacího stromu.

Odhadování rolí protihráčů se dá označit jako klasifikace, jelikož objekty na základě jejich vlastností zařazujeme do konečného počtu tříd, kde objekty jsou hráči, to jak hrají své tahy, jsou jejich vlastnosti a zařazujeme je do tříd Pomocník, Bandita a Odpadlík. Rozhodovací stromy pro konkrétní situace jsem vytvořil na základě svých bohatých zkušeností s touto hrou. K rozhodnutí je vždy zapotřebí analyzovat všechny podstatné informace o aktuální situaci ve hře. Za experta na hru Bang! se nepovažuji, ale dalo by se říci, že se jedná o expertní systém s usuzováním řízeným daty (dopředné řetězení).

Nejdůležitější podproblém hry Bang! je však právě odhadování rolí ostatních hráčů, jelikož na základě této klasifikace se plánuje většina akcí, které chce hráč zahrát. Tyto role jsou pro hráče tajné (nikdo neví, který hráč hraje za jakou roli), veřejná je pouze role Šerifa a celkový počet hrajících hráčů udává počet Banditů, Pomocníků a Odpadlíků ve hře. Každý hráč však zná svoji roli a při zabití jakéhokoliv hráče se jeho role odtajní. S postupem hry se tudíž počet možností a tím i náročnost odhadování rolí snižuje.

Klíčové pro odhadování rolí je pochopitelně sledovat jak ostatní hráči hrají. Je zřejmé, že Pomocník nebude útočit na Šerifa, ale spíše právě na ty hráče, kteří na Šerifa útočí, protože s největší pravděpodobností (pokud hrají racionálně) se jedná o Bandity, apod. Každá role má svůj vlastní cíl (viz kapitola 2.1) a hráči hrají tak, aby tohoto cíle dosáhli a jejich tahy se tudíž jistým způsobem liší.

4.1 Testovací množina dat

Od autora projektu kBang (viz kapitola 1.1) jsem získal záznam herních událostí (dále už jen log soubory) všech her, které se na jeho serveru hrály od spuštění projektu 16. května 2009 až do přerušení 3. listopadu 2009, na výzvu vlastníka autorských práv hry. Log soubory jsou uloženy na přiloženém datovém nosiči, který je součástí této práce a byly využity, z důvodů uvedených níže v části Analýza log souborů, pouze ke statistickému porovnávání úspěšnosti jednotlivých metod mezi sebou.

Každý log soubor obsahuje informace o právě jedné hře. Na začátku je seznam hráčů se všemi potřebnými informacemi, poté záznam o jednotlivých kartách, které si jednotliví hráči berou do začátku hry. Následuje seznam všech herních událostí. Na konci každého log souboru řádně dohrané hry jsou informace o rolích jednotlivých hráčů a kdo vyhrál.

Ukázka tvaru log souboru:

```
onPlayerCreated(id=1, name="VoidAI #25086", role="unknown", character="bart cassidy")
onPlayerCreated(id=2, name="Fiber", role="unknown", character="pedro ramirez")
onPlayerCreated(id=3, name="krabi", role="sheriff", character="sid ketchum")
onPlayerCreated(id=4, name="exoomer", role="unknown", character="kit carlson")
onPlayerDrawFromDeck(player=1, cards=[card(panic,8,diamonds)])
onPlayerDrawFromDeck(player=2, cards=[card(bang,9,diamonds)])
...
onPlayerDrawFromDeck(player=3, cards=[card(panic,12,hearts),card(bang,5,diamonds)])
onPlayerPlayCard(player=3, card=card(duel,8,clubs), targetPlayer=1)
onGameContextChange()
onPlayerRespondWithCard(player=1, card=card(bang,13,hearts))
onGameContextChange()
onPlayerRespondWithCard(player=3, card=card(bang,5,diamonds))
onGameContextChange()
onPlayerRespondWithCard(player=1, card=card(bang,7,diamonds))
onGameContextChange()
onPlayerRespondWithCard(player=3, card=card(bang,12,hearts))
onGameContextChange()
onPlayerRespondWithCard(player=1, card=card(bang,9,clubs))
onGameContextChange()
onPlayerPass(player=3)
onLifePointsChange(player=3, lifePoints=4)
onGameContextChange()
onPlayerPlayCard(player=3, card=card(beer,8,hearts))
onLifePointsChange(player=3, lifePoints=5)
onGameContextChange()
...
onPlayerDied(player=3, causedBy=4)
onGameFinished(player=1, role="outlaw", winner="0")
onGameFinished(player=2, role="outlaw", winner="0")
onGameFinished(player=3, role="sheriff", winner="0")
onGameFinished(player=4, role="renegade", winner="1")
```

V této ukázce je tah jednoho kráče - zahraná karta Duel hráčem č.3 na hráče č.1, jejich vzájemné reakce, ubrání života a poté zahrání karty Pivo.

Analýza log souborů:

Log soubory obsahují celkem 105.669 her, kde každá hra obsahuje zhruba 300 až 1400 herních událostí podle délky hry. Jedno kolo hry (tah všech hráčů) trvá zhruba 50 herních událostí.

Z celkového počtu 105.669 her:

- je 44.541 (42%) řádně nedohraných (předčasně ukončené v průběhu hry, tudíž v nich chybí především informace o rolích hráčů).
- 92.451 (88%) obsahuje původní počítačem ovládané hráče – VoidAI.

Data v tomto tvaru a objemu, v jakém jsem je od autora dostal, se na první pohled jeví jako ideální pro metody strojového učení (viz. kapitola 3.3) – dostatečně splňují vlastnosti dat jako je přesnost, úplnost a interpretovatelnost. Ve fázi předzpracování jsem však objevil jejich obrovskou slabinu a tou je jejich vlastnost nazývaná přidaná hodnota (tj. míra prospěšnosti dat pro řešení dané úlohy). V téměř všech hrách (88% z celkového počtu) je většina hráčů ovládaných počítačem (mají jméno „VoidAI #...“). A jak jsem se již v kapitole 1.1 zmínil, tito původní počítačem ovládaní hráči (dále už jen VoidAI) hrají sice podle pravidel, ale své cíle volí náhodně, tudíž velmi zřídka hrají tak, aby splnili cíl své role.

Druhý problém je ten, že spousta hráčů hru Bang! nikdy před tím nehrála a tudíž své tahy také nevolili tak jak by měli. Důsledek je ten, že naprosto drtivá většina tahů v zaznamenaných hrách jsou tahy neracionálních hráčů. Taková data jsou tudíž naprosto nevhodná pro jakoukoliv metodu strojového učení, protože ze záznamu špatně resp. náhodně hrajících hráčů se žádná metoda a technika nemůže naučit hrát správně. Tato negativní vlastnost těchto dat mě přivedla k závěru, že jediné využití log souborů je ke statistickému porovnávání úspěšnosti jednotlivých metod mezi sebou.

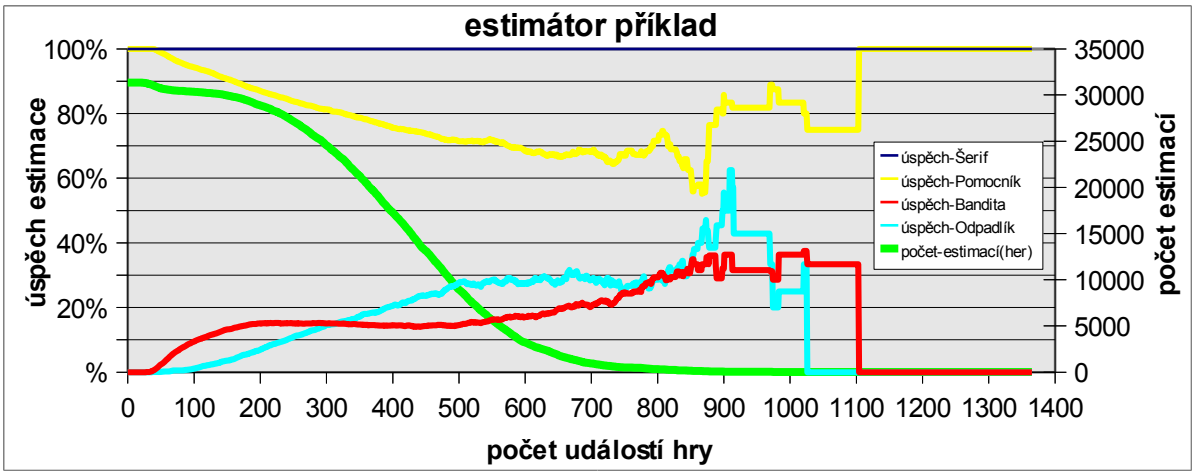
4.2 Modul pro odhadování rolí hráčů - estimátor

Pro potřebu odhadování rolí ostatních hráčů jsem implementoval modul své umělé inteligence (pojmenované „AwareAI“) `estimátor`.

Estimátor prochází hru, událost po události, a vždy se snaží odhadnout role všech hráčů, tudíž jakoby z pohledu Šerifa, který nezná roli žádného hráče (všichni hráči znají pouze svoji roli). Výstup estimátoru je tedy v kterékoliv fázi hry odhadnutí rolí všech hráčů, právě podle toho, jak doposud hráli své tahy. Výhodou takto navrženého estimátoru je jeho přizpůsobivost ke změně strategie hráčů, jelikož může analyzovat všechny tahy od začátku hry nebo např. pouze poslední dva tahy daného hráče, apod.

K testování estimátoru jsem využil log soubory z již odehraných zaznamenaných her (viz kapitola 4.1). Pro potřebu zpracování zaznamenaných her, jsem jako součást estimátoru implementoval jednoduchý parser, který zpracovává jednotlivé herní události uložené v logu zaznamenané hry. Na základě analýzy těchto herních událostí estimátor odhaduje který z hráčů hraje za jakou roli.

Pro testování jednotlivých nastavení estimátoru jsem jeho výstup po každé události zaznamenával, porovnával se skutečnými rolemi daných hráčů a nakonec vypočítával pravděpodobnost, že dané role uhodne správně. Tyto hodnoty jsem vynášel do následně vypadajícího grafu:



Obrázek 4.1: Graf úspěšnosti estimátoru

Hlavní osa Y (levá svislá osa) zobrazuje úspěšnost uhodnutí dané role. Hlavní osa X (vodorovná osa) zobrazuje v jaké události hry (v jak pokročilé fázi hry). V grafu modrá, žlutá, červená a azurová křivka (světle modrá) zobrazuje úspěšnost správného uhodnutí jednotlivých rolí

hráčů (viz legenda) ve vztahu k události hry, ve které se odhaduje (které od začátku uplynuly). Počet pokusů v daných událostech zobrazuje zelená křivka na vedlejší ose Y (pravá svislá osa). Zelená křivka dá se říci zobrazuje poměrem počet her dané délky a slouží k určení oblasti grafu, ve které probíhá většina her, tudíž kde jsou hodnoty v grafu nejpřesnější a nejužitečnější zároveň.

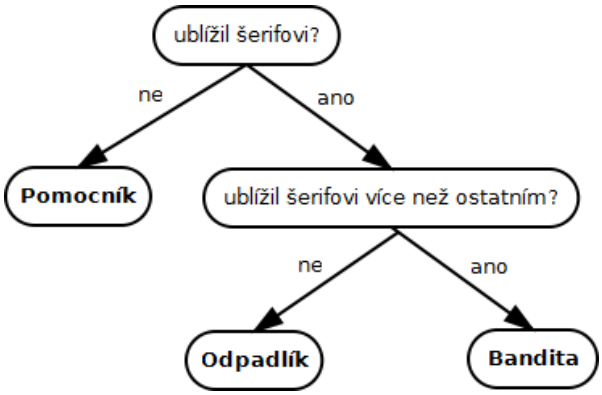
Z grafu na obrázku 4.1 je tedy patrné, že v tomto vzorku dat estimátor v 100. události jednotlivých her provedl 30.000 odhadnutí jednotlivých rolí (zelená křivka) a jeho úspěšnost byla 94% pro Pomocníky (žlutá křivka), 10% pro Bandity (červená křivka) a 1% pro Odpadlíky (azurová křivka). V 300. události jednotlivých her provedl 25.000 odhadnutí jednotlivých rolí a jeho úspěšnost byla 81% pro Pomocníky, 15% pro Bandity a 14% pro Odpadlíky. Tento graf je pouze ilustrující, nezobrazuje výsledky pro žádnou metodu a verzi skutečného estimátoru.

Při testování estimátoru jsem nepracoval s celou množinou více jak 105-ti tisíc zaznamenaných her. Jak jsem již výše zmínil v 88% zaznamenaných her je většina hráčů ovládaná původní umělou inteligencí - náhodně hrající VoidAI. K testování estimátoru jsem využíval pouze vzorek dat, který obsahuje pouze řádně dohrané hry a pouze s lidskými hráči, kde je pravděpodobnost, že hrají racionálně, vyšší než u VoidAI. Tento vzorek dat obsahuje celkem 4.560 her a je uložen na přiloženém datovém nosiči, který je součástí této práce. Všechny testy estimátoru byly provedeny se stejným vzorkem dat.

Vzhledem k absenci trénovací množiny dat (logů her s racionálními hráči), jsem implementaci estimátoru vyvíjel postupně. Jednotlivé výsledky jsem porovnával a následně upravoval jednotlivé parametry. Fázi učení metod jsem tedy provedl ručně na základě mých znalostí hry Bang!.

4.2.1 Použité metody

V této kapitole jsem čerpal z [5]. Základní metodou klasifikace je použití rozhodovacího stromu, což je graf stromové struktury, kde každý vnitřní uzel reprezentuje hodnoty jistého atributu a listy reprezentují třídu, do které bude daný objekt klasifikován. Pro **rozhodovací strom** je klíčové určení atributu (vlastnosti objektu) s nejvyšší rozhodovací schopností. Hodnoty tohoto atributu větví strom v kořeni. Další větvení se provádí atributy s druhou nejvyšší rozhodovací schopností, atd. Ve hře Bang! je vlastností hráče s největší rozhodovací schopností jistě to, zda-li útočil nějakým způsobem na Šerifa. Rozhodovací strom pro určení rolí hráčů ve hře Bang! může vypadat např. takto, kde „ublížit“ je myšleno zahrát karty *Bang!*, *Duel*, *CatBalou* či *Panika!*.



Obrázek 4.2: Příklad rozhodovacího stromu (estimátor 1.1.)

Klasifikaci s využitím neuronových sítí a genetického algoritmu jsem bohužel, vzhledem k absenci trénovacích dat, použít nemohl.

Pravděpodobnost, že hráč, který hraje jistým způsobem, hraje za danou roli, je však podmíněná, a to tím, kolik hráčů s jakou rolí je stále ještě ve hře. Vzhledem k tomuto faktu jsem do implementace estimátoru přidal i **Bayesův vzorec**, který právě podmíněné pravděpodobnosti využívá. estimátor poté pracuje dá se říci jako jednoduchý Bayesovský klasifikátor.

Princip Bayesovské klasifikace:
Nejdříve uvedu následující označení:

- S – značí množinu všech vzorků (zaznamenaných her)
- C_1, C_2, … C_m – značí jednotlivé třídy, do kterých jsou vzory z množiny S klasifikovány. Symbol m udává celkový počet těchto tříd (Pro estimátor jsou 3 a to pro roli Pomocníka, Bandity a Odpadlíka.)
- s_i – značí počet prvků z množiny S, který je klasifikován do třídy C_i, kde $i = 1, \ldots, m$; s_1 tedy například značí počet prvků z množiny S, který je klasifikován do třídy C_1.
- $A_1, A_2, \ldots, A_n$ – značí jednotlivé atributy, pomocí kterých bude prováděna klasifikace. (Pro estimátor je těmito atributy jak hráč hrál své tahy.)
- $X = (x_1, x_2, \ldots, x_n)$, kde x_i je hodnota atributu A_i pro všechna $i = 1, \ldots, m$, značí testovací vzorek, který má být klasikovým do nějaké třídy.

Potom $P(C_i|X)$ pro libovolné $i = 1, \ldots, m$ udává podmíněnou pravděpodobnost toho, že libovolný vzorek padne do třídy C_i, pokud víme, že vzorek má atributy shodné s prvkem X. Jinými slovy, jaká je pravděpodobnost, že prvek X patří do třídy C_i. Úkolem je tedy najít takové i, pro které je hodnota $P(C_i|X)$ největší. Prvek potom bude klasifikován právě do této třídy C_i s největší hodnotou $P(C_i|X)$, neboť je největší pravděpodobnost, že prvek X patří právě do této třídy.

Podle Bayesova vzorce platí:

$$P(C_i|X) = \frac{P(X|C_i)\,P(C_i)}{P(X)}$$

Pro jisté i bude tedy výraz $P(C_i|X)$ maximální právě tehdy, když bude maximální výraz $P(X|C_i)P(C_i)$, neboť $P(X)$ je pro konkrétní vzorek X konstantou.

Estimátor s použitím Bayesova vzorce tedy klasifikuje do tříd $C_{Pomocník}$, $C_{Odpadlík}$ a $C_{Bandita}$. Každý vzorek (jednotlivého hráče) ohodnotí pravděpodobností s jakou patří do jednotlivých tříd (tj. $P(X|C_i)$), podle toho jak hrál své tahy (jaké jsou hodnoty jeho atributů A). Poté zjistí kolik jakých rolí je stále ve hře a vypočítá $P(C_i)$, tedy pravděpodobnost, že libovolný vzorek patří právě do třídy C_i. Nakonec porovná výrazy $P(X|C_i)P(C_i)$, tedy pravděpodobnosti, že hráč patří do dané třídy a klasifikuje hráče právě do třídy, pro kterou je hodnota výrazu největší.

4.2.2 Výsledky jednotlivých metod

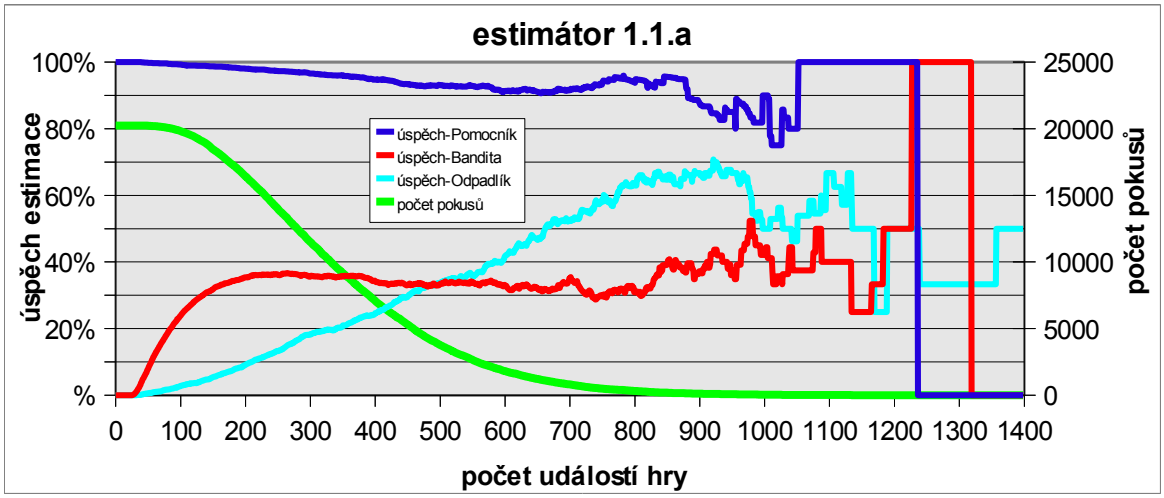
Tato kapitola obsahuje výsledky pro různá nastavení použitých metod včetně příslušných grafů, popisu a zhodnocení.

Všechna měření byla prováděna se stejným vzorkem dat.

Každý estimátor je označen verzí *X.Y.Z*, kde *X* představuje použitou metodu, *Y* různé nastavení pro danou metodu a *Z* počet zkoumaných událostí (jak jsem již zmínil, 1 kolo hry je zhruba 50 událostí) do historie. Hodnota „*a*“ v části *Z* znamená, že se zkoumají všechny události od začátku hry.

Jednotlivá nastavení metod (značení *Y*) pro všechny metody korespondují, aby se výsledky různých metod daly porovnat mezi sebou. Tudíž např. estimátor *2.1.a* založený na Bayesově vzorci využívá pro vyhodnocení atributů hráčů (pravděpodobnost příslušnosti do jednotlivých tříd) stejný rozhodovací strom jako estimátor *1.1.a*, ale pro klasifikaci využívá navíc podmíněnou pravděpodobnost v Bayesově vzorci, kdežto estimátor *1.1.a* klasifikuje pouze podle rozhodovacího stromu.

1. **estimátor *1.1.a***



Obrázek 4.3: Graf úspěšnosti estimátoru 1.1.a

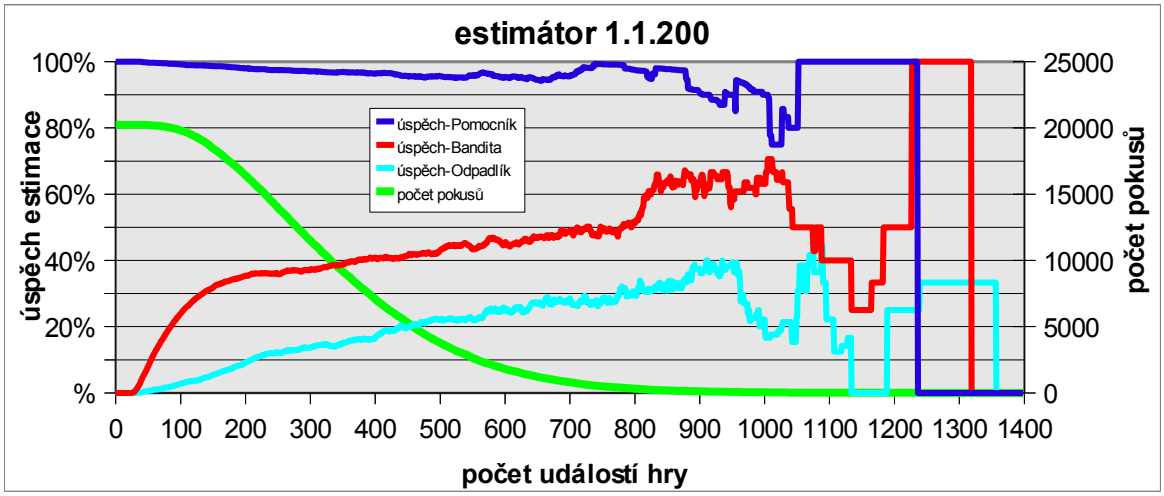
Metoda: Klasifikace probíhá pouze na základě rozhodovacího stromu z obrázku 4.2, viz výše, kde „ublížit“ je myšleno zahrát karty *Bang!*, *Duel*, *CatBalou* či *Panika!*. Zpracovává všechny události od začátku hry.

Popis: Takto nastavený rozhodovací strom velmi dobře odhaduje Pomocníka, jelikož ten, kdo na Šerifa od začátku hry nevystřelil, bývá právě pouze Pomocník. Úspěch správného odhadu Bandity je od 3. kola 30-40%, jelikož Bandité ne vždy střílí na Šerifa více než na ostatní, aby se neprozradili, či na Šerifa nemají dostřel. Úspěšnost odhadu Odpadlíka je rostoucí a 50% překročí až po 13. kole hry, což je téměř na konci většiny her, ale Odpadlík se projevuje právě až na konci hry.

Výhoda: Velmi dobrý odhad Pomocníků.

Nevýhoda: Špatný odhad Banditů a Odpadlíků na začátku hry, než se projeví.

2. **estimátor *1.1.200***



Obrázek 4.4: Graf úspěšnosti estimátoru 1.1.200

Metoda: Klasifikace probíhá pouze na základě rozhodovacího stromu z obrázku 4.2, viz výše, ale zpracovává se pouze posledních 200 událostí (4 kola hry).

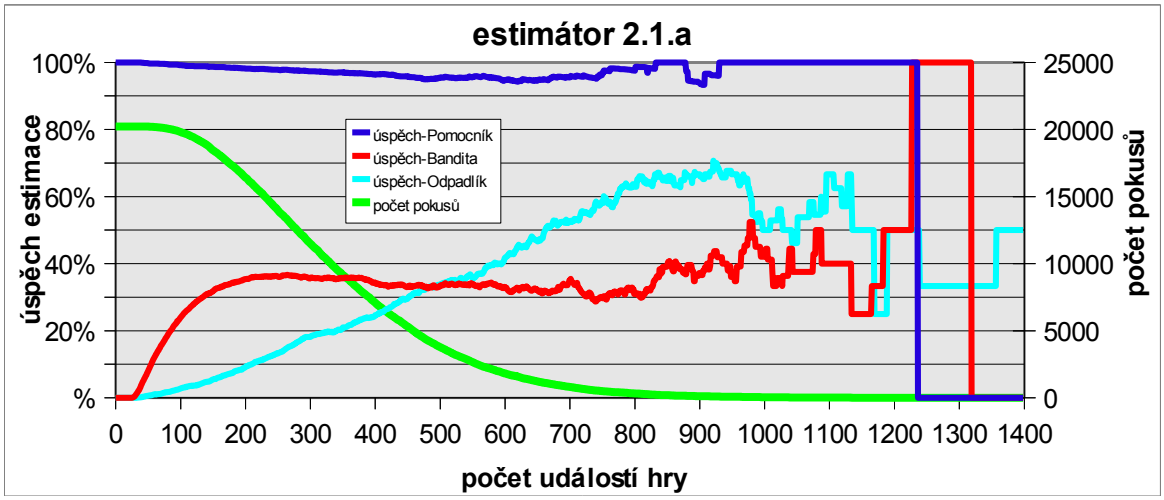
Popis: Při zkoumání pouze posledních 4. kol hry stejný rozhodovací strom podává podobné výsledky jako při zkoumání všech událostí od začátku hry. Odhad Pomocníka je stejný. Úspěšnost odhadu Banditů už není od 3. kola stabilní, ale naopak rostoucí, což je zlepšení. Kdežto růst úspěšnosti odhadu Odpadlíka je mírnější.

Výhoda oproti *1.1.a*: Lepší odhad Banditů s postupem hry.

Nevýhoda oproti *1.1.a*: Horší odhad Odpadlíka s postupem hry.

estimátory používající Bayesovskou klasifikaci:

3. **estimátor *2.1.a***



Obrázek 4.5: Graf úspěšnosti estimátoru 2.1.a

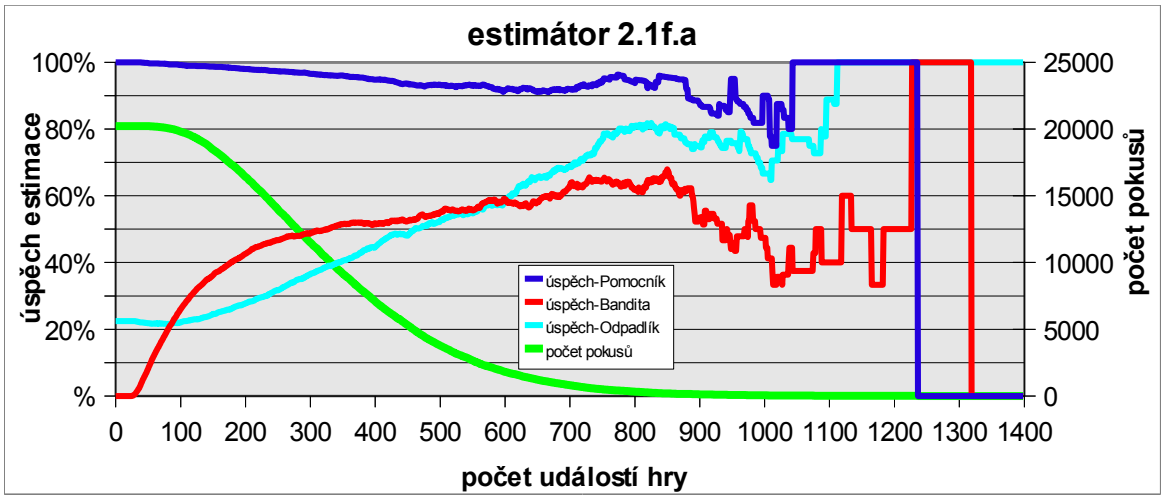
Metoda: Klasifikace probíhá na základě rozhodovacího stromu z obrázku 4.2, viz výše a zároveň využívá podmíněné pravděpodobnosti pomocí Bayesova vzorce. Zpracovávají se všechny události od začátku hry.

Popis: Výsledky jsou totožné s *1.1.a*! Rozhodovací strom z obrázku 4.2, totiž diskretizuje atributy příliš hrubým způsobem a klasifikuje vždy ostře právě do jedné množiny, což znemožňuje využít podmíněné pravděpodobnosti Bayesovým vzorcem.

4. **estimátor *2.1.200***

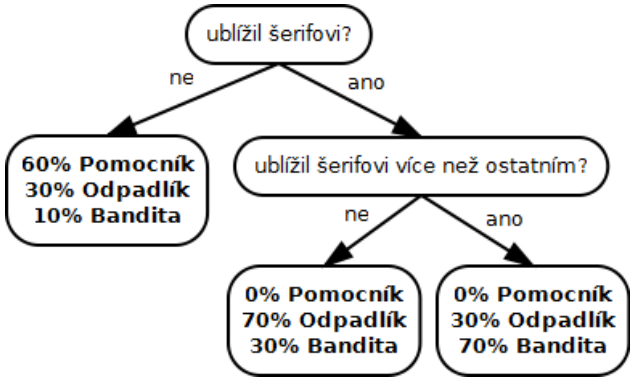
   Graf pro toto nastavení estimátoru je stejný jako pro estimátor *1.1.200*, a to ze stejného důvodu proč jsou výsledky pro estimátor *2.1.a* stejné jako pro *1.1.a*.

5. **estimátor *2.1f.a***



Obrázek 4.6: Graf úspěšnosti estimátoru 2.1f.a

Metoda: Klasifikace probíhá na základě rozhodovacího stromu z obrázku 4.7, viz níže, který bere v úvahu více možností klasifikace hráče s danými atributy a zároveň využívá podmíněné pravděpodobnosti pomocí Bayesova vzorce. Zpracovávají se všechny události od začátku hry.



Obrázek 4.7: Rozhodovací strom estimátoru 2.1f

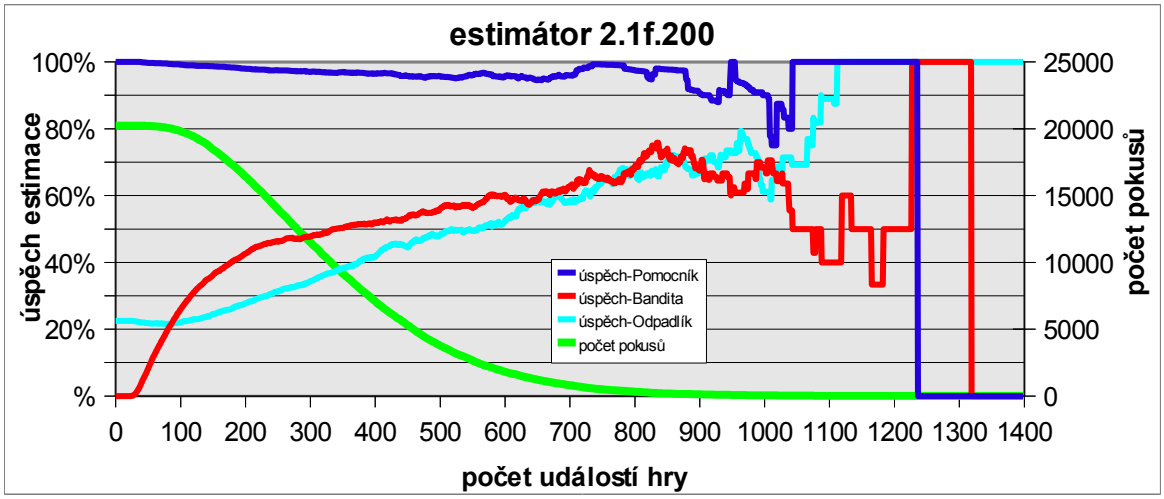
Popis: Takto nastavený strom v Bayesově vzorci využívá i podmíněné pravděpodobnosti a tudíž jsou výsledky lepší než pro pouhý rozhodovací strom. Úspěšnost odhadu Bandity je po 7. kole

vyšší než 50% a roste až na více jak 60%. Úspěšnost odhadu odpadlíka je 50% po 9. kole a dále roste až 80%. Odhad Pomocníka je opět velmi dobrý a drží se nad 90%.

Výhoda oproti *1.1.a*: Znatelně lepší odhad Banditů a Odpadlíka.

Nevýhoda: Špatný odhad Banditů a Odpadlíků na začátku hry, než se projeví.

6. **estimátor *2.1f.200***



Obrázek 4.8: Graf úspěšnosti estimátoru 2.1f.200

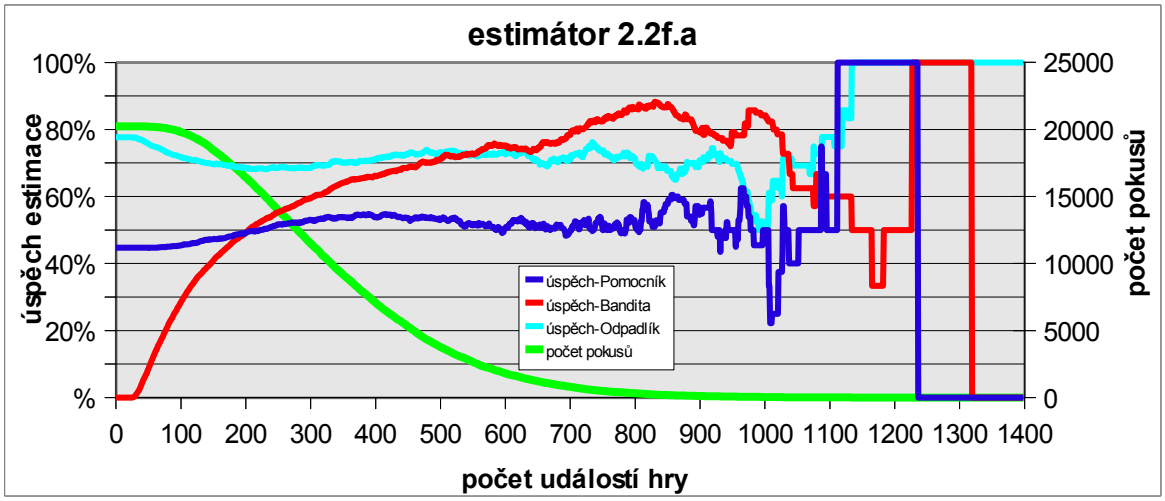
Metoda: Klasifikace probíhá stejně jak u *2.1f.a* (na základě rozhodovacího stromu z obrázku 4.7 a zároveň využívá podmíněné pravděpodobnosti pomocí Bayesova vzorce). Zpracovává se pouze posledních 200 událostí (4 kola hry).

Popis: Výsledky jsou velmi podobné jako pro *2.1f.a*.

Výhody oproti *2.1f.a*: Lepší odhad Banditů na konci dlouhých her, jelikož nejspíš až tehdy začali střílet na Šerifa.

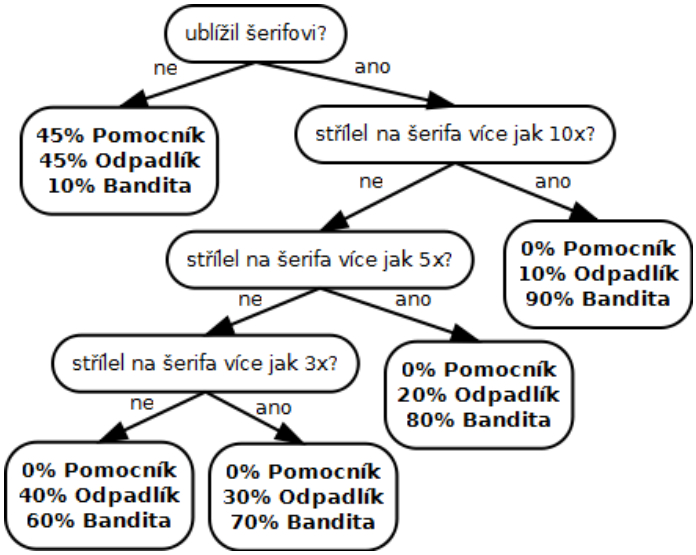
Nevýhody oproti *2.1f.a*: Horší odhad Odpadlíka na konci dlouhých her.

7. **estimátor *2.2f.a***



Obrázek 4.9: Graf úspěšnosti estimátoru 2.2f.a

Metoda: Klasifikace probíhá na základě rozhodovacího stromu z obrázku 4.10, viz níže, který bere v úvahu více možností klasifikace hráče s danými atributy, využívá podmíněné pravděpodobnosti pomocí Bayesova vzorce a inspiruje se principem fuzzy množin – nediskretizuje atributy tak hrubě. Zpracovávají se všechny události od začátku hry.



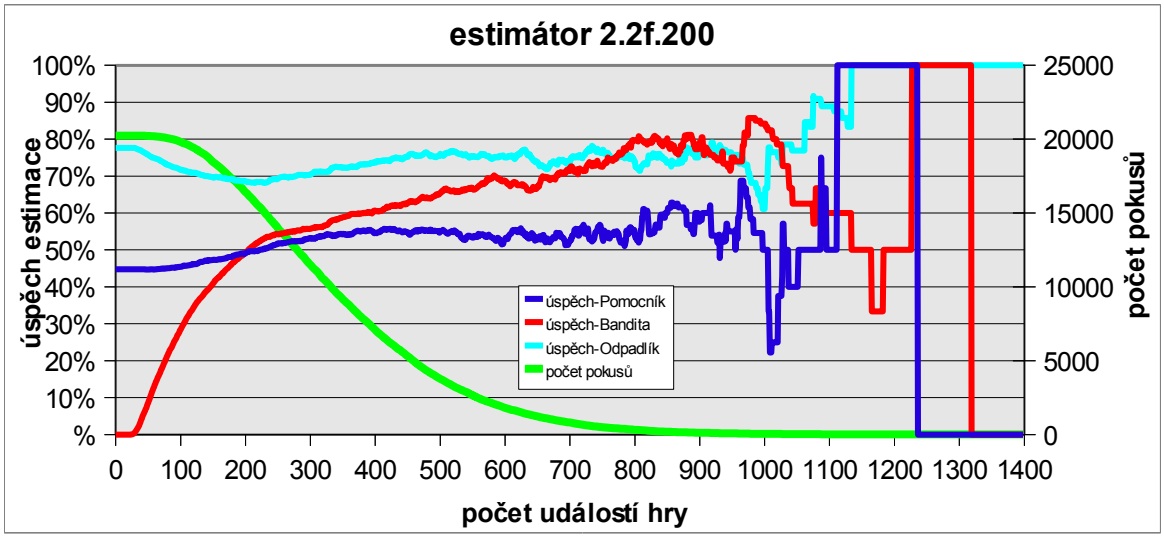
Obrázek 4.10: Rozhodovací strom estimátoru 2.2f inspirovaný fuzzy množinami

Popis: Úspěšnost odhadu role Odpadlíka je poměrně stabilně kolem 70% po celou dobu hry. Úspěšnost pro Pomocníka je zhruba o 15% nižší také po celou dobu hry. Úspěšnost odhadu pro Bandity je od 4. kola 50% a plynule roste až k 85%.

Výhody: Od 4. kola jsou úspěšnosti odhadu pro všechny role více jak 50%. Nejlepší odhad Banditů a Odpadlíka ze všech metod.

Nevýhody: Horší odhad Pomocníka.

8. estimátor *2.2f.200*



Obrázek 4.11: Graf úspěšnosti estimátoru 2.2f.200

Metoda: Klasifikace probíhá stejně jak u *2.2f.a* (na základě rozhodovacího stromu z obrázku 4.10 a zároveň využívá podmíněné pravděpodobnosti pomocí Bayesova vzorce a inspiruje se principem fuzzy množin). Zpracovává se pouze posledních 200 událostí (4 kola hry).

Popis: Výsledky jsou velmi podobné jako pro *2.2f.a*.

Výhody oproti *2.2f.a*: O něco lepší odhad pro Pomocníka a Odpadlíka.

Nevýhody oproti *2.2f.a*: O něco horší odhad Banditů.

4.2.3 Shrnutí výsledků

Před tím, než zhodnotím použité metody a jejich nastavení ještě bych rád připomenul, že testování estimátoru probíhalo z pohledu Šerifa, tudíž z nejhorší možné pozice ve hře. Všichni ostatní hráči mají při odhadech o jednu velmi užitečnou informaci navíc, a to jaká je jejich vlastní role. Proto se dá předpokládat, že z pohledu všech ostatních hráčů budou odhady rolí ostatních hráčů mít znatelně vyšší úspěšnost.

Při klasifikování pouze na základě rozhodovacího stromu (estimátor *1.1.z*) nebyly výsledky příliš uspokojující. Dá se říci, že odhad Banditů nepřesáhl 50%, což je velmi špatné nejen z důvodu, že odhadnout Bandity potřebuje Šerif ve své hře prioritně.

Klasifikátor rozšířený o podmíněnou pravděpodobnost, využívající Bayesův vzorec, podával dle očekávání mnohem lepší výsledky. Úspěšnost odhadů pro všechny role měla stoupající tendenci, což odpovídá reálné situaci ve hře, kde s každým vyřazeným hráčem se situace postupně vyjasňuje. Při použití principu fuzzy množin pro ohodnocení atributů hráčů se odhady pro všechny role, dá se říci, zprůměrovaly a dosahovaly velmi dobrých výsledků. Od 4. kola hry byla úspěšnost odhadu pro všechny role více jak 50% a s pokračující hrou se správný odhad Banditů vyšplhal až k 85% a Odpadlíka k 70%, přičemž odhad Pomocníka zůstával stabilní a to kolem právě 50%. Je zřejmé, že takto nastavený estimátor (*2.2f.a*) si občas pletl Pomocníka s Odpadlíkem, ale i tato vlastnost odpovídá skutečnosti v každé reálné hře a nedá se jí spolehlivě vyhnout.

Všechna nastavení estimátoru podávala srovnatelné výsledky jak při zpracovávání všech událostí hry od jejího začátku, tak při zpracování pouze posledních 4 kol. Toto rozšíření má význam pouze ve hře, kde některý z hráčů blafuje a v průběhu hry znatelně změní svoji strategii. Toto je vlastnost výhradně pokročilých hráčů, kteří dokáží správně vyhodnotit situaci z pohledu jiných hráčů a hrát právě tak, jak se jim to vůči jistému hráči v tu chvíli nejvíce hodí. Jak jsem se již zmínil v kapitole 4.1 pokročilých hráčů v „zaznamenaných hrách bylo minimum, což se i tímto potvrdilo.

Součástí mé umělé inteligence („AwareAI“) v projektu KBang bude tedy nejlepší z testovaných estimátorů, což je již zmíněný *2.2f*.

5 Implementace

Tato kapitola popisuje veškeré změny, které jsem provedl v původním zdrojovém kódu projektu a způsob implementace nové umělé inteligence – třídy `AwareAI`.

Měla by sloužit především autorovi původního zdrojového kódu pro usnadnění procesu převzetí mé umělé inteligence, ale i všem ostatním, kteří by chtěli v projektu pokračovat.

Podrobnější, programátorský popis všech metod je součástí komentářů zdrojového kódu, ze kterého lze snadno vygenerovat programátorskou příručku např. pomocí aplikace *Doxygen*[1]. Popis implementace projektu KBang je součástí technické zprávy bakalářské zprávy autora projektu, která je uložena na přiloženém datovém nosiči.

5.1 portování KBang pro *Microsoft® Visual Studio®* a *Microsoft® Windows®*

Původní projekt KBang autora projektu Michala Čevory (viz kapitola 1.1) nebyl vyvíjen pod operačním systémem *Microsoft® Windows®* (dále už jen *Windows®*). Tato podkapitola obsahuje veškeré změny, které byly provedeny v původním zdrojovém kódu ať už z důvodu přeložitelnosti pod *Windows*, tak z důvodu implementace nové umělé inteligence – AwareAI.

Vzhledem k tomu, že převážně jsem uživatelem právě majoritního operačního systému *Windows*, nejdříve bylo nutné založit „workspace“ v nějakém vývojovém prostředí. KBang je implementován v objektově orientovaném jazyce C++ a používá převážně prostředky knihovny *Qt4*, díky čemuž je do jisté míry multiplatformní, i když nějaké menší nekompatibility se v *Qt* vyskytují (`QString` vs. `std::iostream`, `stdio` vs. `QTextStream`, apod.). Jako vývojové prostředí tohoto projektu jsem zvolil *Microsoft® Visual Studio®* (dále už jen *Visual Studio®*), pro jeho nespočet výhod.

KBang je rozdělen do dvou projektů a to aplikace *BangClient* a *BangServer*, obě používají společnou knihovnu *common*.

Abych zprvu mohl projekt vůbec přeložit, bylo potřeba poměrně hodně upravit zdrojové kódy KBangu, protože neodpovídaly snad žádnému známému standardu a byly z velké části nepřeložitelné.

Zde je **seznam chyb v původním zdrojovém kódu projektu**, které byly opraveny:

- aplikace *Client*:
  1. `cardmovement.cpp` – chyby způsobené ambiguitou konverze z `float` na `int` v některých z matematických funkcí
  2. `common.cpp` – funkce `randomToken()` je již definována jinde (`common\util.cpp`), byla odstraněna
  3. `serverconnection.cpp` – oprava některých tříd předdefinovaných jako `struct`, ale přitom jsou typu `class`
- složka *common*:
  4. `util.cpp` – definováno statické pole s rozměrem v (nekonstantní) proměnné

1 Doxygen – Source code documentation generator tool [online]. [cit. 20.5.2010]. Dostupné z URL: <http://www.stack.nl/~dimitri/doxygen/>

® = registrovaná ochranná známka společnosti Microsoft Corporation

- aplikace *Server*:
    5. Ve všech souborech `cardbang.cpp`, `cardbang.h`; `cardbarrel.cpp`, `cardbarrel.h`; `cardbeer.cpp` `cardbeer.h`; `carddrawcards.cpp`, `carddrawcards.h`; `cardduel.cpp`, `cardduel.h`; `carddynamite.cpp`, `carddynamite.h`; `cardgeneralstore.cpp`, `cardgeneralstore.h`; `cardhorse.cpp`, `cardhorse.h`; `cardjail.cpp`, `cardjail.h`; `cardmultishoot.cpp`, `cardmultishoot.h`; `cardtaker.cpp`, `cardtaker.h`; `weaponcard.cpp`, `weaponcard.h` – Návratový typ funkce `ReactionCard::play()` je `bool`. Zde zděděná funkce se ho pokouší měnit na `void`. (změněno na `bool`)
    6. `cardjail.cpp`, `cardjail.h` – nesprávné předávání ukazatelů na funkce, pokus o jejich referencování
    7. `cardplayable.cpp` – chybělo `#include "game.h"`. V konstruktoru třídy `PlayingCard` chyběly hodnoty povinných parametrů
    8. `checkresulthandler.h` – v deklaraci třídy chyběla hlavička konstruktoru `CeckResultHandler()`, jejíž tělo však v odpovídajícím souboru `.cpp` existovalo.
    9. `console.cpp`, `console.h` – vyřešena nekompatibilita mezi `QTextStream` a *Windows* (známá chyba v *Qt*), použito `std::cin` a `std::cout`, pro tisknutí do `cout` vyřešena konverze z `QString` do ansi (latin1)
    10. `gamelogger.cpp`, `gamelogger.h` – vyřešena nekompatibilita `QString` a `cout`
    11. `main.cpp` – opraveno aby využívalo opravenou konzoli
    12. `playingcard.cpp`, `playingcard.h` – vyřešen zmatek mezi špatně přetíženými `void play()` a `bool play()`. přidána chybějící `friend` třída `CardPlayable`
    13. `publicgameview.h` – obsahovalo funkce deklarované jako `inline`, i když tělo nebylo uvedeno a činilo je tak nepoužitelnými

Poté bylo nutné přeložit *Qt Designer* (`*.ui`) soubory. Při pokusu o vytvoření globálního pravidla *Visual Studio*® odmítalo klient linkovat. Vzhledem k nízké nutnosti upravovat jakékoliv soubory `*.ui`, se překládají pomocí dávky (`build_client_qt.bat`, příkaz `uic`), stejně bylo vyřešeno i **kompilování *meta-object* souborů** (`build_client_qt.bat`, `build_common_qt.bat` a `build_server.bat` příkazem `moc`). Dalším problémem byla ***Qt resource*** sekce. Nakonec se podařilo přepsat (po vzoru oficiálního *Windows* klienta) načítání grafiky na načítání ze souborů namísto z *resource*.

- apliakce *Client*:
    1. vygenerovány *meta-object-compiler* soubory
    2. zkompilovány `*.ui` soubory (Nepodařilo se nastavit přímo v *Visual Studiu*, používá se externí dávka. Jde o soubory, které stejně nechceme měnit.)
    3. `joingamedialog.cpp`; `playerwidget.cpp`; `opponentwidget.cpp` – úprava adres ikonek *GUI* z *Qt resource* na soubory z disku (v `opponentwidget.cpp` se také nachází Šerifova hvězda)
- složka *common*:

® = registrovaná ochranná známka společnosti Microsoft Corporation

4. vygenerovány *meta-object-compiler* soubory

- aplikace *Server*:
  5. vygenerovány *meta-object-compiler* soubory

Dalším krokem bylo vytvoření nových souborů `AwareAI.cpp` a `AwareAI.h`, obsahující třídu mé nové umělé inteligence. Jejich obsah byl na začátku odvozen od původní VoidAI a následně byl podle potřeby modifikován.

Nakonec bylo nutné mírně připravit a **vylepšit server pro potřeby nové AI**. Původní design nebyl příliš šťastný, protože původní AI nemá spoustu informací o hře, jako informace o počtu spoluhráčů, počty hráčů různých rolí ve hře, vzdálenosti mezi hráči (pro účely výpočtu dostřelu) a podobně. Originální AI tyto problémy řeší slepým zkoušením možností, až dokud se nepodaří zahrát platný tah, což je pro jakoukoliv praktickou implementaci samozřejmě nepoužitelné. Také byla dopsána alternativní cesta nahlížení AI na historii hry, a to skrz seznam událostí, uchovaných v třídě `GameLogger`.

- složka *common*:
  1. `gameenums.h` – přidány některé nové položky do výčtu `GameMessageType` pro události v logu, které předtím v původním projektu nebyly pojmenovány (`GAMEMESSAGE_PLAYERCREATED`, ...)
  2. `parser\parserstruct.h` – do struktury `GameMessage` přidány některé nové položky, které se vyskytují v logu, ale v `GameMessage` nejsou uloženy (karta výzvy, počet životů hráče který byl střelen a výsledek výzvy)
- aplikace *Server*:
  3. `consolecommand.cpp` – **přidány nové příkazy serveru** `set-ai-delay` a `go-ai`; Tyto příkazy slouží k vytvoření „rychlé“ lokální herní smyčky pro účely testování AI. Příkaz `set-ai-delay` má jeden parametr a tím je číselná hodnota v milisekundách zpoždění reakce hráčů ovládaných serverem – AI. Příkaz `go-ai` má parametry dva: první je počet hráčů a druhý je kolik her se zadaným počtem hráčů (AI) má server sekvenčně vytvořit a spustit. Typická sekvence příkazů do konzole spuštěného *BangServeru* pro otestování AI je tedy:

     ```
     > set-ai-delay 0      //nastaví zpoždění reakcí AI na 0 milisekund
     > go-ai 5 1000        //spustí sekvenčně 1000 her o 5-ti hráčích
     ```

  4. `game.cpp`, `game.h`
     - změna používání hráčů ovládaných serverem z třídy `VoidAI` na `AwareAI`
     - přidání funkce `getDistance()`, pracující pouze s třídou `publicPlayerView` (ne s celou třídou `Player`), pro AI aby měla k dispozici vzdálenosti mezi jednotlivými hráči
     - úprava funkce `startGame()`, že pokud je startující hráč `NULL`, vybere se automaticky, toto nebylo implementováno (využívá smyčka testování AI)
     - přidán příjemce (getter) `getDistance()` pro třídu `GameLogger`
     - přidány příjemci (getters), zjišťující počty jednotlivých rolí hráčů ve hře (`getGoodGuys_Count()`, `getSheriff_Count()`, `getVice_Count()`, `getOutlaw_Count()` a `getRenegade_Count()`)

5. `gamelogger.cpp`, `gamelogger.h`
   - přidána mpžnost pro zjištění jména ukládaného souboru s logem hry
   - `GameLogger` si uchovává všechny dosavadní události hry v paměti, ty jsou pak k dispozici AI k nahlédnutí (používá je estimátor rolí). Využití tohoto seznamu lze předpokládat u více druhů AI a proto byl implementován přímo v třídě `GameLogger` a ne v třídě `AwareAI`. Dalším argumentem je, že takto existuje pouze jedna kopie tohoto seznamu, jinak by jich existovalo tolik, kolik je AI hráčů.
6. `gameserver.cpp`, `gameserver.h` – přidána možnost pro vytváření instance serveru mimo implicitní *`singleton`* (pro hry k testování AI). Konzole totiž běží v jiném vlákně než server a není tedy možné volat server, protože alokace, co server udělá, jsou v paměťové režii jiného vlákna a při pokusu o jejich re/de-alokaci konzolí by došlo k chybě.

5.2 Implementace umělé inteligence - třídy AwareAI

Obsah třídy byl na začátku odvozen od původní třídy umělé inteligence `VoidAI` a následně byl podle potřeby modifikován. Tato kapitola popisuje chování AI hráčů, ovládaných serverem. Nemá význam popisovat jaké změny jsem provedl vzhledem k původní třídě `VoidAI`, jelikož jsem změnil, dá se říci, naprosto vše. Jediné co jsem ponechal, je výhodné používání výjimek při hraní karet, což mnoho problémů znatelně usnadní a rapidně sníží možnost pádu aplikace či porušení pravidel hry, způsobeném špatně zahranou kartou, při opominutí ošetřit určitou situaci.

Hlavní funkcionalita třídy je v metodě `requestWithAction()`. Tato metoda je zavolána hrou vždy, když se od AI hráčů ovládaných „serverem“ očekává nějaká akce či reakce na události hry (např. když je na tahu, má zareagovat na střelu mířenou na něj, apod.). Tělo metody je rozvětveno na části, podle hodnoty atributu `m_requestType`, udávajícího co je od hráče očekáváno. Možné hodnoty atributu jsou celkem čtyři a to:

1. `REQUEST_DRAW`
   - Událost před začátkem vlastního tahu hráče, tedy **fáze 1.: „vzití si dvou karet“** (viz. kapitola 2.1.2). Standardně si hráči berou 2 karty z lízacího balíčku do ruky, pokud nevyužijí speciální schopnosti některých postav, a poté začínají svůj tah.
   - Pokud má hráč před sebou vyloženou kartu *Dynamit* či *Vězení*, pomocí výjimek je ošetřeno, aby si nejdříve lízl na tyto karty, jak říkají pravidla, a až poté si může standardně brát karty z lízacího balíčku.
   - Další nestandardní situace nastane v případě, že hráč hraje za postavu *Pedro Ramirez*, *Kit Carlson*, *Jesse Jones* či *Black Jack*. Tyto postavy mají před začátkem tahu braní karet odlišné, mohou využít své speciální schopnosti.
     V případě postavy *Pedro Ramirez* hráč nejdříve zkontroluje kartu na vrcholu odhazovacího balíčku a pokud se mu hodí (viz kapitola 5.2.1 část Vytvoření množiny karet, které chci), tak si ji vezme namísto první karty lízacího balíčku.
     Postava *Kit Carlson* si bere tři karty z lízacího balíčku a vybírá si z nich dvě, podle toho které se mu více hodí. Tento výběr je řešen jako reakce na výzvu, stejně jako

Hokynářství (viz. níže), jen karty nejsou pro ostatní viditelné a hráč si bere dvě ze tří karet, místo jedné ze zbylých, jak je tomu u *Hokynářství*.
Postava *Jesse Jones* si první kartu namísto z lízacího balíčku bere od hráče, který pro daný tah je jeho cíl, a kterého se snaží oslabit či zabít.
Vlastnost postavy *Black Jack* je hrou řešena hrou automaticky, nedá se totiž nijak hráčem ovlivnit.

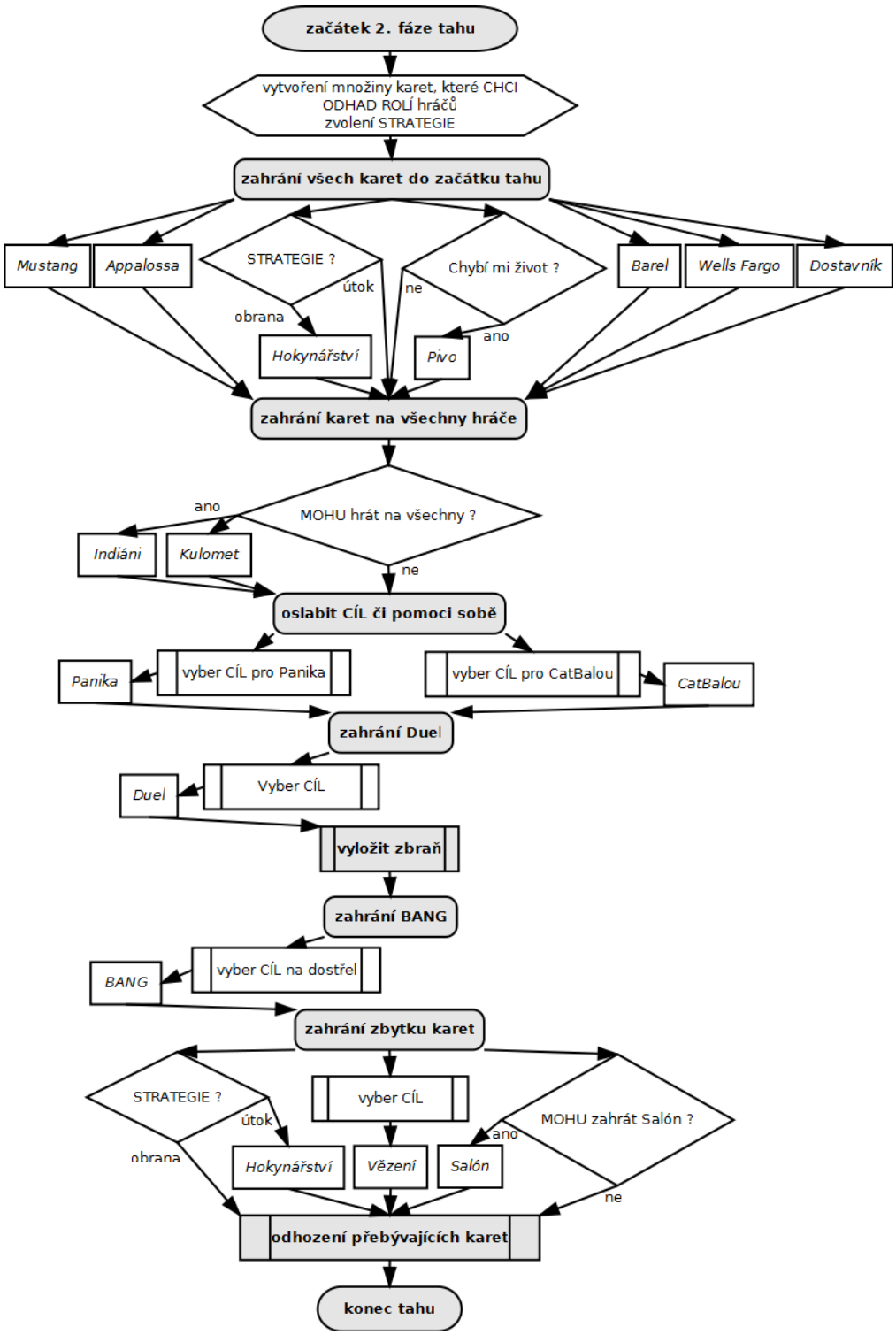
2. `REQUEST_PLAY`:
   - Jedná se o událost vlastního tahu hráče, nazývaná **fáze 2.: „zahrání herních karet“** (viz. kapitola 2.1.2). Tato událost je vyvolána po skončení fáze 1.: „vzití dvou karet“ (`REQUEST_DRAW`). Na konci této události je řešena i fáze 3.: „odhození přebývajících karet“. Popisu chování AI hráčů v této, rozhodně nejdelší a nejdůležitější, části tahu se věnuje níže podkapitola 5.2.1.
3. `REQUEST_RESPOND`:
   - Popisuje chování hráče na výzvu. Tato výzva v základním balíčku hry může být *Hokynářství*, reakce na *BANG!* a *Duel* mířený na hráče či reakce na kartu *Indiáni!* a *Kulomet*.
   - V případě výzvy na zahrané *Hokynářství* se hráč podívá, která z karet se mu nejvíce hodí (viz kapitola 5.2.1 část Mohu zahrát tuto kartu?) a právě tu si vezme.
   - Při reakci na karty *BANG!*, *Duel*, *Indiáni!* a *Kulomet* se hráč nejdříve zkusí schovat za vyložený *Barel*. To samozřejmě vyjde pouze v případě, že reaguje na *BANG!* či *Kulomet* a lízne si na *Barel* srdcovou kartu. Když *Barel* nevyjde, resp. ho žádná karta ze stolu nezachrání, hledá použitelnou kartu v ruce. V případě, že reaguje na kartu *Duel* či *Indiáni!*, hledá na reakci kartu *BANG!*. V případě reakce na *BANG!* a *Kulomet* hledá kartu *Vedle!*.
   - Nestandardní situace nastává v případě postavy *Lucky Duke*, který si na *Barel* může líznout dvě karty a vybrat si tu, která se mu více hodí. Další nestandardní situace nastává v případě, že hráč reaguje na kartu *BANG!* zahranou postavou *Slab the Killer*. Na jeho *BANG!* totiž jsou potřeba dvě karty *Vedle!*, proto pokud hráč nemá dvě karty *Vedle!* (v případě postavy *Calamity Janet*, jakoukoliv kombinaci dvou karet *Vedle!* a *BANG!*), rovnou si bere život.
4. `REQUEST_DISCARD`:
   - Popisuje chování hráče na událost při výzvě na odhození karty z ruky. Tato událost nastane vždy, když někdo na hráčovu kartu v ruce zahraje kartu *CatBalou* nebo *Panika!*. Kterou kartu iniciující hráč označil (ač rubem vzhůru) nemá význam. Karta je vždy vybrána náhodně ze všech karet, které má hráč v ruce, tak jak to správně má být.

5.2.1 Zahrání herních karet – 2. fáze tahu

Proces zahrání herních karet během vlastního tahu nejlépe popíše následující vývojový diagram na obrázku 5.1, zobrazující průběh celého tahu a pořadí hraných karet.

Sémantika vývojového diagramu byla dodržena, tudíž k popisu jen krátce. Začátek a konec procesu vlastního tahu je označen symbolem elips. V symbolu šestiúhelníku jsou vyjmenovány důležité akce přípravy, provedené před samotným hraním karet. V obdélnících jsou akce zahrání

konkrétních karet. Oválné obdélníky značí začátky jednotlivých částí hlavního procesu. Kosočtverce jsou větvení procesu a nakonec obdélníky s dvojitými svislými stěnami jsou podprocesy, popsané níže v textu. Některé symboly mají šedé pozadí pouze pro větší přehlednost.



Obrázek 5.1: Proces zahrání karet během vlastního tahu zobrazený vývojovým diagramem

Celý proces vlastního tahu, by se dal velmi krátce popsat, jako posloupnost vyložení či zahrání vesměs obraných karet do začátku tahu, oslabení protihráčů, oslabení hráče, který je můj cíl pro dané kolo, případné vyložení potřebné zbraně, zaútočení na cíleného hráče a nakonec odhození přebývajících karet, které si vzhledem k aktuálnímu počtu životů nemohu v ruce nechat.

Následuje bližší popis jednotlivých podprocesů vlastního tahu.

Odhad rolí hráčů:

Odhad rolí ostatních hráčů provádí podtřída `CRoleEstimator`. Tato podtřída třídy `AwareAI` vznikla na základě estimátoru z kapitoly 4.2. Má tedy přístup k logu hry, ve kterém jsou zaznamenány všechny události, které se během hry staly. Na základě těchto událostí a podmíněné pravděpodobnosti s využitím Bayesova vzorce (viz kapitola 4.2.1) odhaduje role hráčů.

Na rozdíl od původního estimátoru s označením *2.2f.a*, který na log souborech odehraných her (viz kapitola 4.1), získaných od autora projektu, podával nejlepší výsledky (viz kapitola 4.2.2), byl estimátor použitý pro moji umělou inteligenci ještě rozšířen:

1. Pokud opravdu neví jaká je role hráče, netipuje, ale označí ho za neznámého (`ROLE_UNKNOWN`).
2. Na konci estimace porovná počet odhadnutých rolí se skutečným počtem daných rolí ve hře, a pokud mu v odhadu chybí jen jedna role, tak všechny neznámé odhaduje právě touto rolí. Velmi výhodné v situaci, kdy např. jeden hráč se straní hry a neprojevuje se.
3. Poslední rozšíření, vzniklé v rámci testování, je na základě hráčovy role a funguje způsobem vylučovací metody. Analýzou hry jsem zjistil, následující fakta:

- Pokud Šerif odhadne více Pomocníků než jich je skutečně ve hře, tak zbytek hráčů jsou právě Odpadlíci a Bandité. Podle toho jaké role mu v odhadu chybí, tak těmi odhaduje neznámé hráče.
- Dále pokud Pomocník odhadne více Banditů, než je ve hře, tak neznámé hráče odhadne jako Pomocníky, jelikož za Banditu se dá splést spíše Odpadlík.
- Pokud Bandita odhadne více Pomocníků než je ve hře, tak neznámé hráče odhadne jako Odpadlíky, jelikož ti se nejspíše zatím straní hry.

Krom odhadování rolí ostatních hráčů má tato třída i přístup ke skutečným rolím hráčů. Tyto informace však používá výhradně k testovacím účelům a sběru statistik o tazích hráčů.

Obrázek 5.2: Karty jednotlivých rolí, převzato z [6]

Vytvoření množiny karet, které chci:

V situacích jako je např. reakce na zahrané *Hokynářství* si hráč musí vybrat jednu z nabízených karet. Pro tyto účely jsem jako součást třídy `AwareAI` implementoval metodu `void WCL_set()`, která podle aktuální situace ve hře pro daného hráče vytvoří množinu karet, seřazenou podle priority, které se hráči v daný moment hodí nejvíce.

Část množiny je pro všechny hráče tvořena stejně, na základě jednoduchých, obecných pravidel, a část je tvořena na základě role hráče. Kdy například pro Banditu se hodí karta *CatBalou*, pokud Šerif má na stole např. kartu *Mustang*, díky které na něj Bandita nedostřelí, apod.

Všechna implementovaná pravidla, na jejichž základě se množina vytváří, zde nemá smysl uvádět. Zdrojový kód je totiž poměrně dobře čitelný i komentovaný.

V situaci, kdy si hráč má vybírat z nějakých karet, či rozhodnout, zda se mu nějaká karta hodí či nikoliv, se tedy podívá právě do této množiny (uložené jako atribut `std::vector<PlayingCardType> wantedCardList` třídy `AwareAI`) a od nejvyšší priority k nejnižší hledá v nabízených kartách nějakou kartu vyskytující se v této množině. Toto porovnávání a výběr karty provádí funkce `PlayingCard* WCL_search()`. Pokud se mu žádná karta vědomě nehodí, vybere si ji náhodně.

Obrázek 5.3: Některé typicky žádané karty, převzato z [6]

Zvolení strategie:

Strategie hráče (uložená v atributu `enum Strategy currentStrategy`) vybraná pro daný moment hry je zvolená metodou `Strategy getStrategy()`.

Zvolená strategie může nabývat tří hodnot a to `strat_KeepBack` „neprozradit se“, `strat_Offense` „útok“ a `strat_Defense` „obrana“.

Konkrétní strategie je zvolena na základě aktuální situace ve hře a určitou měrou ovlivňuje to, jak bude hráč hrát, jaké si nechá karty v ruce na konci tahu, apod.

Za zmínku stojí strategie „neprozradit se“, kterou využívají Bandité. Na začátku hry, než začnou plnit cíl své role, a nikdo ještě neví že jsou Bandité, takto využívají momentu překvapení k tomu, aby při svém prozrazení udělali Šerifovi co největší škodu. Na jak velkou škodu pro prozrazení čekají, se nastavuje parametrem třídy `#define _BanditOffensePowerToAttackSheriff`. Hodnota 0 znamená, že na nic nečekají a útočí na Šerifa hned jak mohou a jakákoliv číselná hodnota větší než 0 znamená, že s útokem čekají až budou mít k dispozici právě tolik útočných karet, jaká je hodnota parametru. Jak hodnota tohoto parametru ovlivňuje chování a výsledky je uvedeno v kapitole 6.

Vybrání cíle:
Klíčovým pro všechny hráče je výběr cíle, na který budou směřovat svůj útok. O toto se stará funkce `PublicPlayerView* getTarget()`. Tato funkce vytváří podle role hráče a odhadnutých rolí ostatních hráčů seřazený seznam možných cílů, který potom promazává o cíle, na které hráč nemá dosah a nakonec vrací prvního hráče v seznamu, což tedy je cíl s největší prioritou, na který hráč má dosah.

Funkce využívá seznamů hráčů podle role, poskytnutých estimátorem, které si řadí podle porovnávací funkce `bool ComparePlayers()`.

- Šerif na začátek seznamu vkládá samozřejmě všechny Bandity a poté Odpadlíky.
  Pokud už ve hře není žádný Pomocník, přidává na konec seznamu i odhadnuté Pomocníky (hráče, kteří nejspíše dobře blafovali nebo hráli neracionálně) a neznámé hráče, jelikož v tu chvíli už všichni ostatní hráči hrají proti Šerifovi.
- Pomocníci mají svůj cíl stejný jako Šerif, volí tudíž svůj seznam cílů stejným způsobem.
- Primární cíl Banditů je Šerif. Za něj přidávají Pomocníky a Odpadlíky.
  V případě, že Šerif má ještě hodně života (parametr třídy `#define _ShLowLifes`) a nějaký Pomocník již málo, tak právě tohoto Pomocníka upřednostní před Šerifem za účelem ho zabít a zjednodušit si tak plnění svého cíle.
  V případě jistoty, že ve hře už není žádný další Bandita, přidává na konec seznamu i Bandity (blafující hráče) a neznámé.
- Volba cíle Odpadlíka je rozhodně nejsložitější. Aby Odpadlík splnil svůj cíl, musí se vždy přiklánět na slabší stranu, aby držel vyrovnané síly, ale zároveň musí zabránit zabití Šerifa.
  Pro určení priorit svých cílů tudíž nejdříve porovnává sílu jednotlivých stran, a to podle součtu počtu zbývajících životů všech zástupců obou stran. Nakonec ještě ze svých cílů úplně vynechává stranu, která je znatelně slabší např. Pomocníky v případě, že Šerif má málo života, aby mu pomohli zabít Bandity dříve, než oni zabijí Šerifa.
  V případě, že ve hře už zbývá jen Šerif, a nebo jeden slabý Bandita, na kterého nemá Odpadlík dosah, a silný Šerif, začíná Odpadlík útočit na Šerifa.

Funkce má krom předaného estimátoru ještě dva vstupní parametry. Prvním je `int distance`, kterým se dá uměle upravit dosah hráče pro potřeby zjištění cíle pro zahrání karet, jejichž dosah se liší od dostřelu s vyloženou zbraní, např. *Panika!* či *Duel*. Druhým parametrem je `unsigned int shift`, který určuje pozici v seznamu cílů, kterou funkce vrátí. Používá se např. když Bandita se nechce prozradit a primárním cílem je pro něj Šerif, tak chce cíl další.

Vybrání cílové karty pro zahrání karty *Panika!*:
Kartu *Panika!* hráči využívají převážně k získání nějaké karty ze stolu, která se jim hodí či Bandité pro oslabení Šerifa. O výběr karty v dosahu na kterou je *Panika!* hráčem zahrána se stará funkce `PlayingCard* whereToPlayPanika()`.

- Pokud hráč nemá zatím dostřel na žádný ze svých právě preferovaných cílů (viz výše Vybrání cíle), hledá na stole zbraň, se kterou by již dostřel měl.
- Pokud je primárním cílem Šerif, tak nejdříve zkouší sebrat kartu *Vězení* vyloženou před nějakého jiného Banditu, aby ho z něj zachránil. Poté se snaží oslabit Šerifa a pomoci ostatním, aby na něj dostřelili tím, že mu vezme ze stolu zbraně, koně či *Barel*. Pokud ani doteď nenašel svoji cílovou kartu, tak vezme zbraň protihráči, který bez ní na hráče nedostřelí.

- Nakonec si vybere zbraň *Volcanic*, pokud ji chce (viz níže Mohu zahrát tuto kartu?), a nebo *Barel,* pro svoji obranu.

Pokud hráč nenašel na stole žádnou kartu, kterou by si chtěl vzít či sebrat protihráči, oslabuje svůj cíl pro útok tím, že mu sebere kartu z ruky.

Vybrání cílové karty pro zahrání karty *CatBalou*:

Kartu CatBalou hráči využívají převážně k oslabení protihráče tím, že mu odhodí nějakou obranou kartu ze stolu. Výběr karty protihráče k odhození obstarává funkce `PlayingCard* whereToPlayCatBalou()`.

- Šerif a Pomocníci ničí Banditům takovou kartu, bez které by na Šerifa nedostřelili – typicky zbraň, nebo *Barel* aby snížili Banditovu obranu.
- Bandita naopak ničí Šerifovi a Pomocníkům karty, bez kterých by na hráče nedostřelili. A nebo kartu *Vězení* vyloženou před nějakého jiného Banditu, aby ho z něj zachránil. Nakonec případně Šerifův *Barel* pro snížení jeho obrany.
- Odpadlík podle aktuální situace ve hře, která strana je slabší (viz Vybrání cíle), oslabuje buď Bandity, a nebo Pomocníky podobným způsobem jak je popsáno výše u Šerifa a Banditů. Pokud zbývá s Šerifem poslední ve hře, tak oslabuje Šerifa.

Pokud hráč nenašel na stole žádnou kartu, kterou by chtěl protihráči odhodit, oslabuje svůj cíl pro útok tím, že mu odhodí kartu z ruky.

Obrázek 5.4: Karta Panika! a CatBalou, převzato z [6]

Vyložení zbraně:

Velmi častou situací je, že je potřeba vyložit kartu zbraně, ať už z důvodu zvýšení vlastního dostřelu, či namísto odhození při přebytku karet na konci tahu. V základním herním balíčku jsou celkem 4 druhy zbraní, lišící se dostřelem a jedna speciální *Volcanic*, se kterou jako jedinou lze zahrát neomezený počet karet *BANG!*. O problematiku výběru a vyložení zbraně se stará funkce `PlayingCard* getWeaponToPlay()`.

Kdykoli je funkce zavolána, vrátí podle aktuální situace ve hře kartu zbraně z ruky hráče, pokud ji má v daný okamžik vyložit, nebo vrátí 0, pokud momentálně žádnou zbraň nemá, není vhodná chvíle na vyložení, apod.

Bandité jsou ve vykládání zbraní opatrnější, nejen aby neprovokovali dostřelem na Šerifa. Pokud tedy nemají čím střílet, či mají dostřel dostatečný, nic nevykládají.

Nejdříve se vyhodnotí, zda chci (viz níže Mohu zahrát tuto kartu?) vyložit zbraň *Volcanic*, která má schopnost měnit události ve hře. Poté, pokud mám zbraň s větším dostřelem v ruce než na stole, tak se vyhodnocuje, zda-li mi tato „větší“ zbraň pomůže k dostřelu na cíl, na který nyní nedostřelím. Pokud ano, tak ji vyložím. Všichni krom Banditů pokud do této chvíle nenašli zbraň, kterou chtějí vyložit, vyloží tu s největším dostřelem, co mají v ruce.

Funkce má krom předaného estimátoru ještě jeden vstupní parametr a to `bool needToDiscardACard`. Výchozí hodnota tohoto parametru je `false`, pokud je však funkce zavolána s hodnotou tohoto parametru `true`, jedná se o speciální případ, kdy hráč má v ruce více karet než aktuální počet životů a potřebuje se nějakých karet zbavit. Je zřejmé, že i když „větší“ zbraň momentálně nepotřebuje, je lepší ji vyměnit za „menší“ na stole než ji odhodit. Proto v případě, že hodnota tohoto parametru je `true`, funkce doporučí vyložení „největší“ zbraně, nehledě na další okolnosti. Speciální případ je pouze s vyloženou zbraní *Volcanic*, který funkce vyměnit nedoporučí.

Obrázek 5.5: Některé karty zbraní, převzato z [6]

Mohu zahrát tuto kartu?:
Ve hře je několik karet u kterých není jednoznačné, zda-li je pro hráče výhodné je zahrát či nikoliv. Jedná se typicky o karty s efektem na všechny hráče, apod. O tom, zda tyto karty hráč má zahrát či nikoliv, rozhoduje funkce `bool canIPlay()`. Karta o které se má rozhodnout je předána jako parametr `PlayingCardType card`.

Funkce rozhoduje o zahrání karet *Indiáni!*, *Kulomet*, *Salón* a *Dynamit*, ale také o vyložení karty *Volcanic*. Karta *Volcanic* je funkcí doporučena zahrát, pokud to pro hráče má smysl (má alespoň dvě střely *BANG!*, bude mít i s pouhým *Volcanicem* dostřel, apod.) Karty s efektem na všechny hráče jsou doporučeny zahrát pokud zisk resp. ztráta pro hráče a jeho spoluhráče je větší resp. menší než zisk resp. ztráta jeho protihráčů. Dá se říci, že se jedná o metodu *MiniMax* (viz kapitola 3.4.1) v jednom tahu.

Odhození přebývajících karet:
Pravidla hry stanovují, že na konci tahu může mít hráč v ruce nejvýše tolik karet, kolik je jeho aktuální počet životů. Proto občas nastává situace, že hráči přebývají v ruce karty, které nemůže, nebo neví na koho, zahrát. V takovém případě se volá funkce `PlayingCard* whatToDiscard()`, která na základě aktuální situace ve hře vybere kartu, kterou hráč potřebuje nejméně.

Speciální případ nastává v případě postavy *Sid Ketchum*, který za dvě odhozené karty si může vrátit jeden život. Pokud mu tedy život chybí, odhodí dvě karty, i když třeba musí jen jednu, a doplní si tak chybějící život.

Funkce vybírá nejdříve karty, které hráč nepotřebuje, či nemůže zahrát. Poté nadbytečné karty *Vedle!* a *BANG!*, kterých si nechává jen určitý počet pro svoji obranu podle zvolené strategie (viz výše Zvolení strategie). Nakonec jsou považovány za přebytečné karty zbraní s menším dostřelem, než má již vyložená zbraň na stole.

Pokud není karta k odhození vybrána na základě pravidel v této funkci, je vybrána ta karta, kterou má hráč v ruce vícekrát. Pokud nemá ani žádnou kartu vícekrát (hodí se mu tedy všechny stejně), odhazuje kartu vybranou náhodně.

Další Funkce:

Pro potřeby vyhledávání karet v ruce či vyložených na stole a zjišťování počtu daných karet byly implementovány funkce `static PlayingCard* isThere()` a `static int howManyThere()`.

Dále pro potřeby testování a sběru statistik, z důvodu zhodnocení výsledků, byla implementována metoda a funkce `void SuccesEvaluate()`, `void PrintAverageSucces()`.

Speciální schopnosti postav:

Každý hráč, krom své tajné role, hraje ještě za určitou postavu. Každá postava má svoji speciální schopnost, která je veřejná, a ovlivňuje určité pravidlo hry (viz Příloha 2.).

Schopnosti některých postav jsou automatické a hráč je nijak nemůže ovlivnit (např. postava *Black Jack*, atd.). Tyto automatické schopnosti řeší hra sama.

Některé postavy ale mají schopnost, kterou je potřeba vyvolat ve vhodný okamžik hráčem, či s ní počítat při rozhodování o zahrání karet (např. postava *Sid Ketchum* a *Willy the Kid*, atd.). Schopnosti těchto postav byly implementovány tak říkajíc „ad-hoc“, tzn. že zdrojový kód byl na konkrétních místech upraven právě pro potřeby dané postavy.

Obrázek 5.6: Karty některých postav, převzato z [6]

6 Zhodnocení výsledků

K zhodnocení výsledků je vždy potřeba najít a správně použít nějakou vhodnou metriku. V případě této práce, se jedná o zhodnocení toho, jak dobře hrají hráči ovládaní umělou inteligencí, kterou jsem vytvořil. Ve hře Bang! je mnoho věcí ovlivněno faktorem náhody, může nastat nespočet různých herních situací a především neexistuje žádná obecně nejlepší strategie. Každý hráč může danou situaci vyhodnotit jinak a hrát v daný moment odlišně a není možné říci, která reakce byla lepší.

Chování vytvořené umělé inteligence není možné porovnávat s nějakým obecně nejlepším chováním hráče hry Bang!. Hráče tedy budu hodnotit na základě následujících, mnou vytvořených metrik, které popisují jednotlivé podkapitoly.

Třída umělé inteligence `AwareAI` má (jak již bylo v kapitole 5.2 v části Zvolení strategie zmíněno) parametr `_BanditOffensePowerToAttackSheriff`, ovlivňující chování Banditů resp. jejich zdrženlivost před útokem na Šerifa. V následujících podkapitolách je diskutován i vliv hodnoty tohoto parametru na výsledky jednotlivých metrik. Hodnoty parametru jsou označovány `KB0`, `KB2` a `KB3` (KB od pojmenování strategie `strat_Keep_Back`, kterou tento parametr ovlivňuje), kde číselná hodnota znázorňuje právě počet použitelných útočných karet, na které Bandita čeká k začátku útoku na Šerifa.

6.1 Úspěšnost jednotlivých rolí dle oficiálního turnajového hodnocení

Snad jedinou oficiální metrikou ve hře Bang! je celosvětový oficiální turnajový systém hodnocení[2]. Jedná se o systém hodnocení hráčů, kteří spolu odehrají určitý počet her. Tento systém by měl být spravedlivý ke všem rolím v kombinaci se všemi počty hráčů, se kterými se turnaje hrají, což je 5, 6 a 7 hráčů. Hodnocení spravedlivosti tohoto systému není úkolem této práce. Já tento systém pouze využiji k porovnání úspěšnosti mé umělé inteligence hrající za všechny role při hrách o různých počtech hráčů.

Popis oficiálního turnajového hodnocení:
Převzato z oficiálních stránek hry[2]. Každý hráč na konci každé hry, podle toho kdo vyhrál, za jakou roli hráč hrál, a zda-li přežil či nepřežil až do konce, obdrží určitý počet bodů. Bodové hodnocení pro jednotlivé role ve všech výherních situacích popisuje následující tabulka:

role hráče / *odměna ve hře:*	*5-ti hráčů*	*6-ti hráčů*	*7-mi hráčů*
V případě, že hru vyhrál Šerif:			
Šerif	3000	4500	4500
Pomocník co přežil	2200	3300	3300
Pomocník co nepřežil	1400	2100	2100

2 Oficiální turnajový systém hodnocení hráčů ve hře Bang! [online]. [cit. 22.5.2010]. Dostupné z URL: <http://www.davincigames.net/bang/torneo/BANG! Campionato nazionale - Regolamento da torneo.pdf>

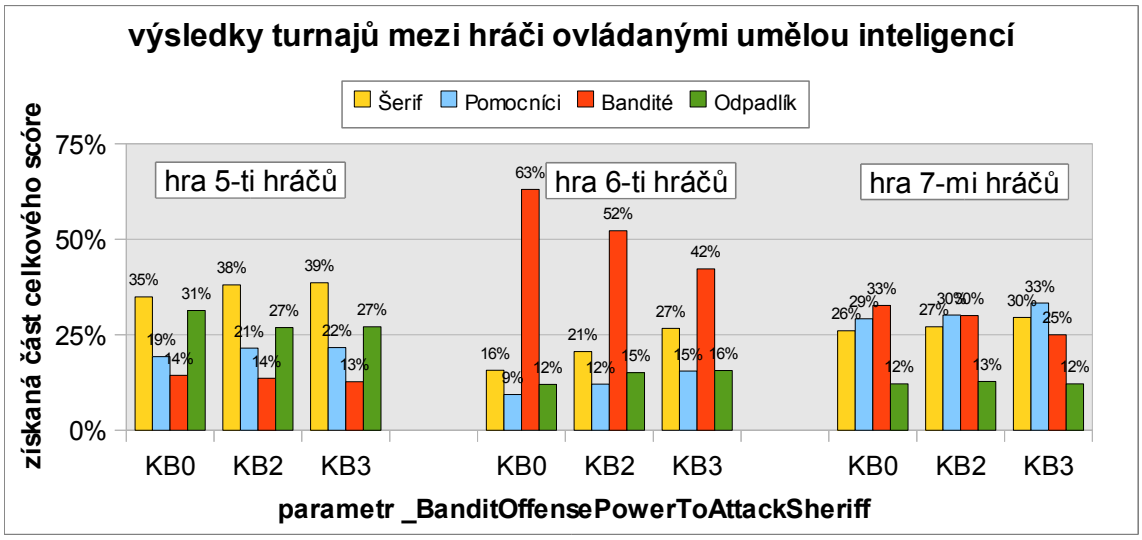
role hráče	*odměna ve hře:*	*5-ti hráčů*	*6-ti hráčů*	*7-mi hráčů*
V případě, že hru vyhrál Šerif:				
Odpadlík, který byl v závěrečném duelu s Šerifem		2000	2400	2800
Odpadlík, který nebyl v závěrečném duelu s Šerifem		0	0	0
Bandita		0	0	0
V případě, že hru vyhráli Bandité:				
Bandita co přežil		2000	3000	3000
Bandita co nepřežil		1400	2100	2100
Odpadlík co přežil		500	600	700
Odpadlík co nepřežil		0	0	0
Šerif		0	0	0
Pomocník		0	0	0
V případě, že hru vyhrál Odpadlík:				
Odpadlík		6500	7800	9100
Šerif		500	600	700
Pomocník		0	0	0
Bandita		0	0	0

Tabulka 1: Oficiální turnajový systém hodnocení hry Bang!

6.1.1 Získaná data

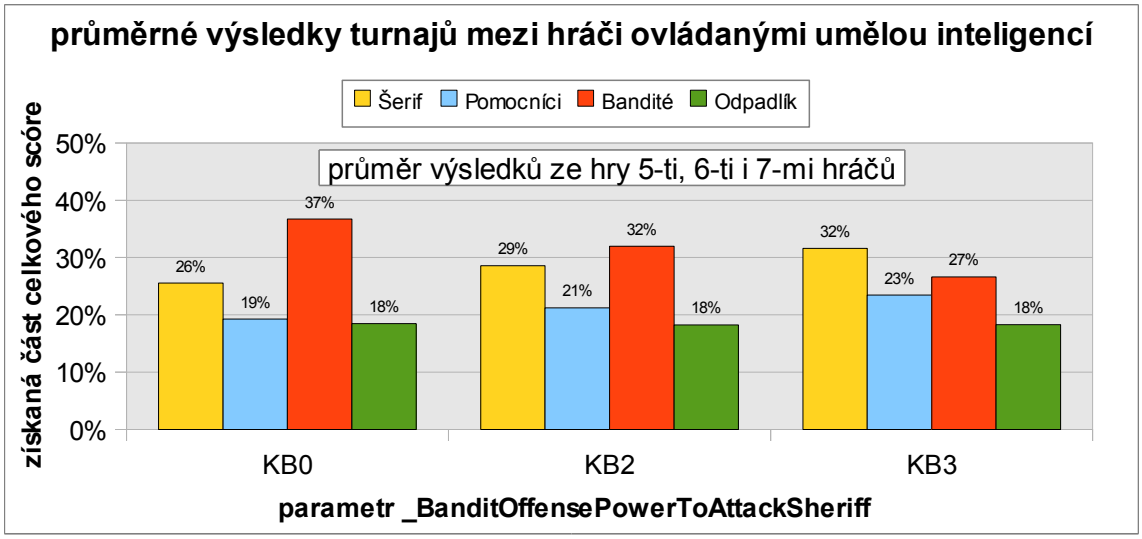
Na *BangServeru* (viz kapitola 5.1) jsem simuloval turnaj ve hře 5-ti, 6-ti i 7-mi hráčů a to vždy o 500-ti hrách, ve kterých hráli pouze hráči ovládaní umělou inteligencí. Na konci každé jednotlivé hry jsem, dle oficiálního turnajového hodnocení (viz výše), všem rolím přičítal získané body.

Výsledky jednotlivých turnajů jsem vynesl do následujícího grafu.



Obrázek 6.1: Graf úspěšnosti jednotlivých rolí ve hrách o 5-ti, 6-ti a 7-mi hráčích

Více vypovídající o úspěšnosti jednotlivých rolí ve vztahu k hodnotě parametru `_BanditOffensePowerToAttackSheriff` (`KB0`, `KB2`, `KB3`) na hru, jsou jistě průměrné výsledky pro všechny hry o různých počtech hráčů. Tyto obecné výsledky vlivu parametru na hru zobrazuje následující graf.



Obrázek 6.2: Graf úspěšnosti jednotlivých rolí, průměrné výsledky

6.1.2 Diskuse výsledků

Vyrovnanost úspěšnosti ve hrách o různých počtech hráčů:
Pro 4 role je ideálně vyrovnaná úspěšnost 25%. Celková vyrovnanost úspěšnosti se tedy dá vypočítat jako součet odchylek pro všechny role od této ideální hodnoty. Tyto odchylky zobrazuje následující tabulka.

hra:	5-ti hráčů	6-ti hráčů	7-mi hráčů	průměrná odchylka
odchylka při `KB0`	32,6%	76,0%	25,8%	44,8%
odchylka při `KB2`	29,8%	54,5%	24,5%	36,3%
odchylka při `KB3`	34,1%	37,8%	25,6%	31,6%
průměrná odchylka	31,3%	56,1%	25,3%	

Tabulka 2: Odchylka vyrovnanosti úspěšnosti rolí od ideální úspěšnosti 25%

Nejvyrovnanější výsledky podává umělá inteligence ve hře 7-mi hráčů, kde na každou roli připadá průměrná odchylka pouhých 6,3%.

Ve hře **5-ti hráčů** je nejúspěšnější Šerif a Odpadlík. Je to logické, důvod je ten, že právě ve hře s nejmenším počtem hráčů je pro Odpadlíka nejjednodušší vyhrát, proto ve hře vetšinou jako poslední zůstávají právě tito dva hráči a o výhry se dělí.

Hra **6-ti hráčů** je naopak nejvýhodnější pro Bandity, kterých je, oproti hře 5-ti hráčů, o jednoho více. Mají tak mnohem větší šanci Šerifa usmrtit. Úspěšnost ostatních rolí je poměrně vyrovnaná, ale několikanásobně nižší než úspěšnost Banditů.

Úspěšnost všech jednotlivých rolí je nejvyrovnanější ve hře **7-mi hráčů**. Pouze pro Odpadlíka je toto nejsložitější situace, jelikož se musí postarat o vyřazení ze hry nejvíce hráčů.

Vliv parametru `_BanditOffensePowerToAttackSheriff` na vyrovnanost výsledků:
Nejvyrovnanější výsledky podává umělá inteligence s hodnotou parametru `KB3` (viz Tabulka 2, pravý sloupec).

S hodnotou parametru **`KB0`** jsou nejúspěšnější Bandité. Z toho plyne, že Banditům se nevyplácí otálet s útokem na Šerifa, ale naopak statisticky nejvýhodnější pro ně je, zaútočit ihned, jak mají možnost.

Při hodnotě parametru **`KB2`** jsou stále nejúspěšnější Bandité, ale už ne s takovým náskokem od Šerifa s Pomocníky.

Při hodnotě parametru **`KB3`** je nejúspěšnější Šerif, ale s pouhým 5-ti procentním náskokem před Bandity.

Na výsledky Odpadlíka parametr nemá vliv.

6.2 Odhad rolí estimátorem

Jako další metriku ke zhodnocení mé umělé inteligence jsem zvolil úspěšnost estimátoru rolí, který je velmi důležitou, možná tou nejdůležitější, součástí hráčovy umělé inteligence.

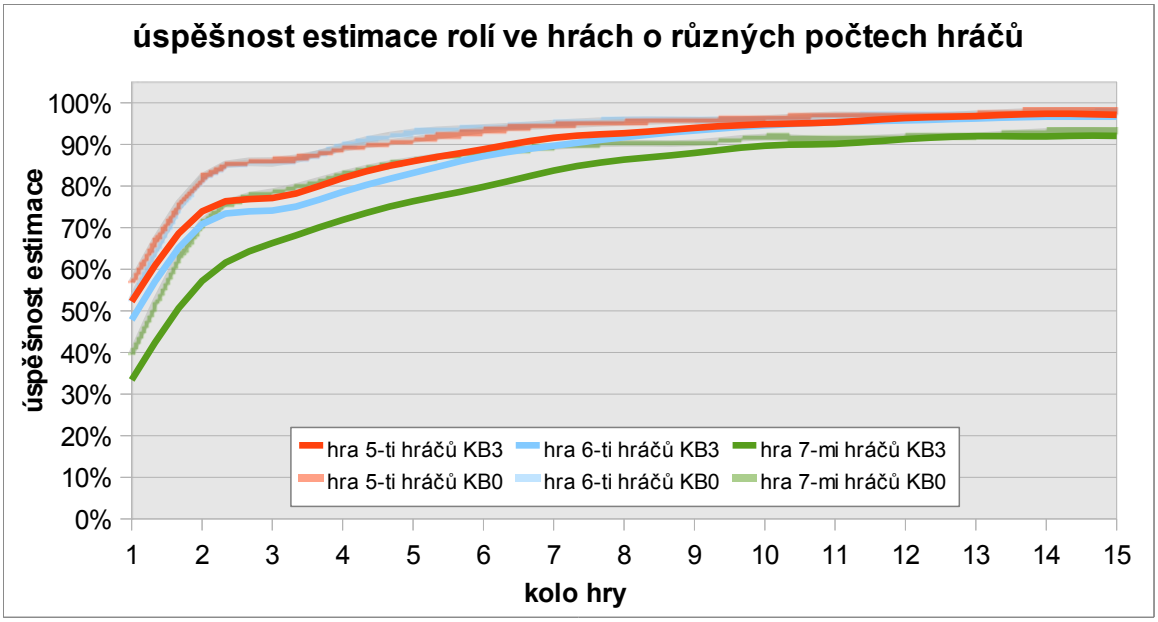
Vývoj tohoto estimátoru začal na log souborech zaznamenaných her, které jsem získal od autora projektu KBang (viz kapitola 4.1). Na základě těchto dat, jsem zjistil, která metoda odhadu má nejlepší výsledky. Ve fázi implementace samotné umělé inteligence jsem ještě estimátor rozšířil o další způsoby odhadu rolí hráčů (viz kapitola 5.2.1 část Odhad rolí hráčů).

6.2.1 Získaná data

Na *BangServeru* (viz kapitola 5.1) jsem spustil 500 her postupně s 5-ti, 6-ti i 7-mi hráči, ve kterých hráli pouze hráči ovládaní umělou inteligencí. Během všech tahů každého hráče jsem zaznamenával kolik hráčů odhadl estimátor správně a kolik špatně. Z těchto údajů jsem vypočítal úspěšnost estimátoru pro všechny hráče ve všech tazích každé hry. Výsledky pro jednotlivé tahy jsem v rámci jednotlivých hráčů pro všech 500 odehraných her zprůměroval.

Úspěšnost estimátoru pro jednotlivé role ve hře 5-ti, 6-ti i 7-mi hráčů jsem vynesl do grafů v Příloze 3..

Následující graf zobrazuje srovnání úspěšnosti estimátoru ve hrách o různém počtu hráčů. Celková úspěšnost pro hru o daném počtu hráčů byla získána zprůměrováním úspěšnosti všech rolí.



Obrázek 6.3: Graf úspěšnosti estimátoru ve hrách o různých počtech hráčů

6.2.2 Diskuse výsledků

Úspěšnost estimátoru ve hrách o různých počtech hráčů:
Úspěšnost estimátoru je, s přibývajícími odehranými koly hry, rostoucí, což je správně, jelikož hráči odehrají více tahů, které vypovídají o jejich roli. Dle mého názoru je úspěšnost obecně velmi dobrá. Po prvním odehraném kole je pro všechny situace úspěšnost více jak 50% a během pár dalších kol více jak 80%. Průměrná hra trvá 15 až 30 kol a estimátor, ve všech případech, si je zhruba od 10. kola už téměř 100% jist všemi rolemi.

Ve hře 5-ti hráčů podává estimátor samozřejmě nejlepší výsledky. Ve hře 6-ti hráčů je jeho úspěšnost po prvním kole o 4% nižší a v dalších kolech už je rozdíl menší jak 2%. Ve hře 7-mi hráčů je situace pro estimátor nejsložitější. Po prvním a druhém kole, ve hře 7-mi hráčů, je úspěšnost estimátoru nižší o 13% než ve hře 6-ti hráčů. V dalších kolem se rozdíl postupně snižuje, ale odhad je stále o více jak 5% horší než ve hře 6-ti hráčů.

Vliv parametru `_BanditOffensePowerToAttackSheriff` na úspěšnost estimátoru:
Úspěšnost estimátoru jsem zkoumal pouze pro hodnotu parametru `KB0` a `KB3`, jelikož právě mezi výsledky s těmito hodnotami je největší rozdíl.

Strategie Banditů `KB3` – „držet se zpátky“ má samozřejmě za následek horší čitelnost hráčů, která se projeví především ze začátku hry. Proto s hodnotou parametru `KB3` má estimátor ve všech případech zhruba o 10% horší výsledky ze začátku hry, než bez této strategie – s hodnotou parametru `KB0`. Od 7. kola hry je rozdíl 5% a od 12. kola hry je rozdíl menší než 1%, vždy samozřejmě v neprospěch parametru `KB3`.

6.3 Chybovost rolí

Snad poslední metrikou, která se dá na hráče hry Bang! použít je správnost jejich tahů. Jak jsem již několikrát zmínil, správnost tahů se dá velmi těžko posuzovat, jelikož každý hráč může danou situaci vyhodnotit jinak. Zaměřím se tedy pouze na jednoznačnou část této problematiky, kterou je cílový hráč, na kterého hráč hraje své útočné karty, jelikož právě toto se dá objektivně posoudit.

Výsledky této metriky jsou závislé na estimátoru rolí resp. na celkovém počtu hráčů ve hře, který udává počet jednotlivých rolí. Je zřejmé, že ve hře 5-ti hráčů, Pomocník a Odpadlík těžko může cílit své útočné karty špatně, jelikož všichni ostatní hráči (nepočítaje Šerifa, jehož role je však veřejná) jsou jejich protihráči. Pro vyhodnocení této metriky se tedy zaměřím pouze na nejsložitější situaci pro estimátor rolí, kterou je hra 7-mi hráčů.

Stejně jako v kapitole 6.2 jsem tuto metriku zkoumal pouze pro hodnotu parametru `_BanditOffensePowerToAttackSheriff` `KB0` a `KB3`, jelikož právě mezi výsledky s těmito hodnotami je největší rozdíl.

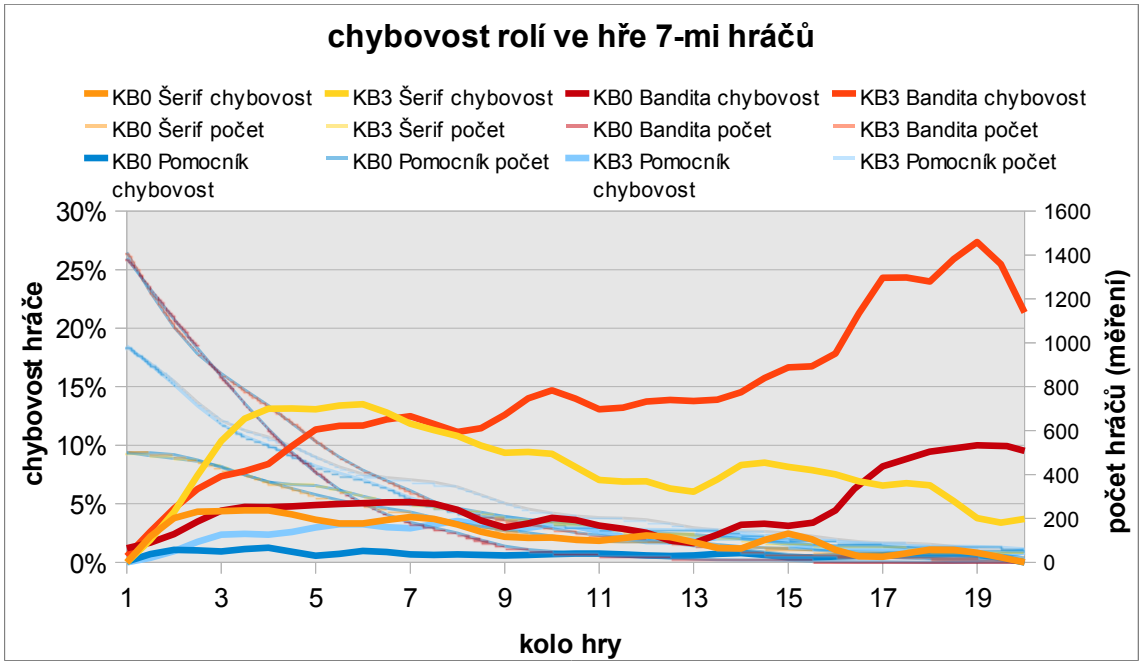
6.3.1 Získaná data

Na *BangServeru* (viz kapitola 5.1) jsem spustil 500 her postupně s 5-ti, 6-ti i 7-mi hráči, ve kterých hráli pouze hráči ovládaní umělou inteligencí. Během celého tahu všech hráčů jsem zaznamenával na koho hráči útočí a porovnával role cílů s rolí hráče. Na konci každého tahu jsem vypočítal a zaznamenal podíl karet mířených na spoluhráče.

Hráč hrající za Odpadlíka je vždy protihráčem všem ostatním hráčům, žádnou kartu tedy nemůže zahrát na spoluhráče. Z tohoto důvodu není v grafu role Odpadlíka uvedena.

Chybovost (podíl zahraných útočných karet na své spoluhráče oproti protihráčům) jednotlivých rolí ve hře 5-ti a 6-ti hráčů je zobrazena v grafech v Příloze 4..

Chybovost jednotlivých rolí ve hře 7-mi hráčů zobrazuje následující graf.



Obrázek 6.4: Graf chybovosti(podíl útoků na spoluhráče oproti protihráčům) rolí ve hře 7-mi hráčů

6.3.2 Diskuse výsledků

Chybovost rolí:
Ve hře 5-ti a 6-ti hráčů (viz Příloha 4.) je chybovost Šerifa zanedbatelná a dá se říci, že téměř nikdy na svého Pomocníka neútočí. Ve hře 7-mi hráčů, v případě kdy Bandité matou ostatní (hodnota parametru `KB3`), se chybovost Šerifa pohybuje kolem 10%, což považuji za poměrně dobrý výsledek.

O chybovosti Pomocníků ve hře 5-ti a 6-ti hráčů nemůže být řeč, jelikož všichni ostatní hráči (krom Šerifa) jsou jeho spoluhráči. Ve hře 7-mi hráčů je chybovost Pomocníka pod hodnotou 4% i v případě, kdy Bandité matou ostatní, což je naprosto zanedbatelné.

Chybovost Banditů, když matou ostatní, je vždy zhruba dvakrát vyšší než když útočí na Šerifa, hned jak mají možnost (hodnota parametru `KB0`). Ale i v nejtěžší situaci je pod hranicí 15%, což je velmi dobrý výsledek.

Na grafu na obrázku 6.4 je od 15. kola hry vidět znatelný nárůst chybovosti Banditů až k hodnotě téměř 30%. Jedná se o statistickou chybu, jelikož v této pokročilé fázi her jsou údaje o chybovosti Banditů založeny pouze na pár měřeních (viz křivky počtu měření na vedlejší ose Y).

Vliv parametru `_BanditOffensePowerToAttackSheriff` na chybovost rolí:
Hodnota parametru `KB3`, ovlivní hru tím způsobem, že Bandité se neprozradí hned jak na Šerifa mají dosah, ale až později, až mu mohou způsobit větší škodu. Toto má za následek, že ze začátku hry není jasné, kdo hraje za jakou roli. Působí to samozřejmě na všechny hráče, včetně Banditů samotných i Šerifa.

Při hodnotě parametru `KB0` je chybovost všech rolí poměrně nízká, a až na výjimky u Banditů se drží pod 5%. Při hodnotě parametru `KB3` je chybovost Banditů vyšší o zhruba 7%, Šerifova chybovost je ze začátku hry vyšší o 10% a poté (když se Bandité prozradí) klesá, a je vyšší o 6%. Chybovost Pomocníků parametr příliš neovlivní, s hodnotou `KB3` je jejich chybovost vyší o 1 až 3%, ale stále pod hodnotou 5%.

6.4 Hra s reálnými hráči

Jako další způsob otestování umělé inteligence musí vždy bezesporu být její srovnání s lidmi. Tím myslím srovnání výsledků umělé inteligence a lidí při řešení stejného problému. V tomto případě jde o problematiku hraní hry Bang! reálným hráčem ve srovnání s hráčem ovládaným umělou inteligencí.

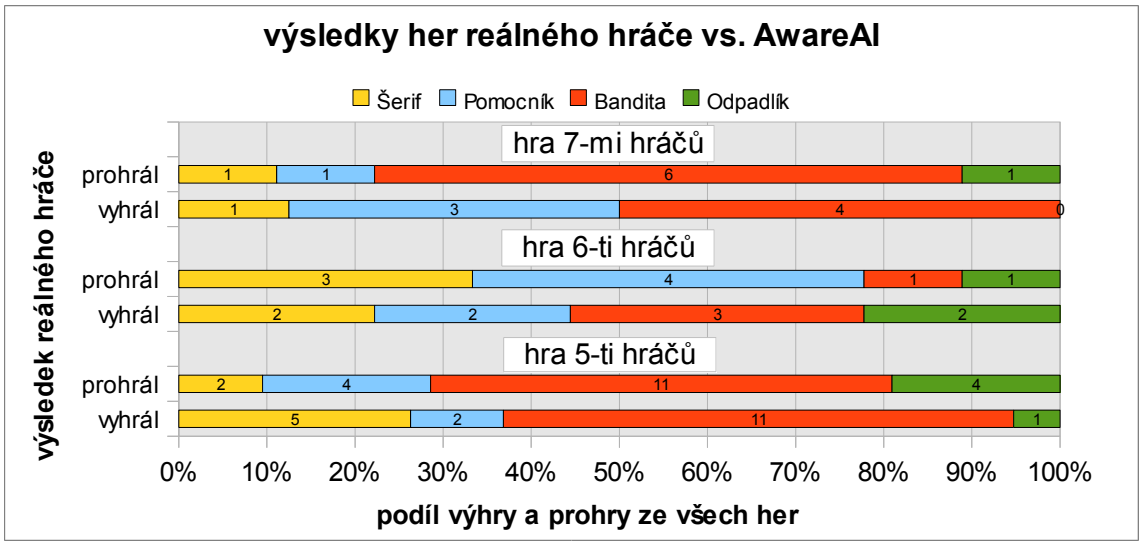
Problém však nyní nastává u otázky: „Co považovat za úspěch?“. Týmy lidí vyvíjející např. umělou inteligenci pro hru Šachy, mají v této otázce jasno. Jejich cílem je vytvořit umělou inteligenci, která v co nejvíce případech nejlepšího lidského hráče porazí. Úkolem této práce však není vytvořit neporazitelnou umělou inteligenci. Mým cílem bylo a je, vytvořit umělou inteligenci takovou, která bude přiměřeným protihráčem, kterého nebude příliš snadné, ale ani příliš těžké, porazit.

Testování umělé inteligence, ve hře s reálnými hráči, jsem se jako její autor nemohl zúčastnit, jelikož její způsob uvažování znám do nejmenších detailů a mé výsledky by nebyly objektivní. Do procesu jsem tedy zapojil několik dobrých hráčů hry Bang! z okruhu lidí, co znám. Zprvu jsem je využíval k testování a optimalizování algoritmů. Konec této fáze jsem poznal tak, že výpis nedostatků a chyb, které mi posílali, se velmi zkrátil a jejich odezva byla vesměs pouze pozitivní.

6.4.1 Získaná data

Po skončení fáze testování a optimalizace jsem projekt přestal vyvíjet a nechal testery odehrát řadu her, pro potřeby srovnání umělé inteligence ve hře s reálnými hráči.

Následující graf zobrazuje poměr vyhraných a prohraných her reálných hráčů za různé role, za které hráli, většinou jako jediní reální hráči ve hře.



Obrázek 6.5: Graf srovnání reálných hráčů a umělé inteligence v poměru výher a proher

6.4.2 Diskuse výsledků

Graf zobrazuje výsledky pouze několika desítek her, nejedná se o žádný statisticky silný vzorek. Počet her reálných hráčů dané kategorie udává číslo vně buněk. Získat statisticky silný vzorek her s reálnými hráči není totiž vůbec jednoduché. Pro potřeby této práce, ale i takový vzorek v této části stačí. Z grafu je vidět, že za všechny role mohou reální hráči umělou inteligenci porazit, ale stejně tak i sami být poraženi. Toto bylo jedním z nejdůležitějších cílů této práce.

7 Závěr

V rámci této práce jsem si vyzkoušel pokračovat v cizím projektu a poznal jsem všechny výhody, ale především i nevýhody, takové práce. Dále jsem si vyzkoušel, co vše obnáší vytvoření umělé inteligence pro typicky lidskou činnost jako je např. hraní her.

Cílem této práce bylo vytvořit umělou inteligenci do karetní hry Bang!, která by byla rovným soupeřem dobrým hráčům, ale na druhou stranu by nemělo být nemožné ji porazit, aby zůstala zachována podstata této hry, kterou je zábava pro lidi.

Testování umělé inteligence, nejen ve hrách s reálnými hráči, potvrdilo (viz kapitola 6), že cíle práce byly splněny. Reální hráči umělou inteligenci dokáží porazit a stejně tak i umělá inteligence dokáže porazit reálné hráče. Úspěšnost estimátoru rolí je vynikající. Během pár kol hry, ve všech situacích, má estimátor téměř jistotu v roli všech hráčů. Výjimky, které kazí statistiku, jsou nejasnosti mezi Pomocníky a Odpadlíky, které z pohledu ostatních hráčů je až do poslední chvíle téměř nemožné rozlišit a není to ani nezbytné. Ve hrách o různých počtech hráčů podává umělá inteligence výsledky odpovídající realitě. To svědčí o správné implementaci, vyrovnané pro všechny role. Chybovost (podíl zahraných útočných karet na své spoluhráče oproti protihráčům) hráčů hrajících za všechny role je na velmi dobré úrovni. I v těch nejsložitějších situacích je maximální hodnota chybovosti 15% a to pouze u Banditů. Chybovost Šerifa a Pomocníků je ve všech případech téměř zanedbatelná. Umělá inteligence hraje tedy téměř bez chyb. Z vlastních zkušeností z reálných her s pouze lidskými hráči vím, že hodnoty chybovosti mé umělé inteligence jsou ve srovnání s reálnými hráči vynikající. I ti nejlepší hráči totiž udělají téměř ve všech hrách nějakou chybu. Tato skutečnost ale k této hře patří a považuji ji spíše za zábavnou než negativní. Nejvýhodnější nastavení parametru `_BanditOffensePowerToAttackSheriff` je `KB3`, kdy umělá inteligence podává nejvyrovnanější výsledky pro všechny role a zároveň dělá hru nejzajímavější, jelikož vše není jasné hned od začátku. Úspěšnost estimátoru je sice s hodnotou parametru `KB3`, ze začátku hry, nižší o 10%, ale s postupem hry se tento rozdíl snižuje velmi rychle a to až k zanedbatelné hodnotě menší než 1%.

I přes poměrně velmi dobré výsledky má tato, stejně jako každá jiná, umělá inteligence mnoho možností k rozšíření. V aktuálním stavu hraje UI stejně jako nadprůměrný hráč. Existuje ale řada možností jak její kvality vylepšit. Jednou z nich je např. využití spolehlivé paměti počítače, která si bez problému může pamatovat všechny karty, které již byly zahrány, a na základě znalosti o počtu všech karet v balíčku, predikovat jaké karty budou ještě ve hře, jaká je pravděpodobnost, že protihráč má nějakou konkrétní kartu na obranu, apod. Dalším možným rozšířením by mohlo být využití náhodných čísel ve správném rozsahu pro hodnoty některých parametrů. Umělá inteligence by si vždy na začátku hry tyto parametry náhodně zvolila. Toto rozšíření by vedlo ke složitějšímu predikování tahů UI především pro hráče, kteří by proti UI hráli pravidelně a snažili se vylepšit své výsledky na základě odhalení algoritmů UI a predikce jejích tahů.

Součástí této práce jsou kompletní zdrojové kódy projektu KBang včetně nově implementované umělé inteligence AwareAI. Dále zdrojové kódy parseru pro zpracování logů zaznamenaných her a všechny log soubory (viz kapitola 4), které jsem získal od autora prvotní verze projektu. Všechny tyto soubory, včetně veškerých dat, která byla použita ke zhodnocení výsledků a k vygenerování všech grafů, jsou uloženy na přiloženém datovém nosiči.

Všechny náležitosti této práce vzhledem k zadání a v rámci diplomové práce byly splněny.

Literatura

[1] Pravidla hry Bang! [online]. [cit. 2.1.2010]. Dostupné z URL: <http://bang.cz/index.php?op=bang&lang=cs>

[2] „přispívající“: Bang! [online]. [cit. 2.1.2010]. Dostupné z URL: <http://en.wikipedia.org/wiki/Bang!>

[3] Zbořil, F., Zbořil, F. ml.: *Studijní opora předmětu IZU – Základy umělé inteligence*. Brno, VUT FIT, 2006

[4] „přispívající“: Umělá inteligence [online]. [cit. 2.1.2010]. Dostupné z URL: <http://cs.wikipedia.org/wiki/Kategorie:Umělá_inteligence>

[5] Zendulka, J., Bartík, V., Lukáš, R., Rudolfová, I.: *Studijní opora předmětu ZZN – Získávání znalostí z databází*. Brno, VUT FIT, 2009

[6] Čevora Michal: projekt KBang [online]. [cit. 2.1.2010]. Dosupné z URL: <http://code.google.com/p/kbang/>

[7] Hrubý, M.: Studijní materiály předmětu THE – Teorie her. Brno, VUT FIT, 2009

[8] Munakata, T.: *Fundamentals of the New Artificial Intelligence*, Springer-Verlag New York, Inc., 1998. ISBN 0-387-98302-3

Seznam příloh

Příloha 1.: Základní terminologie kolem karetní hry Bang!

- **Šerif, Bandita, Pomocník** a **Odpadlík** jsou **role** postav (hráčů)
- **odhazovací balíček**: místo pro odhozené, již nepoužitelné, karty
- **lízací balíček**: zbytek použitelných karet po počátečním rozdání
- **zahrát**: znamená použít kartu podle pravidel k provedení jejího efektu
- **odhodit**: je výraz pro přemístění karty na odhazovací balíček – hřbitov
- **líznout**: je otočení horní karty z lízacího balíčku a její odhození na odhazovací balíček
- **tah**: jsou všechny dle pravidel možné a nutné akce jednoho hráče
- **kolo**: jeden tah všech hráčů
- **vzít si kartu**: je proces přesunu jedné karty z lízacího balíčku do ruky hráče
- **zranit**: znamená ubrat život hráči
- **zabít**: znamená vyřadit hráče ze hry
- **oslabit**: znamená zranění hráče či sebrání mu jedné či více karet
- **karta tzv. ve hře**: je vyložená před hráčem

Příloha 2.: Postavy a jejich schopnosti

Příloha obsahuje jména postav v abecedním pořadí a popis jejich schopností v základním balíčku hry Bang!.

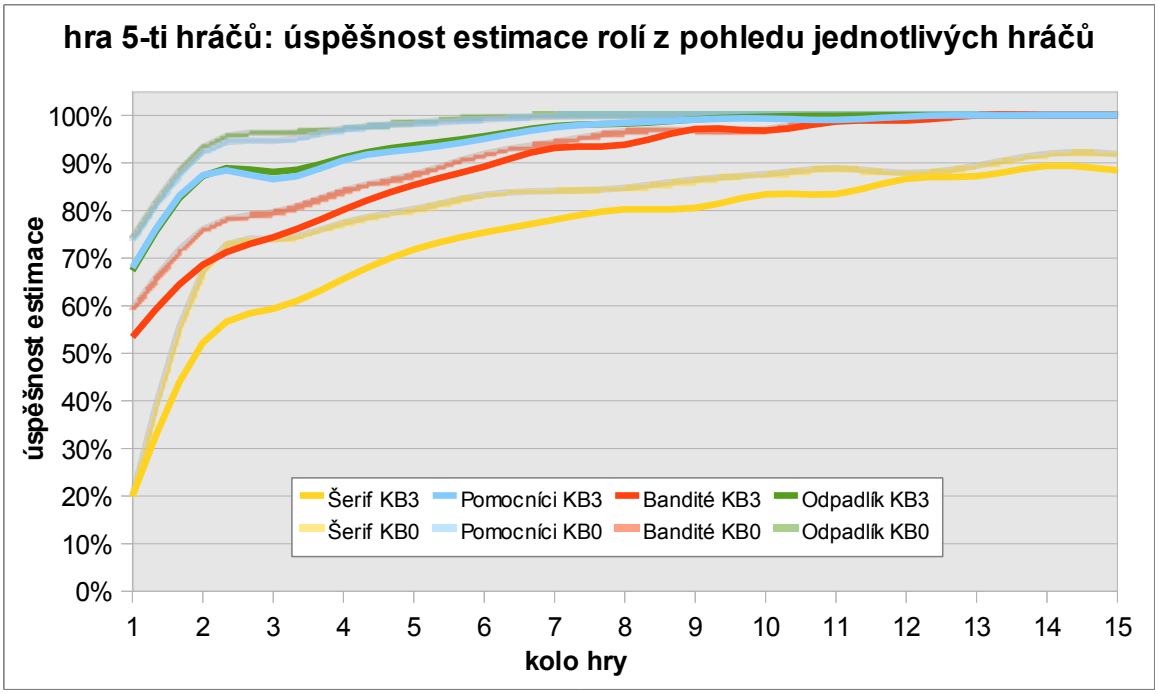
Převzato z [1]. Text pouze drobně upraven do terminologie používané v této práci - popsané v Úvodu.

- **Bart Cassidy** (4 životy): kdykoli ztratí život, vezme si kartu.
- **Black Jack** (4 životy): během fáze 1 svého tahu ukáže druhou kartu, co si vzal: je-li ta karta srdcová či kárová, vezme si ještě jednou navíc (již opět skrytě).
- **Calamity Janet** (4 životy): může používat karty *BANG!* jako karty *Vedle!* a naopak. Zahraje-li kartu *Vedle!* jako *BANG!* (jako efekt normálního výstřelu), nemůže zahrát další *BANG!* (tedy pokud nemá ve hře *Volcanic*). Této schopnosti může využívat i při *Duelu*, *Indiánech* atp.
- **El Gringo** (3 životy): vezme si náhodně kartu od hráče, který mu způsobí ztrátu života (za každý ztracený život jednu). Nic se neděje, nemá-li protivník již žádnou kartu. Pozor na to, že zranění od *Dynamitu* není způsobeno hráčem, který *Dynamit* vyložil.
- **Jesse Jones** (4 životy): během fáze 1 svého tahu si může vybrat, vezme-li si první kartu z balíčku nebo z ruky jiného hráče. Druhou kartu si vždy vezme z balíčku.
- **Jourdonnais** (4 životy): bere se, jako kdyby měl ve hře kartu *Barel*. Jakýkoli opravdový *Barel* ve hře přidává efekt k tomu fiktivnímu, čímž si jeho hráč může lízat pro zrušení *BANG!* dvě karty (za každý *Barel* jednu).
- **Kit Carlson** (3 životy): během fáze 1 svého tahu se podívá na vrchní tři karty balíčku, vybere si 2, které si vezme a třetí vrátí na vršek balíčku.
- **Lucky Duke** (4 životy): vždy, když si má líznout, otočí si vrchní 2 karty balíčku a vybere si, která se mu hodí. Obě karty jsou následně odhozeny.
- **Paul Regret** (3 životy): bere se, že má ve hře kartu *Mustang*. Jakýkoli opravdový *Mustang* se přičítá k tomu fiktivnímu.
- **Pedro Ramirez** (4 životy): během fáze 1 svého tahu, si může vzít horní kartu z odhazovacího balíčku a teprve druhou si z lízacího.
- **Rose Doolan** (4 životy): bere se, že má ve hře kartu *Appaloosa*. Jakákoli opravdová karta *Appaloosa* se sčítá s tou fiktivní.
- **Sid Ketchum** (4 životy): kdykoli může odhodit 2 karty z ruky, aby si vyléčil jeden život. Tuto schopnost může použít několikrát za sebou.
- **Slab the Kille**r (4 životy): protivníci, kteří chtějí zrušit jeho karty *BANG!* musí zahrát 2 karty *Vedle!* Schová-li se někdo úspěšně za *Barel*, bere se to jako jedna zahrána karta *Vedle!* (je tedy nutné zahrát ještě jednu kartu *Vedle!*).
- **Suzy Lafayette** (4 životy): kdykoli nemá žádnou kartu v ruce, vezme si novou z lízacího balíčku.
- **Vulture Sam** (4 životy): kdykoli je hráč vyřazen ze hry, Sam získá všechny karty daného hráče, co měl v ruce a před sebou na stole.
- **Willy The Kid** (4 životy): může zahrát libovolný počet karet *BANG!* ve svém tahu.

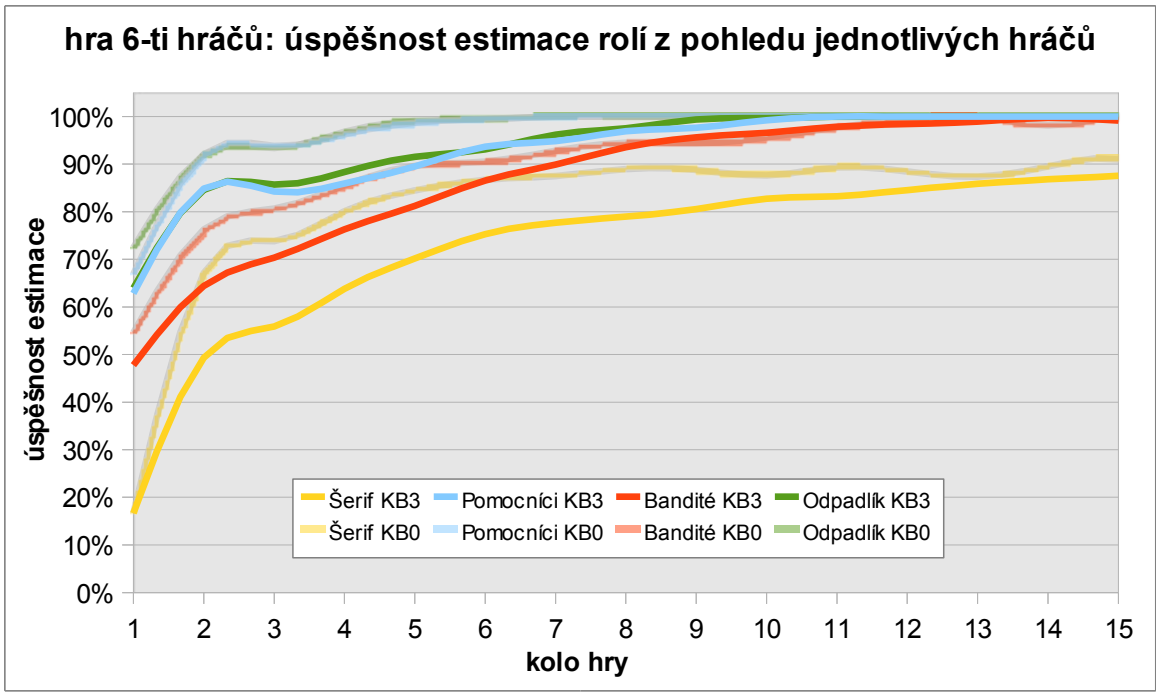
Příloha 3.: další grafy úspěšnosti estimátoru

V této příloze se nacházejí rozšiřující grafy pro kapitolu 6.2.

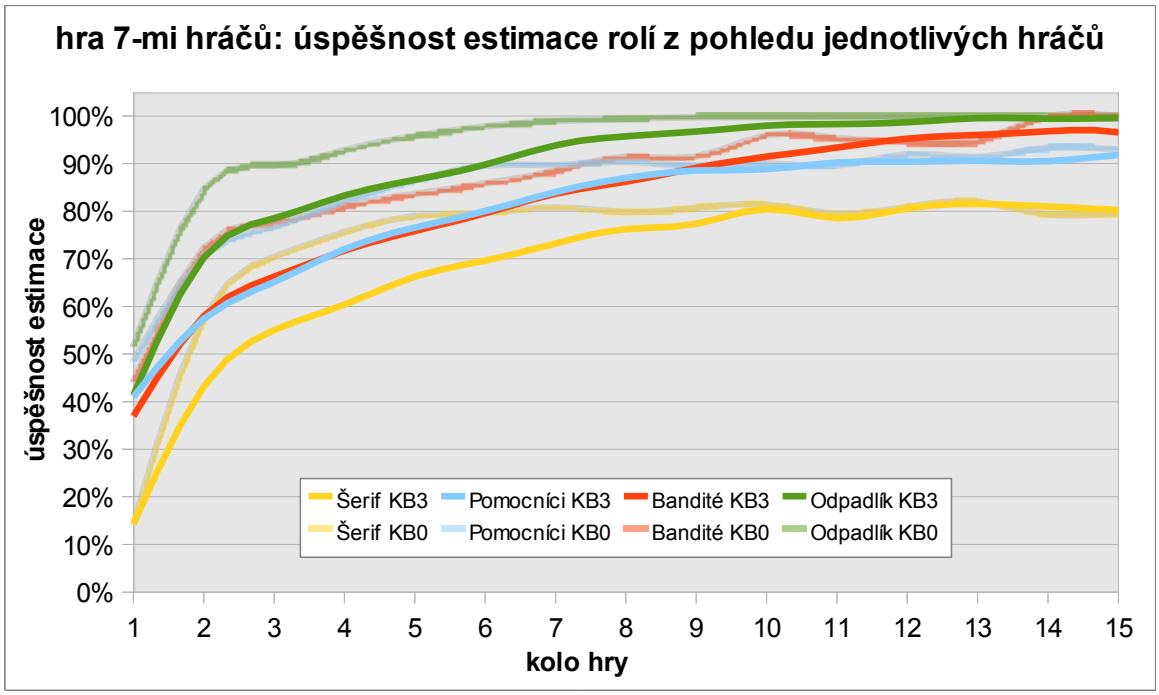
Tyto grafy zobrazují úspěšnost odhadu rolí ostatních hráčů z pohledu jednotlivých rolí ve hrách o různém počtu hráčů. Z grafů je například vidět o kolik je odhadování rolí těžší pro Šerifa než pro ostatní hráče, apod.



Obrázek 7.1: Graf úspěšnosti estimátoru z pohledu jednotlivých hráčů ve hře o 5-ti hráčích



Obrázek 7.2: Graf úspěšnosti estimátoru z pohledu jednotlivých hráčů ve hře o 6-ti hráčích



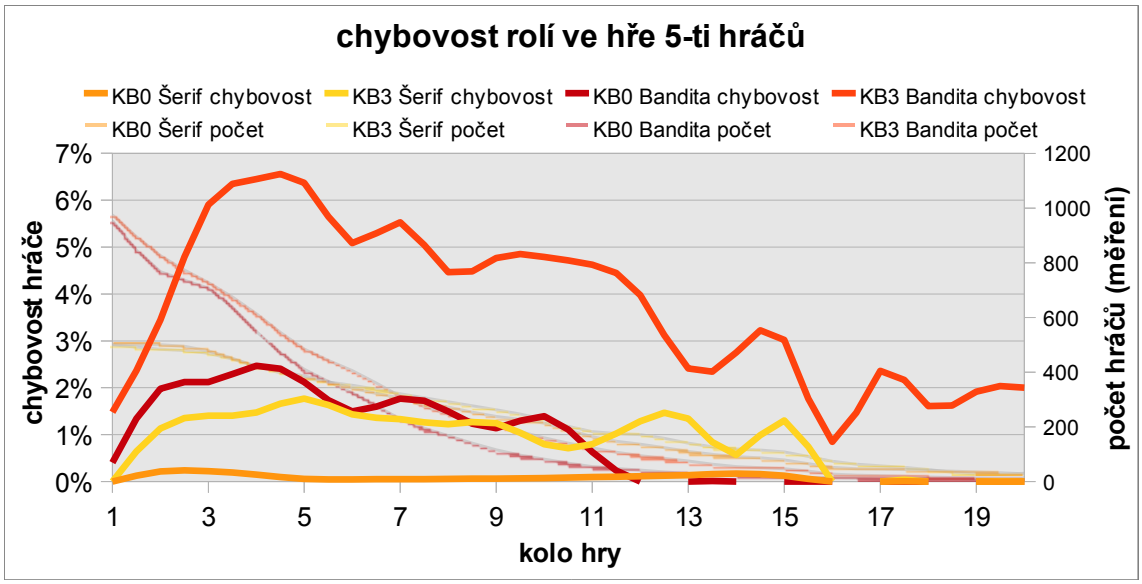
Obrázek 7.3: Graf úspěšnosti estimátoru z pohledu jednotlivých hráčů ve hře o 7-mi hráčích

Příloha 4.: další grafy chybovosti hráčů

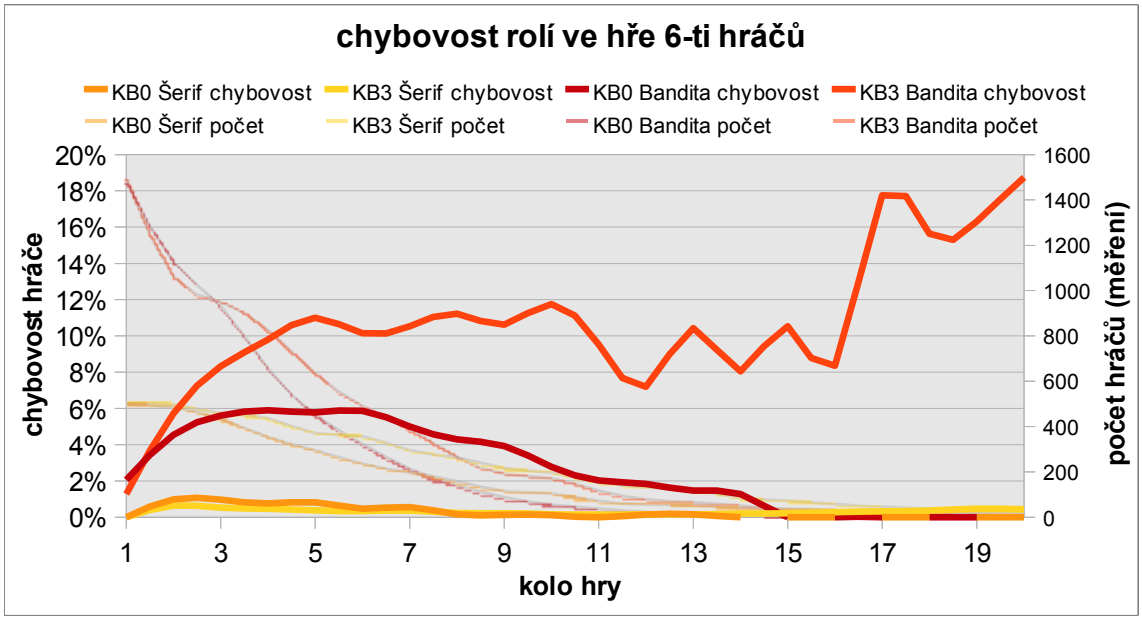
V této příloze se nacházejí rozšiřující grafy pro kapitolu 6.3.

Tyto grafy zobrazují chybovost (podíl zahraných útočných karet na své spoluhráče oproti protihráčům). Z grafů je například vidět jak malou chybovost má umělá inteligence ve hře 5-ti hráčů.

V grafech není uvedena chybovost Pomocníka ani Odpadlíka, jelikož tito hráči jsou v těchto hrách jediní zástupci svých rolí, tudíž všichni ostatní (samozřejmě krom Šerifa) jsou jejich protihráči.



Obrázek 7.4: Graf chybovosti(podíl útoků na spoluhráče oproti protihráčům) rolí ve hře 5-ti hráčů



Obrázek 7.5: Graf chybovosti(podíl útoků na spoluhráče oproti protihráčům) rolí ve hře 6-ti hráčů