Katedra Informatiky
Fakulta Matematiky, Fyziky a Informatiky
Univerzita Komenského, Bratislava

Implementácia FOSS verzie hry Mariáš s dôrazom na algoritmus pre počítačového protihráča

(Diplomová práca)

Michal Kováč

Vedúci: RNDr. Michal Forišek, PhD.

Bratislava, 2011

Katedra Informatiky
Fakulta Matematiky, Fyziky a Informatiky
Univerzita Komenského, Bratislava

Implementácia FOSS verzie hry Mariáš s dôrazom na algoritmus pre počítačového protihráča

(Diplomová práca)

Michal Kováč

kód: cbca28a2-49ec-473a-b25c-8a3882c766b3

Vedúci: RNDr. Michal Forišek, PhD.

Bratislava, 2011

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Michal Kováč
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský

Názov : Návrh a implementácia FOSS verzie hry Mariáš

Cieľ : Cieľom práce je vytvoriť kompletný softvérový produkt: free a open source implementáciu hry Mariáš. Hlavný dôraz pri vypracovaní práce však má byť kladený nie na samotnú implementáciu herných mechanizmov, ale na návrh a implementáciu algoritmu pre počítačových protihráčov.

Vedúci : RNDr. Michal Forišek, PhD.

Dátum zadania: 08.12.2009

Dátum schválenia: 18.02.2011

prof. RNDr. Branislav Rovan, PhD.
garant študijného programu

Kováč
študent

vedúci práce

Dátum potvrdenia finálnej verzie práce, súhlas s jej odovzdaním (vrátane spôsobu sprístupnenia)

3.5.2011
vedúci práce

Čestne prehlasujem, že som túto diplomovú prácu vypracoval samostatne s použitím citovaných zdrojov.

. . . . . .

Poďakovanie

Osobitná vďaka patrí vedúcemu diplomovej práce RNDr. Michalovi Foríškovi PhD. za cenné rady, námety, podnetné pripomienky a všestrannú pomoc, ktorú si hlboko vážim. Len vďaka mnohým prínosným konzultáciám a intenzívnej spolupráci som bol schopný napísať toto dielo. Nesmiem zabudnúť ani na prof. RNDr. Pavla Ďuriša PhD., prof. RNDr. Branislava Rovana PhD. a spolužiakov za to, že si na diplomovom seminári našli čas, aby si vypočuli moju prezentáciu diplomovej práce. Ďalšie poďakovania venujem rodine, známym a kolegom vo firme Bonet Development, ktorí to so mnou dokázali vydržať posledné týždne pred odovzdaním.

Abstrakt

Autor: Michal Kováč

Názov práce: Implementácia FOSS verzie hry Mariáš s dôrazom na algoritmus pre počítačového protihráča

Typ práce: Diplomová práca

Škola: Univerzita Komenského v Bratislave

Fakulta: Fakulta matematiky, fyziky a informatiky

Katedra: Katedra informatiky

Školiteľ: RNDr. Michal Forišek, PhD.

Bratislava, 2011

Práca sa zaoberá vývojom free open source verzie hry mariáš a algoritmu pre počítačového hráča, o ktorom sa predpokladá, že bude lepší ako už existujúce implementácie, ktoré rozhodujú o nasledujúcom ťahu pomocou sady podmienok navrhnutých skúsenými hráčmi. Práca obsahuje aj prehľad problematiky s definíciami základných pojmov teórie hier, prieskum existujúcich implementácií a známych metód, ktoré sa dajú použiť pri vývoji počítačového hráča. Predpokladá sa, že hráč, ktorý je založený na prehľadávaní stavov, bude mať úspech aj pri tejto hre troch hráčov s neúplnou informáciou.

Graficky znázornené výsledky testovania potvrdzujú tento predpoklad.

Kľúčové slová: Mariáš, Teória hier, Minimax, Neúplna informácia.

Abstract

Author: Michal Kováč

Thesis title: Implementácia FOSS verzie hry Mariáš s dôrazom na algoritmus pre počítačového protihráča

Thesis type: Diplomová práca

School: Univerzita Komenského v Bratislave

Faculty: Fakulta matematiky, fyziky a informatiky

Department: Katedra informatiky

Advisor: RNDr. Michal Forišek, PhD.

Bratislava, 2011

This thesis deals with the development of free open source version of card game mariáš popular in Central Europe. First main goal is to create an application with option to choose an algorithm of computer player to play against. The second main goal is to develop computer player that is expected to be stronger than existing implementations, because they decide the next move using a set of conditions proposed by experienced players. The thesis contains an overview of the topic, overview of basic concepts of game theory,

existing implementations and review of existing known methods that can be used in research of new computer player. It is assumed that the player, which is based on game tree search using advanced techniques will succeed also in this three-player game under uncertainty. Graphically displayed test results confirm this assumption.

Keywords: Mariáš, Game Theory, Minimax, Uncertainty.

Predhovor

Moje prvé stretnutie s mariášom nastalo už na strednej škole, keď sme so spolužiakmi hrávali cez prestávky túto hru. Táto hra ma natoľko zaujala svojou robustnosťou a zložitými pravidlami, že som sa rozhodol o nej napísať diplomovú prácu. Existujúce implementácie počítačových hráčov mariášu sú slabé, nevyužívajú potenciál výpočtovej sily počítača. Preto som sa rozhodol spraviť efektívnejšieho hráča, ktorého schopnosti budem môcť prakticky overiť v hre proti iným počítačovým hráčom.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Kapitola 1

Úvod

Témou tejto diplomovej práce je Implementácia FOSS verzie hry Mariáš s dôrazom na algoritmus pre počítačového protihráča. Ciele práce sú:

1. implementovať software, ktorý umožní hráčovi vybrať si dvoch protihráčov a hrať s nimi mariáš
2. vytvoriť prostredie, v ktorom sa bude dať ľahko vyvíjať počítačový hráč
3. vyvinúť algoritmus pre počítačového protihráča, ktorý bude hrať čo najlepšie

V prvej kapitole definujeme základné pojmy. Dokážeme jednoznačnosť pravidiel hry - po každej hráčovej akcii je jednoznačné, čo sa stane. Definujeme aj mariáš podľa terminológie teórie hier. Spomenieme už existujúce implementácie.

V druhej kapitole sa zameriame na vývoj algoritmu pre počítačového hráča. Spravíme prieskum možných metód použiteľných pri vývoji hráča. Základ tvorí prehľadávanie stavov. Spomenieme a pokúsime sa obísť aj rôzne

problémy a úskalia, s ktorými sa stretli ľudia pri vývoji iných hráčov použitím prehľadávania.

Popis implementácie vrátane charakteristiky implementovaných hráčov spravíme v tretej kapitole.

V štvrtej kapitole porovnáme výkony hráčov a vyvodíme z toho závery.

Kapitola 2

Prehľad problematiky

V tejto kapitole definujeme základné pojmy potrebné pre pochopenie pravidiel mariášu. Je to potrebné kvôli jednoznačnosti vyjadrovania v ďalších kapitolách. Spravíme prieskum existujúcich implementácii a priamo kontaktujeme autorov existujúceho softwaru za účelom zistenia použitých algoritmov pri implementácii počítačového hráča.

2.1 Základné definície

Definícia 2.1.1 Karta *je usporiadaná štvorica (rub,líce,hodnota,farba), pričom hodnota a farba sú zobrazené na líci. Farba karty môže byť červeň, zeleň, žaluď, alebo guľa. Hodnota karty môže byť 7, 8, 9, 10, dolník, horník, kráľ alebo eso. Kartu s hodnotou Y nazývame Y. Kartu s farbou X nazývame X. Kartu s farbou X a hodnotou Y nazývame X Y s vyskloňovaným X podľa rodu Y.*

Príklad 2.1.1 Príklad: Ak má karta hodnotu eso a farbu zeleň, tak ju nazývame zelené eso, lebo eso je stredného rodu.

Definícia 2.1.2 Balíček kariet *je množina kariet, ktoré majú rôzne líca a rovnaký rub.*

Poznámka 2.1.1 Sú 4 farby a 8 hodnôt, takže je 32 možností, ako sa dajú tieto vlastnosti kombinovať. Balíček teda môže mať maximálne 32 kariet. V mariáši má balíček vždy 32 kariet.

Definícia 2.1.3 *Hráč* **vidí** *kartu, ak vidí jej líce, takže vie zistiť jej vlastnosti.*

Definícia 2.1.4 Ruka *je množina kariet patriaca hráčovi. Hráč má práve jednu ruku a vidí všetky jej karty. Hráči nevidia karty iných hráčov.*

Definícia 2.1.5 Stôl *je množina kariet, ktoré nepatria žiadnej ruke. Ak sú karty na stole lícom nahor, tak ich hodnoty a farby vidí každý hráč. Ak sú rubom nahor, tak ich hodnoty a farby nevidí žiaden hráč.*

2.2 Priebeh hry

Mariáš hrajú traja hráči v dvoch koalíciách. Jeden hráč je **forhont**. Forhont je v prvej koalícii. Ostatní dvaja sú jeho protihráči, alebo súperi. Protihráči sú spolu v druhej koalícii, ale nemôžu sa vzájomne dorozumievať. Po každej hre sa forhontom stáva hráč naľavo od aktuálneho forhonta.

Definícia 2.2.1 Hra *je postupnosť udalostí.* **Mariáš** *je hra, ktorá má nasledovné udalosti (fázy):*

1. *Zamiešanie kariet*

2. *Prvá fáza rozdávania*

3. *Voľba tromfa*

4. *Druhá fáza rozdávania*

5. *Vklad do talóna*

6. *Licitovanie*

7. *10 kôl*

8. *Vyhodnotenie*

2.2.1 Zahájenie hry

Definícia 2.2.2 Zamiešanie kariet *je náhodné preusporiadanie kariet v balíčku.*

Definícia 2.2.3 Rozdávanie kariet *je presunutie určitého počtu kariet z balíčka na ruku určitým hráčom. V mariáši sa rozdáva v dvoch fázach.*

Definícia 2.2.4 *V* **prvej fáze rozdávania** *sa rozdá 7 kariet forhontovi a ostatní hráči dostanú po 5 kariet.*

Definícia 2.2.5 *Vo fáze* **voľba tromfa** *forhont vyberie zo svojej ruky jednu kartu, ktorá bude tromf. Pod pojmom tromf sa niekedy myslí aj farba tromfu, či ľubovoľná karta s rovnakou farbou ako tromf. Tromf vyloží pred seba rubom nahor.*

Definícia 2.2.6 *V* **druhej fáze rozdávania** *sa rozdá 5 kariet každému hráčovi.*

Definícia 2.2.7 *Karty 10 a eso sú* **mastné karty**.

Definícia 2.2.8 *Vo fáze* **vklad do talóna** *forhont vyberie zo svojej ruky 2 karty, ktoré nie sú mastné, a dá ich do talóna. Talón je množina dvoch kariet na stole rubom nahor.*

2.2.2 Licitácia

Definícia 2.2.9 *Vo fáze* **licitácia** *hráči určujú, ktoré z cieľov hry sa budú hrať. Sú tri možné ciele.*

1. **Hra***: na konci si každý spočíta body v štichoch, hlášky a body za posledný štich. Ak má forhont viac bodov, ako ostatní, hra je vyhraná.*

2. **Sedma***: hráč, ktorý ju zahlásil, musí tromfovou sedmou zobrať posledný štich. Ak tú sedmu v poslednom štichu niekto prebije, je to zabitá sedma.*

3. **Stovka***: hráč, ktorý ju zahlásil, musí uhrať sto bodov, pričom do tohto počtu sa ráta iba jedna hláška. Ak stovku zahlásil protihráč forhonta, body sa sčítavajú pre obidvoch protihráčov forhonta.*

Na začiatku licitácie forhont otočí tromf lícom nahor a povie ciele hry. Jeden z cieľov musí byť hra. Následne hráči sa vyjadrujú v smere hodinových ručičiek:

- *Niektorý z cieľov hry, ak ešte nebol zahlásený.*

- *Zvýši hodnotu niektorých cieľov hry. Takéto zvyšovanie sa volá* **flekovanie**. *To môže spraviť len pre také ciele, ktorých hodnota bola naposledy zvýšená protihráčom, alebo hodnota ešte nebola zvýšená, ale cieľ zahlásil*

protihráč. Hodnota sa zvýši na dvojnásobok. Maximálny počet zvýšení hodnoty jedného cieľa je 7. [1]

- *Mlčí.*

Licitácia končí v momente, keď:

- *Forhont mlčí.*
- *Obaja protihráči forhonta hneď po sebe mlčia.*

2.2.3 Prebíjanie

Definícia 2.2.10 Kopa *je jediná postupnosť kariet na stole, ktorá je lícom nahor. Prvá karta na kope sa nazýva spodná karta na kope.*

Definícia 2.2.11 *Akcia, kedy sa karta presunie z ruky hráča na kopu sa nazýva, že hráč* **hrá** *kartu, alebo hráč* **hodí** *kartu. Karta sa dá na vrch kopy.*

[1] Podľa počtu zvýšení hodnoty cieľa sa zvýšenie nazýva:

1. Flek!
2. Re!
3. Tutti!
4. Boty!
5. Kalhoty!
6. Kaiser!
7. Rekaiser!

Definícia 2.2.12 Kolo *je postupnosť troch akcií, pri každej zahrá jeden hráč kartu. Každý hráč zahrá v každom kole práve jednu kartu. Prvú kartu v kole zahrá* **začínajúci hráč**. *V prvom kole je začínajúci hráč forhont. Pokračujú ostatní hráči v smere hodinových ručičiek.*

Definícia 2.2.13 *Na konci každého kola sa vyhodnotia karty, ktoré to kolo zahrali hráči a určí sa hráč, ktorý zoberie všetky karty na kope (ďalej len* **štich***). Štich sa dá pred toho hráča rubom nahor.*

Definícia 2.2.14 *Relácia* **sila karty** $<$ *je neostré úplné usporiadanie na množine hodnôt kariet. Usporiadanie je nasledovné:*

7<8<9<10<dolník<horník<kráľ<eso.

Definícia 2.2.15 *Karta A je* **silnejšia** *ako karta B, ak $A > B$. Tiež hovoríme, že B je* **slabšia** *ako A.*

Definícia 2.2.16 *Karta A* **prebíja kartu** *B, ak je splnená jedna z nasledovných podmienok:*

1. *A má rovnakú farbu ako B a $A > B$.*
2. *A je tromf a B nie je tromf.*

Veta 2.2.1 *Prebíjanie karty je tranzitívne.*

Dôkaz 2.2.2 *Nech A prebíja B a B prebíja C. Potom môžu nastať tri možnosti:*

1. *A má rovnakú farbu ako B a $A > B$, B má rovnakú farbu ako C a $B > C$. Potom A má rovnakú farbu ako C a z tranzitívnosti sily karty vyplýva $A > C$. Takže A prebíja C.*

2. *A má rovnakú farbu ako B a $A > B$, B je tromf a C nie je tromf. Potom A je tromf a C nie je tromf, teda A prebíja C.*

3. *A je tromf a B nie je tromf, B má rovnakú farbu ako C a $B > C$. Potom A je tromf a C nie je tromf, teda A prebíja C.*

Definícia 2.2.17 Vedúca karta *na kope je:*

- *spodná karta na kope, ak ju neprebila žiadna iná karta na kope*
- *karta na kope, ktorá prebila spodnú kartu na kope a prebila všetky karty, ktoré prebili spodnú kartu na kope*

Definícia 2.2.18 *Hráč* **prebíja kopu**, *ak hrá kartu, ktorá prebíja vedúcu kartu na kope.*

Definícia 2.2.19 *Hráč* **priznal farbu**, *ak je na kope aspoň jedna karta a zahral kartu s rovnakou farbou, ako je spodná karta na kope.*

Definícia 2.2.20 Hláška *je dvojica kariet horník a kráľ rovnakej farby.*

Pravidlá prebíjania

Hráč, keď hrá kartu, musí dodržať nasledovné pravidlá:

1. Ak môže priznať farbu, tak musí priznať farbu.
2. Ak môže prebiť kopu, tak musí prebiť kopu.
3. Ak nemôže priznať farbu, ani nemôže prebiť a má na ruke tromf, musí hrať tromf.

4. Nemôže hrať kráľa zo žiadnej hlášky, ktorú má na ruke.

Príklad 2.2.1 Tromf je červeň, na kope je guľový dolník, ktorý bol prebitý červeným esom. Hráč Pišta má na ruke zelenú deviatku, červeného dolníka, červeného horníka a červeného kráľa. Prvé pravidlo sa neuplatňuje, lebo nemôže priznať farbu, lebo nemá guľu. Druhé pravidlo sa tiež nemôže uplatniť, lebo vedúca karta na kope je červené eso a Pišta na ruke silnejšiu kartu nemá. Tretie pravidlo sa ale uplatní, lebo má na ruke červeň, tak bude musieť hrať červeň. Štvrté pravidlo sa tiež uplatní, lebo má červenú hlášku. Preto nemôže zahrať červeného kráľa. Ostáva mu na výber hrať červeného dolníka alebo horníka. V obidvoch prípadoch ostane červené eso vedúcou kartou, preto hráč, ktorý ho zahral, berie štich.

Štich berie hráč, ktorý ako posledný prebil kopu. Ak nikto neprebil kopu, tak štich berie hráč, ktorý dal spodnú kartu.

2.2.4 Vyhodnotenie

Zisk forhonta je súčtom ziskov pre každý cieľ hry. Ak bol tromf červeň, všetky zisky sú dvojnásobné. Zisk koalície súperov je opačný ako zisk forhonta. To, čo je pre forhonta zisk, je pre súperov strata. Zisk koalície súperov sa rozdelí rovným dielom medzi hráčov v tej koalícii.

Vyhodnotenie hry

Každý hráč si spočíta body nasledovne:

1. Za každú mastnú kartu vo svojich štichoch 10 bodov.

2. Za každú hlášku, ktorú zahral, 20 bodov. Ak je hláška tromfová, tak 40 bodov.

3. Za posledný štich 10 bodov.

Ak niektorá koalícia má aspoň 100 bodov, ale nehrala sa stovka, volá sa to **tichá stovka**. Každých 10 bodov nad 90 hodnotu hry zdvojnásobí. Ak má forhont viac bodov ako jeho protihráči spolu, tak hra je vyhraná, inak je hra prehraná. Ak je hra vyhraná, zisk forhonta za hru je dvojnásobok hodnoty hry.

Vyhodnotenie sedmy

Hodnota nefleknutej sedmy je 2. Každé fleknutie ju zdvojnásobí. Ak je sedma zabitá, tak sa jej hodnota zdvojnásobí. Ak sa nehrala sedma, ale:

- niekto zobral posledný štich tromfovou sedmou, volá sa to **tichá sedma**. Hodnota tichej sedmy je 1.

- niekto zahral sedmu v poslednom štichu, ale bola prebitá inou kartou, volá sa to **zabitá tichá sedma**. Hodnota zabitej tichej sedmy je 1.

Ak je sedma vyhraná, koalícia, ktorá ju hlásila, si zvýši zisk o dvojnásobok hodnoty sedmy. V opačnom prípade si zisk takto zvýši súperova koalícia.

Vyhodnotenie stovky

Hodnota nefleknutej stovky je 2. Každé fleknutie ju zdvojnásobí. Koalícia, ktorá hrala stovku, si spočíta body tým istým postupom ako pri vyhodnotení hry, pričom sa počíta maximálne jedna hláška.

- Ak má koalícia aspoň 100 bodov, stovka je vyhraná, inak je prehraná.
- Ak je stovka vyhraná, k bodom sa pripočítajú aj body za hlášky, ktoré sa doteraz nepočítali. Každých 10 bodov nad 100 hodnotu stovky zdvojnásobí. Teoretické maximum je 190 bodov, čo predstavuje zisk $2^9 = 512$. Ak je na stovku rekaiser, tak sa to ešte vynásobí $2^7 = 128$. Výsledný zisk za stovku by bol 65536.
- Ak je stovka prehraná, a koalícia, ktorá hlásila stovku, uhrala x bodov, zisk koalície protihráča bude rovnaký ako v prípade, keby bola stovka vyhraná s $190 - x$ bodmi.

2.3 Zaradenie v terminológii teórie hier

Nasledujúce definície, ak nie je napísané inak, sú čerpané z [Ras07].

Definícia 2.3.1 Hra *pozostáva z hráčov, akcií, ziskov a informácií.*[2]

Definícia 2.3.2 Hráč *je jednotlivec, ktorý robí rozhodnutia.*

Definícia 2.3.3 Príroda *je pseudo-hráč, ktorý robí náhodné rozhodnutia podľa danej pravdepodobnostnej distribúcie.*

Definícia 2.3.4 Akcia *(alebo ťah) hráča P je rozhodnutie, ktoré robí hráč P.*

Definícia 2.3.5 Stav *je okamih v hre, v ktorom buď niektorý hráč, alebo príroda, robí rozhodnutie, alebo hra končí.*

[2] podľa originálu: PAPI = {players, actions, payoffs, information}

Definícia 2.3.6 Koncový stav *je stav, v ktorom hra končí.*

Poznámka 2.3.1 Pre každý koncový stav je priradený zisk každého hráča. Cieľom každého hráča je výberom akcií maximalizovať svoj zisk.

Definícia 2.3.7 Informácia hráča *P je množina stavov, o ktorých hráč P vie, že môžu byť aktuálne, ale nevie ich rozlíšiť priamym pozorovaním.*

Poznámka 2.3.2 Pravidlá hry zaručujú, že informácia každého hráča obsahuje aspoň jeden stav - aktuálny.

Definícia 2.3.8 Stratégia *je funkcia $I \rightarrow A$, kde I je množina informácií a A je množina akcií.*

Poznámka 2.3.3 Hráči si vyberajú stratégie, podľa ktorých pre danú informáciu volia nasledovnú akciu.

Definícia 2.3.9 *Hra s nulovým súčtom je podľa [TS01] taká hra, kde v ľubovoľnom koncovom stave je súčet ziskov všetkých hráčov nulový.*

Definícia 2.3.10 *Hra so symetrickou informáciou je taká, že pre každého hráča i jeho informácia v:*

1. *stave, v ktorom robí rozhodnutie*
2. *koncovom stave*

je podmnožinou prieniku informácií ostatných hráčov.

Definícia 2.3.11 *Hra s asymetrickou informáciou je hra, ktorá nie je so symetrickou informáciou.*

Definícia 2.3.12 *Hra s neúplnou informáciou je taká, že v nej existuje stav, v ktorom robí rozhodnutie príroda a aspoň jeden hráč nepozná toto rozhodnutie.*

Definícia 2.3.13 *Hra s úplnou informáciou je hra, ktorá nie je s neúplnou informáciou.*

Definícia 2.3.14 *Hra s dokonalou informáciou je podľa [TS01] taká hra, kde hráč vo svojom ťahu vie všetko o rozhodnutiach, ktoré sa dovtedy spravili.*

Definícia 2.3.15 *Hra s dokonalou informáciou je podľa [Ras07] taká hra, v ktorej informácia každého hráča na ťahu obsahuje práve jeden stav.*

Poznámka 2.3.4 Keď predpokladáme, že je jeden počiatočný stav, tak definícia 2.3.14 je špecifickejšia ako 2.3.15, lebo vyžaduje aj znalosť všetkých predošlých rozhodnutí. Rozdiel môže nastať pri hre, kde sa dá rôznymi rozhodnutiami dostať do toho istého stavu. Pre naše potreby stačí definícia 2.3.15. Ak by história predošlých rozhodnutí bola dôležitá, zahrnie sa táto história do stavu a vtedy sú definície ekvivalentné.

Definícia 2.3.16 *Náhodná hra je taká hra, kde existuje stav, v ktorom rozhodnutie robí príroda.*

Veta 2.3.1 *Hra s dokonalou informáciou je aj so symetrickou a úplnou informáciou.*

Dôkaz 2.3.2 *Podľa definície 2.3.15 pri hre s dokonalou informáciou informácia každého hráča na ťahu obsahuje práve jeden stav. Podľa poznámky*

2.3.2 je to aktuálny stav, ktorý patrí do informácie každého hráča, preto je to hra aj so symetrickou informáciou.

Podľa definície 2.3.14 vie hráč o všetkých rozhodnutiach, preto neexistuje rozhodnutie prírody, ktoré by nepoznal. Preto je to hra s úplnou informáciou.

Príklad 2.3.1 Šach je hra s úplnou symetrickou dokonalou informáciou s nulovým súčtom. Informácia každého hráča na ťahu obsahuje práve jeden stav.

Príklad 2.3.2 Pexeso je hra s neúplnou symetrickou nedokonalou informáciou s nenulovým súčtom. Kartičky sa rozložia náhodne, hráči nevedia, ako sú rozložené a odkrývaním má stále každý hráč rovnaké informácie.

Príklad 2.3.3 Lodičky sú hra s úplnou asymetrickou nedokonalou informáciou. Všetky rozhodnutia robia hráči, ale každý pozná svoje rozloženie lodí a nepozná súperove.

Príklad 2.3.4 Mariáš je hra s neúplnou asymetrickou nedokonalou informáciou s nulovým súčtom. Každý hráč síce vie, aké karty majú ostatní hráči spolu, ale nevie, ako sú rozdelené. Zisk jednej koalície je strata druhej koalície, preto má nulový súčet. Jediné náhodné rozhodnutie je na začiatku pri miešaní kariet a nikto nevie, ako sú karty preusporiadané, preto je to hra s neúplnou informáciou.

2.4 Existujúce implementácie

2.4.1 Mariáš 2.7

Mariáš 2.7 od Martina Staufčíka ([St04]). Vydaný v roku 2004 ako shareware so skúšobnou dobou 30 dní. Od verzie 2.6 sú zverejnené zdrojové kódy. Kód je naprogramovaný v delphi, počítačoví hráči rozoberajú všetky prípady a rozhodujú sa tak, ako by sa rozhodol skúsený hráč.

2.4.2 Mariáš z www.marias.ik.cz

Slúži hlavne na online hranie ligy na www.talon.cz. Je implementovaná aj umelá inteligencia, tvorca uviedol, že do programu zahrnul svoje myslenie pri hraní, takže sa rozhoduje na základe podmienok z vlastných skúseností.

2.4.3 Becherovka Mariáš

[Fia09]: "Umělá inteligence simuluje ostatní hráče, kteří se Vás snaží sedřít z kůže.". Stratégia tiež vychádza zo skúseností tvorcu, pozostáva z veľkého množstva podmienok.

Kapitola 3

Prehľad vhodných metód riešenia

V tejto kapitole spomenieme známe metódy teórie hier a analyzujeme ich možné využitie pri našej implementácii.

3.1 Stavový priestor

Prvý krok k formalizácii hry je definovanie stavového priestoru. Ten pozostáva zo stavov a ťahov. Stav predstavuje množinu relevantných informácií. Priradenie stavu množine týchto informácií je jednoznačné.

Definícia 3.1.1 *Stav hry mariáš zahŕňa nasledovné informácie:*

- *Kto je forhont.*
- *Aká je fáza hry.*
- *Zvolený tromf.*

- *Karty v talóne.*
- *História licitovania.*
- *História hraných kariet.*
- *Karty na rukách hráčov.*

Definícia 3.1.2 Ťah *je usporiadaná dvojica stavov.*

Poznámka 3.1.1 Je rozdiel medzi ťahom a povoleným ťahom. Ťah môže existovať medzi ľubovoľnými dvomi stavmi, avšak pravidlá hry vymedzujú, ktoré ťahy sú povolené. Nakoľko sa nepovolené ťahy už nebudú spomínať, pod pojmom ťah budeme rozumieť povolený ťah. Každému stavu je priradený hráč, ktorý je na ťahu. To znamená, že rozhoduje, do ktorého stavu sa dostane hra po jeho ťahu.

Definícia 3.1.3 Stavový priestor *je usporiadaná dvojica* (S, T)*, kde* S *je neprázdna konečná množina stavov a* T *je množina povolených ťahov na* S*.*

Poznámka 3.1.2 Stavový priestor je definovaný podobne ako orientovaný graf, preto sa aj často reprezentuje práve orientovaným grafom, ktorý sa zvykne nazývať herný strom.

3.2 Prehľadávanie

Stavový priestor nám umožňuje sa pozrieť na hru takým pohľadom, že pri sledovaní hry sa dá v každom momente určiť stav, v ktorom sa nachádza a

nájsť stavy, ktoré sa dajú dosiahnuť ťahom. Mariáš je hra, v ktorej sa z nejakého stavu nedá dostať niekoľkými ťahmi do toho istého stavu. V stavovom priestore teda nie sú cykly. To znamená, že tranzitívny uzáver povolených ťahov je ostré neúplné usporiadanie. Preto nutne musia existovať stavy, z ktorých už nejde žiaden ťah. Také stavy voláme **koncové**.

V koncových stavoch sa hra končí, preto, aby hra mala zmysel, sú k nim priradené zisky jednotlivých hráčov. Cieľom každého hráča je dosiahnuť, aby sa hra dostala do takého koncového stavu, aby jeho zisk bol v ňom čo najvyšší. V tejto kapitole hľadáme čo najefektívnejší spôsob, akým v danom stave vybrať ťah, ktorým sa dá dosiahnuť čo najvyšší zisk.

Jednou možnosťou je použiť prehľadávanie stavového priestoru. Je to spôsob, akým systematicky traverzujeme stavy za účelom nájdenia koncových stavov. Začína sa v jednom vrchole a traverzujú sa stavy, do ktorých sa dá dostať postupnosťou ťahov. Dva základné spôsoby prehľadávania sú do hĺbky a do šírky.

3.2.1 Prehľadávanie do hĺbky

Keď sme v stave S, z ktorého vedú ťahy do stavov $S_1, S_2, \ldots S_b$. Prehľadávanie do hĺbky rekurzívne pokračuje vo vrchole S_1, keď skončí, potom sa prehľadáva S_2, atď. V každom momente si musíme pamätať cestu, po ktorej sa budeme vracať po skončení prehľadávania. V každom stave na tejto ceste si musíme pamätať, ktoré vetvy sme už prehľadali a ktoré ešte potrebujeme prehľadať. Preto je pamäťová zložitosť $O(bd)$, kde b je faktor vetvenia a d počet ťahov do konca hry.

3.2.2 Prehľadávanie do šírky

Prehľadávanie do šírky si udržuje frontu vrcholov, z ktorých sa ešte netraverzovalo ďalej. Keď sa nachádza v stave S, z ktorého edú ťahy do stavov $S_1, S_2, \ldots S_b$, pridá si tieto stavy do fronty na koniec. Prehľadávanie pokračuje stavom, ktorý je prvý vo fronte. Dĺžka fronty môže byť až $O(b^d)$, preto pre praktické účely budeme používať len prehľadávanie do hĺbky. Neskôr budeme ale potrebovať niektoré vlastnosti prehľadávania do šírky, napríklad vyváženú hĺbku prehľadaných vetiev, keď sa nestihnú prehľadať všetky.

V každom prípade prehľadávame celý stavový priestor, ktorý má rádovo $O(b^d)$ stavov.

3.3 Minimax

Pomocou prehľadávania vieme nájsť koncové vrcholy. V nich potom vieme ohodnotiť hráčov. Ak by sme si vybrali najväčšie ohodnotenie a išli priamo doň, nemusí sa nám to podariť, lebo nie všetky ťahy vie vykonať hráč, ktorý tam chce ísť. Každý ťah v stavovom priestore je jednoznačne priradený hráčovi. Dobrý algoritmus, ktorý pozná stavový priestor by sa mal rozhodovať tak, že spomedzi všetkých možných ťahov protihráča v najhoršom prípade dosiahne čo najvyššie ohodnotenie.

Minimax je algoritmus, ktorý pre daný stav a daného hráča rozhodne, ktorým ťahom sa maximalizuje zisk v najhoršom prípade. Najhorší prípad pre hráča je taký, že protihráč bude hrať tak, aby jeho hodnotenie bolo čo najmenšie. Pre každý stav definujeme jeho minimax hodnotu nasledovne:

1. Ak je to koncový stav, minimax hodnota je rovná zisku hráča.

2. Ak je v tom stave na ťahu hráč, ktorý chce maximalizovať zisk (Max), minimax hodnota je maximum z minimax hodnôt stavov, do ktorých sa dá dostať jedným ťahom.

3. Ak je v tom stave na ťahu protihráč (Min), minimax hodnota je minimum z minimax hodnôt stavov, do ktorých sa dá dostať jedným ťahom.

Podľa [RN03], Minimax Theorem tvrdí, že minimax hodnota naozaj maximalizuje zisk v najhoršom prípade. Platí to ale len pre hry dvoch hráčov. Aj keď sa na prvý pohľad zdá, že pri troch hráčoch v dvoch koalíciách bude Minimax Theorem aplikovaný na koalície platiť tiež, ale nie je to tak. Problém je v tom, že hráči v koalícii sa nemôžu dohodnúť. [HN09] formálne dokazuje, že vo všeobecnosti Minimax Theorem pre troch hráčov, z ktorých dvaja sú v koalícii, neplatí.

Napriek tomu sa dajú dobré výsledky dosiahnuť aj použitím minimaxu. V prípade hráča, ktorý spolupracuje proti forhontovi, nie je výhodné brať najhorší prípad. Práve naopak, ak predpokladáme, že spolupracujúci hráč sa snaží maximalizovať zisk, tak môžeme zobrať najlepší prípad ako ťah, ktorý zahrá. Táto stratégia zlyhá len v prípade zákerného hráča. Preto budeme samostatne testovať hráča, ktorý sa spolieha na svojho spoluhráča a paranoidného hráča, ktorý ide na istotu.

3.4 Alfabeta orezávanie

Alfabeta orezávanie je metóda, ktorá redukuje stavový priestor. Pri prehľadávaní nemusíme prehľadávať všetky stavy. V istom prípade sa môžeme

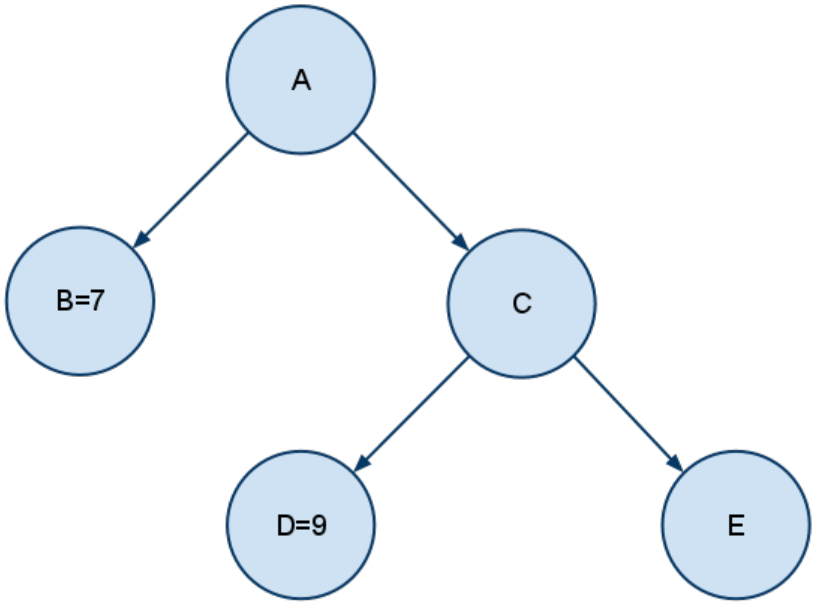
rozhodnúť, že niektorý podstrom nemusíme prehľadávať, lebo nech by bolo ohodnotenie akékoľvek, už by to nezmenilo minimax hodnotu daného stavu.

Príklad 3.4.1 Nech v stave A sa rozhoduje protihráč, chce minimum ohodnotení. Prehľadávaním sa zistilo, že ohodnotenie B je 7. V stave C sa rozhoduje hráč Max. Ako postupne prehľadáva a narazí na stav, ktorého ohodnotenie je vyššie ako B, napríklad na obrázku 3.1 $D = 9 > 7 = B$, potom je už zbytočné prehľadávať ďalej, lebo hráč Max si už nevyberie menej ako D, takže $C \geq D$. Keďže $D \geq B$, tak aj $B \leq C$ a hráč Min si nevyberie C, keď si môže vybrať B, ktoré je menšie.

V istom momente sa teda môžu ignorovať niektoré stavy a vrátiť sa o úroveň vyššie. Efektívnosť alfabeta orezávania závisí od poradia, v akom prehľadávame ďalšie stavy. Podľa [RN03], ak budeme poradie vyberať optimálne, stačí prehľadať $O(b^{m/2})$ namiesto $O(b^m)$, kde b je faktor vetvenia a m je počet ťahov do konca hry.

3.5 Heuristika

Niekedy je stavový priestor taký veľký, že v reálnom čase sa nestihne celý prehľadať. V danom časovom limite dokáže prehľadať len niektoré stavy. Naskytá sa otázka: musia sa v každom stave overiť všetky možné ťahy? Ak by sa vynechal čo i len jeden, môže sa nám stratiť dôležitá informácia. Preto nemá zmysel prehľadať len niektoré vetvy. Buď všetky alebo žiadna. Niektoré sa možno preskočia vďaka alfabeta orezávaniu, v sekcii 3.4 je ukázané, že týmto postupom nestratíme žiadnu informáciu.



Obr. 3.1: Alfabeta orezávanie

Keď sa nestihne prehľadať celý stavový priestor, ostáva jediná možnosť - v niektorých stavoch, ktoré nie sú koncové, nevnoriť sa do žiadnej vetvy. Ostáva teda otázka, akú bude mať taký stav minimax hodnotu. Odpoveď je použitie **heuristickej funkcie**, ktorá odhadne očakávaný zisk v danom stave.

V sekcii 3.4 sme chceli prehľadávať v optimálnom poradí, aby sa často uplatnilo orezanie. To poradie sa dá aproximovať práve takouto heuristickou funkciou. Keď začneme prehľadávať vetvy, kde je najvyšší očakávaný zisk, je vyššia pravdepodobnosť, že ďalšie vetvy sa nebudú prehľadávať kvôli

nižšiemu zisku.

3.6 Iteratívne prehlbovanie

V tejto sekcii vyriešime problém, kedy odhadovať minimax hodnotu pomocou heuristickej funkcie. Predpokladajme, že máme časový limit, ktorý keď vyprší, nemôžeme ďalej prehľadávať.

Definícia 3.6.1 *Hĺbka stavu je najmenšia vzdialenosť od počiatočného stavu.*

Pre daný časový limit chceme maximalizovať hĺbku, po ktorú prehľadáme všetky stavy. **Iteratívne prehlbovanie** prehľadá celý stavový priestor do hĺbky 1, potom do hĺbky 2, a tak ďalej, až kým nevyprší časový limit. V cieľovej hĺbke sa pre nekoncové vrcholy aplikuje heuristická funkcia.

Iteratívne prehlbovanie kombinuje výhody prehľadávania do hĺbky a do šírky. Podobne ako pri prehľadávaní do šírky platí pre každé dva stavy s, t, kde s je v menšej hĺbke ako t, že prehľadávanie príde k stavu s skôr ako k stavu t. Prehľadávanie do šírky má ale veľkú nevýhodu - pamäťovú zložitosť. Tá je $O(b^d)$, kde b je vetviaci faktor a d hĺbka, do ktorej prehľadávame. Podľa [RN03] iteratívne prehlbovanie má pamäťovú zložitosť rovnakú ako prehľadávanie do hĺbky, a to $O(bd)$.

3.7 Monte Carlo

Doteraz spomenuté metódy fungujú, ak prehľadávame zo známeho počiatočného stavu. V praxi to znamená, že máme úplnú informáciu a vieme, v ktorom stave sa nachádza hra. Mariáš úplnú informáciu nemá, preto nevieme vždy

určiť počiatočný stav. Na základe akcií zvyšných hráčov, o ktorých vieme, že hrali podľa pravidiel, môžeme niektoré stavy vylúčiť, lebo určite neprichádzajú do úvahy.

Príklad 3.7.1 Tromf je červeň. Hráč A zahrá zeleného dolníka, hráč B priznáfarbu, dá zelenú deviatku a hráč C guľového horníka.

Keby B mal silnejšiu zelenú kartu, musel by prebiť. Keďže neprebil, tak ju nemá. Stavy, v ktorých hráč B má na ruke silnejší zeleň ako zelený dolník, nepripadajú do úvahy. Podobne hráč C nepriznal farbu, ani nedal tromf, takže stavy, v ktorých má na ruke zeleň alebo červeň, nepripadajú do úvahy.

Stavy, ktoré ostali, tvoria množinu možných počiatočných stavov. Keďže ďalšie informácie o pravdepodobnosti jednotlivých stavoch nemáme, zisk sa bude rovnať priemernému zisku pre všetky prehľadané počiatočné stavy. Ak je možných stavov príliš veľa, treba vybrať vhodnú podmnožinu. Veľkosť tejto podmnožiny nazývame **šírka prehľadávania**. Metóda Monte Carlo spočíva v tom, že sa počiatočné stavy generujú náhodne z množiny možných stavov. Používa sa aj v iných kartových hrách, kde sa berú štichy, napríklad [Fong05].

3.8 Kompromis medzi hĺbkou a šírkou

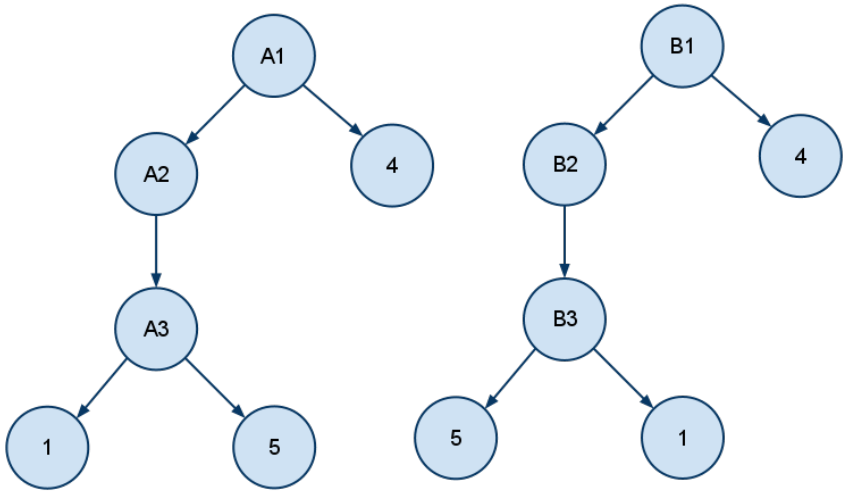
Veľkosť vybranej podmnožiny počiatočných stavov zo sekcie 3.7 a maximálna hĺbka prehľadávania zo sekcie 3.6 sú dva parametre, ktoré treba vhodne nastaviť v závislosti od časového limitu. Ak budeme robiť plytké prehľadávanie, môžeme si dovoliť väčšiu šírku prehľadávania. Naopak, do veľkej hĺbky stihneme prehľadať len málo možných stavov.

Podobne ako [Smith06], naše riešenie zlomok času venuje generovaniu možných rozdaní a zvyšný čas sa využije na iteratívne prehlbovanie.

3.9 Známe problémy

3.9.1 Problém fúzie stratégií

Podľa [FBM98], ak existuje viac možných rozdaní a pre každé prehľadáme celý stavový priestor pomocou minimaxu, môže dôjsť k problému fúzie stratégií (strategy fusion).



Obr. 3.2: Problém fúzie stratégií

Príklad 3.9.1 Na obrázku 3.2 hrajú dvaja hráči, pričom začínajúci nevie, či

je v stave A1 alebo B1. Chce maximalizovať svoj zisk. Ak použijeme minimax pre každý stav, v ktorom sa môžeme nachádzať, a vyberieme ťah, pre ktorý je priemerný zisk maximálny, stane sa nasledovné:

1. V stavoch A3 a B3 funkcia minimax vráti maximum z možných ziskov pod nimi. V oboch prípadoch je to 5.

2. V stavoch A2 a B2 vráti minimum, čo je 5.

3. Pre stavy A1 a B1 sa spriemerujú zisky pre každý možný ťah. Pre ťah "doľava" je to 5 a pre "doprava" 4. Víťazný ťah je "doľava".

Problém môže nastať, keď po dvoch ťahoch stále nebudeme vedieť, či sme v stave A3 alebo B3. Priemerný zisk bude len 3.

Kde nastala chyba? Pri minimaxe sme mali zafixované rozdanie. Predpokladali sme, že vo fixnom rozdaní budeme vždy vedieť vybrať maximum. Ale ak už príde na to rozhodnutie, stále nemusíme vedieť, ako sú karty rozdané, preto to nie je vždy pravda.

V akých situáciach môže nastať tento problém? Musia sa hrať aspoň dva ciele, pričom musí nastať nasledovná voľba:

1. Voľbou istoty si zaistíme výhru cieľa s vyššou hodnotou a prehru cieľa s nižšou hodnotou.

2. Voľbou rizika si zaistíme výhru menšieho cieľa, ale riziko cieľa s vyššou hodnotou.

Navyše musí platiť, že pri voľbe rizika budeme mať výsledok cieľa s vyššou hodnotou ešte neskôr vo vlastných rukách a v stave, kde o tom budeme rozhodovať, nebudeme mať informácie o dopade jednotlivých rozhodnutí.

Väčšine hráčov stačí z tejto situácie prvá voľba istoty a rizika, aby sa už nevedeli rozhodnúť. Problém fúzie stratégií je ešte špecifickejší druhou podmienkou. Týchto situácií je pomerne málo na to, aby mohol napáchať väčšie škody. Preto tento problém ignorujeme.

Na druhej strane, tento problém sa dá využiť pre náš prospech. Idea algoritmu Trappy minimax je podľa [GR07] účelne vyhľadávať stavy s takými rozhodnutiami, aby súpera "vlákal do pasce".

3.9.2 Problém nestabilných stavov

Pri fixnej hĺbke prehľadávania môže nastať podľa [RN03] problém nestabilných stavov[1]. Sú to také stavy, ktorých heuristické ohodnotenie je vysoké, ale v najbližších ťahoch sa nečakane zníži. Napríklad, ak sa pri šachu uplatňuje populárna heuristická funkcia, ktorá predstavuje materiálnu výhodu figúrok, ktoré sú v hre, ale nezohľadňuje sa ich pozícia, môže sa vysoko ohodnotiť stav, po ktorom nastane vyhodenie dámy, čo predstavuje veľký zvrat v hre.

Sú známe dva prístupy k tomuto problému.

- Zohľadňovať v heuristickom ohodnocovaní aj pozíciu figúrok, napríklad v šachu počet ohrození jednotlivých figúrok.
- Heuristické ohodnotenie aplikovať len v stabilných stavoch. Nestabilné stavy treba utíšiť napríklad lokálnym prehľadaním (quiescence search). [RN03] bližšie nešpecifikuje, v ktorých stavoch by sa malo toto prehľadávanie spraviť, ale uvádza, že stačí uvažovať len niektoré významné ťahy, ktoré stabilizujú stav, napr. branie figúrky.

[1] Podľa originálu: non-quiescent states problem

Nakoľko pri iteratívnom prehlbovaní máme pri každej iterácii fixnú hĺbku, druhé riešenie by bolo problematické. V našom heuristickom ohodnocovaní stavu sa zohľadňuje okrem získaných bodov v štichoch aj sila kariet na rukách hráčov, čo realizuje prvý spomínaný prístup.

3.9.3 Problém horizontu

Problém horizontu nastáva, ak nastane v budúcnosti nezvratná udalosť, ktorá veľmi zníži ohodnotenie, ale dá sa oddialiť. [RN03] uvádza ukážkový príklad zo šachu, keď biely vie pešiakom prejsť na dámu, ale čierny mu to môže oddialiť ohrozovaním kráľa. Dokáže to robiť 14 ťahov a potom sa už bielemu podarí získať dámu, čo mu zaručí výhru. Minimax sa snaží nájsť optimálny ťah, preto si myslí, že sa dá vyhnúť tomu, aby biely získal dámu. Riešenia sú dve:

- Pri ohodnocovaní stavu brať do úvahy aj pozíciu figúrok podobne ako v subsekcii 3.9.2
- Dopredné orezávanie[2]. Je to variant techniky zoraďovania stavov podľa heuristického ohodnotenia zo sekcie 3.5. Niekedy sa môžu dopredu orezať vetvy, ktoré majú príliš nízke ohodnotenie. Pri ukážke v [RN03] ostane len jedna vetva a môžeme teda zvýšiť hĺbku prehľadávania.

V mariáši sa analogicky dajú oddialiť stavy, ktorých heuristické ohodnotenie je nízke, hraním slabých kariet, ktoré príliš nezmenia ohodnotenie aktuálneho stavu. Aby sme predišli ďalším rizikám súvisiacich s dopredným

[2]Forward pruning

orezávaním, uspokojili sme sa s prvým riešením - ohodnocovanie kariet na rukách, ktoré pomáha aj pri probléme v subsekcii 3.9.2.

Kapitola 4

Implementácia

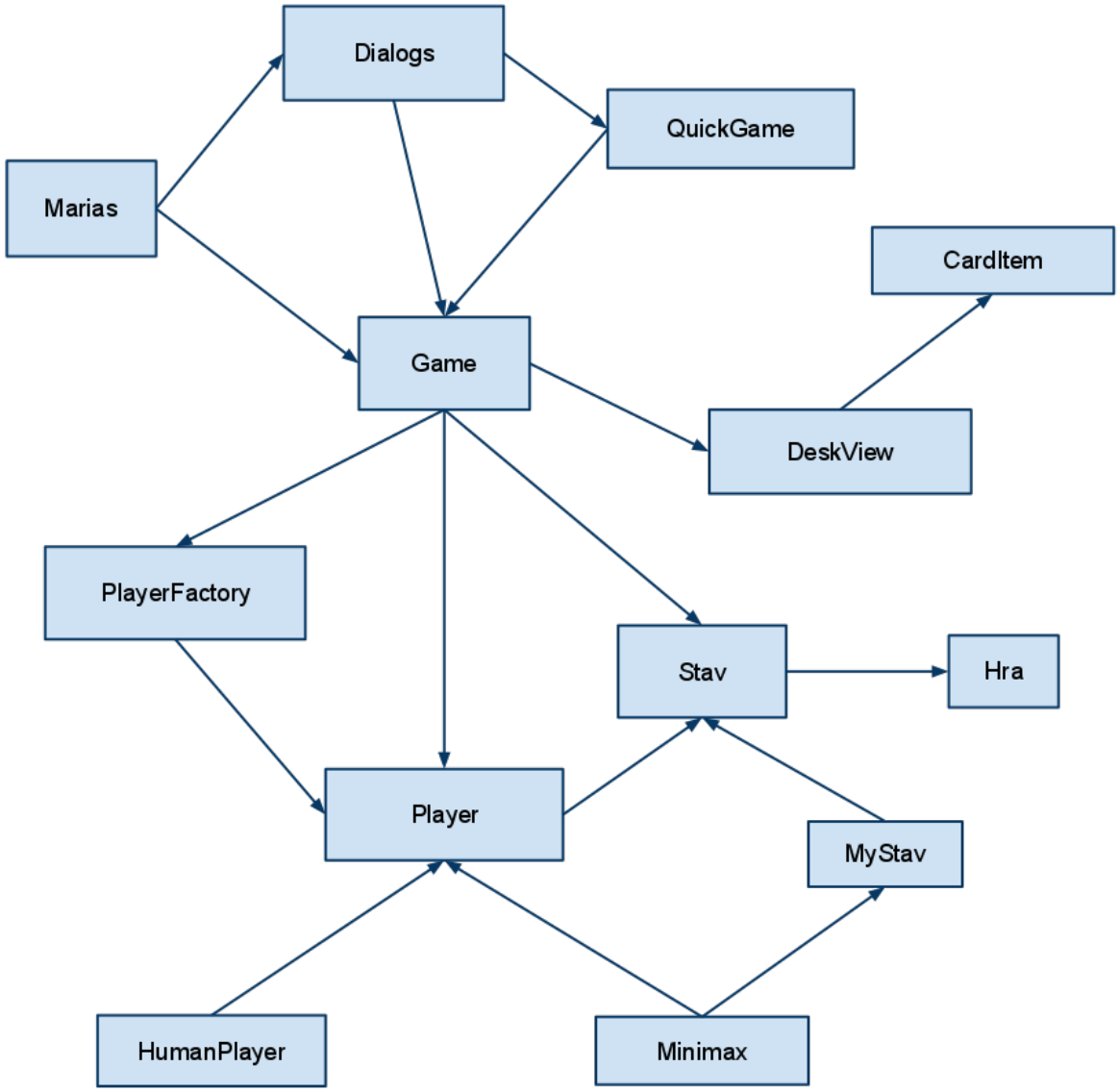
Vyvíjaný software OpenMarias je na stránke http://www.sourceforge.net/openmarias pod licenciou GPL. Použitý je programovací jazyk C++ s frameworkom Qt. Preto sa zdrojový kód dá skompilovať pre platformy, ktoré sú podporované Qt. To zahŕňa napríklad Windows aj Linux. Rozhranie je dostupné v slovenčine a angličtine a jednoducho sa dajú pridať ďalšie preklady.

4.1 Popis hlavných tried

V tejto subsekcii sú popísané niektoré triedy, z ktorých najvýznamnejšie sú znázornené v UML Diagrame na obrázku 4.1.

4.1.1 Marias

Marias je hlavné okno aplikácie, ktoré je zdedené od triedy QMainWindow. Poskytuje štandardné užívateľské rozhranie - "Nová hra", "Nastavenia", "O aplikácii", "Koniec". Vytvára a ukončuje inštancie tried Game a DeskView.



Obr. 4.1: UML diagram

Na začiatku spustí intro.

4.1.2 DeskView

Singleton trieda DeskView zaobaľuje grafickú scénu a poskytuje metódy na vykresľovanie a animácie. Využíva najnovší animačný framework so zloženými paralelnými a sériovými animáciami. Animovať sa dajú okrem polohy aj jednotlivé vlastnosti ako veľkosť a priesvitnosť. Okrem animácií kariet sa využívajú aj animácie textu. Obrázky kariet sa načítavajú z priečinka, takže sa dajú ľahko zmeniť.

4.1.3 Dialogs

Napriek snahe minimalizovať počet dialógov[1] OpenMarias obsahuje 3 dialógy, ktoré nie sú modálne. Výhodou je, že kým je otvorený dialóg, nie je zablokované GUI hlavného okna.

SettingsDialog umožňuje nastaviť niektoré parametre, napríklad rýchlosť animácií, spôsob miešania, mená a typy hráčov na jednotlivých miestach. Ak sú všetci traja hráči počítačoví, môže sa spustiť rýchla hra.

QuickGameDialog ovláda rýchlu hru. Dá sa nastaviť spôsob miešania, počet hier a spustiť simulácia. Simulácia je v samostatnom vlákne QuickGameThread, preto GUI nie je blokované. Tým sa umožnilo zobrazovanie počtu odobratých hier pomocou progressbaru.

BiddingDialog je ovládací panel na licitovanie. Konfigurácia troch prepínačov predstavuje flekovanie jednotlivých cieľov hry. Otvorí sa automaticky,

[1]Dôvod na minimalizovanie počtu dialógov je podobný ako pri tzv. ”vyskakovacích oknách” webových aplikácií. Jednoducho užívateľom prekážajú.

keď forhont zvolí karty do talóna a zavrie sa súčasne s koncom licitácie.

4.1.4 Game

Game je hlavná trieda, ktorá spravuje hráčov a stav hry. Rozhoduje, čo sa stane pri jednotlivých udalostiach v metóde animationFinished.

4.1.5 Stav

Stav obsahuje informácie o hre, ktoré sú dostupné pre všetkých. Poskytuje metódy na vyhodnotenie štichu a vyhodnotenie hry.

4.1.6 Profiler

Profiler umožňuje spúšťať meranie času, sledovať, koľko času uplynulo od spustenia a zastaviť meranie. Hráči tak majú možnosť rozhodovať sa v algoritmoch aj podľa uplynutého času.

4.1.7 RandomGenerator

Implementovali sme aj vlastný generátor náhodných čísel. Je to užitočné preto, že keď chceme zaručiť rovnaké podmienky pre iných hráčov, musíme nastaviť rovnaký random seed. Preto každý hráč dostane vlastnú inštanciu generátora, aby neovplyvňoval rozdávanie kariet.

4.1.8 PlayerFactory

PlayerFactory realizuje návrhový vzor factory, zapuzdruje vytváranie inštancií konkrétnych hráčov (podľa [CG02]).

4.1.9 Player

Player predstavuje abstraktného hráča. Každý hráč, ktorý extenduje triedu Player, musí implementovať voľbu tromfa, talóna, licitovanie a hranie karty. Medzi metódy, ktoré nie sú virtuálne, patrí validate(), ktorá zistí, či hráč môže zahrať zvolenú kartu[2]. Metóda getLegalList() vráti zoznam takých kariet. Udržuje informácie o kartách na ruke a získaných štichoch.

4.1.10 HumanPlayer

Špeciálny typ hráča je HumanPlayer. Návratové hodnoty metód sa ignorujú a čaká sa, kým hráč klikne na kartu v grafickej scéne.

4.2 Implementácia počítačového hráča

Trieda Player poskytuje niekoľko virtuálnych metód. Implementácia počítačového hráča teda znamená extendovanie triedy Player a implementáciu virtuálnych metód zo subsubsekcie 4.1.9, menovite:

- Voľba tromfa
- Voľba talóna
- Licitovanie
- Hranie karty

[2]V neposlednom rade aj či tú kartu má naozaj na ruke.

4.2.1 RandomPlayer

RandomPlayer v každej metóde zistí, aké má možnosti, aby neporušil pravidlá a vyberie si náhodnú. Samo osebe by to nebolo až také zlé, ale náhodným rozhodovaním pri licitácii si zaručuje istú prehru.

4.2.2 MinPlayer

MinPlayer vyberie spomedzi kariet, ktoré môže zahrať, najmenšiu. Ak má viac najmenších, tak vyberie tú, ktorá je najviac vpravo. Pri licitácii sa vyjadruje len vtedy, keď musí.

4.2.3 MaxPlayer

MaxPlayer hrá podobne ako MinPlayer, ale vyberá vždy najväčšiu kartu. Pri licitácii sa vyjadruje vždy, preto bude asi súťažiť s RandomPlayerom o posledné miesto.

4.2.4 SmartPlayer

SmartPlayer pozostáva zo sady podmienok podobne ako už existujúce implementácie zo sekcie 2.4. Tento hráč už hrá na celkom dobrej úrovni, začiatočníci nemajú veľkú šancu. Mierne nadpriemerný hráč, ktorý už pozná veľa rôznych fínt[3], ho avšak už poväčšine dokáže porážať.

[3]Napríklad autor tejto práce

4.2.5 MinimaxPlayer

MinimaxPlayer je hlavným cieľom tejto práce. Skúsime ho navrhnúť tak, aby dokázal porážať SmartPlayera. Ako názov naznačuje, MinimaxPlayer je založený na prehľadávaní stavov. Kompromis medzi hĺbkou a šírkou prehľadávania zo sekcie 3.8 sa rieši tak, že zlomok časového limitu sa len generujú možné rozdania a zvyšný čas paralelne beží iteratívne prehlbovanie na týchto rozdaniach. Ak sa nestihnú vygenerovať všetky rozdania, tak prehľadávanie beží na podmnožine, ako bolo spomenuté v sekcii 3.7.

Nakoľko je veľmi zložité heuristicky odhadnúť zisk v ľubovoľnom stave, orezáva sa len v stavoch, kde je kopa prázdna, a teda majú všetci rovnako veľa kariet na ruke. V strome je to každá tretia úroveň. Pri iteratívnom prehlbovaní táto skutočnosť veľmi neprekáža, ale pri zoraďovaní zo sekcie 3.4 je zvýšenie efektivity alfabeta orezávania menšia, ako keby sa stavy zoraďovali na každej úrovni. Aktuálne sa zoraďuje len v stavoch, kde je na ťahu hráč, ktorý dáva na kopu tretiu kartu. Ten má viac obmedzený výber ako začínajúci hráč, preto je vetviaci faktor nižší. Zoraďovanie môže mať účinok len vtedy, ak má na výber aspoň dve karty.

Pri voľbe talóna a licitácii sa stále používajú metódy, ktoré používa SmartPlayer. Prehľadávanie by bolo problematické, lebo kým nemáme žiadnu informáciu o rozdelení kariet súperov, počet možných rozdaní je príliš veľký. Do budúcnosti preto navrhujeme pre tieto situácie zvážiť niektoré z metód strojového učenia.

Pri voľbe tromfa je situácia iná.

4.2.6 TromfChooser

TromfChooser je trieda, ktorú používa MinimaxPlayer pri voľbe tromfa. Úlohou je vybrať tromf zo 7 kariet. Následne sa zvyšných 25 kariet rozdelí medzi hráčov v pomere 5:10:10. To je $\binom{25}{5}\binom{20}{10} = 9816086280$ možností. Pre každú z týchto možností prehľadávať celý stavový priestor nie je zatiaľ v silách bežného počítača. Ani tak skoro nebude.

Väčšinou sa bude ale situácia opakovať. Počet možností, ako rozdať 7 kariet na voľbu tromfa je $\binom{32}{7} = 3365856$. To je už rozumnejšie číslo. Pre každé rozdanie si môžeme predpočítať, ktorú kartu sa oplatí zvoliť ako tromf. Toto číslo sa dá ešte zmenšiť. Keď napríklad zvolíme zeleň ako tromf, dopadne to rovnako, ako keby sme vymenili zelené a guľové karty a zvolili ako tromf guľu. Vďaka tomuto invariantu stačí mať v tabuľke len rozdania, pri ktorých guľové karty > žaludné karty > zelené karty pri nejakom usporiadaní >. Pre červeň to ale neplatí, lebo keď je tromf červený, všetky zisky sú dvojnásobné. Tým sa nám počet rozdaní zredukoval na 592 720.

4.2.7 Precompute

Precompute je projekt, ktorý vypĺňa tabuľku na voľbu tromfa. Pre každé rozdanie 7 kariet použitím heuristík vylúči farby, ktoré sa určite neoplatia zvoliť ako tromf. Tieto heuristiky nie sú veľmi prísne, vylúčia len také farby, ktorých ohodnotenie bolo nižšie ako polovica ohodnotenia najsilnejšej farby. Ak ostane viac ako jedna farba, spustí simuláciu hrania 3 000 hier pre každého kandidáta na tromf. Do tabuľky vyplní taký tromf, pri ktorom sa dosiahol najvyšší priemerný zisk. Táto tabuľka predstavuje analógiu knižnice otvorení,

ktorá sa používa v mnohých iných hrách, napríklad v šachu.

Predpočítavanie bežalo na štyroch počítačoch jeden mesiac.

Kapitola 5

Testovanie hráčov

Štatistiky môžu overiť efektívnosť implementácie. Pretože pri mariáši je rozhodujúce hlavne to, aké karty dostane hráč, je potrebné odohrať veľa hier, aby boli výsledky relevantné. Ak chceme porovnávať hráčov objektívne, dobrý postup je vymeniť hráčov a rozdávať im tie isté karty. OpenMarias podporuje testovanie hráčov, pričom miešanie kariet môže byť úplne náhodné alebo môže byť generované pseudonáhodnou postupnosťou s pevným základom. Tým sa dajú zopakovať rovnaké podmienky, keď sa stanoví rovnaký základ postupnosti.

5.1 Prvé testovanie

Prvé testovanie prebehlo v januári 2010. Zúčastnili sa ho hráči RandomPlayer, MinPlayer, MaxPlayer a SmartPlayer.

5.1.1 Postup testovania

Každá možná trojica odohrala 10 000 hier. Pritom sa mohli stretnúť aj hráči rovnakého typu. Forhont sa posúva po každej hre doľava. Pre každý typ hráča sa sčítajú zisky, ktoré získali jeho inštancie.

5.1.2 Výsledky testovania

Výsledky testovania sú v tabuľke 5.1. V treťom stĺpci je priemerný zisk z ôsmich meraní.

1.	SmartPlayer	8163
2.	MaxPlayer	-1277
3.	MinPlayer	-2944
4.	RandomPlayer	-3942

Tabuľka 5.1: Výsledky testovania hráčov.

Výsledky dopadli podľa očakávania. SmartPlayer jednoznačne víťazil v každom meraní. Tiež sa ukázalo, že ľubovoľná taktika je lepšia, ako dávať náhodne karty. MinPlayer aj MaxPlayer dopadli lepšie ako RandomPlayer.

5.2 Druhé testovanie

Druhé testovanie prebehlo v apríli 2011. Zúčastnili sa ho SmartPlayer, MinimaxPlayer, TrustingMinimaxPlayer. Medzi prvým a druhým testovaním prešiel SmartPlayer mnohými zmenami, preto bude klásť väčší odpor.

5.2.1 Postup testovania

Ako prvé treba určiť počet hier, ktoré treba odohrať, aby sme dosiahli postačujúcu štatistickú vzorku. Na to použijeme nasledovnú metódu:

Na stoličkách A, B, C budú traja hráči toho istého typu. Keďže hrajú rovnako dobre, pri zvyšovaní počtu hier by mal priemerný zisk za jednu hru konvergovať k nule. Budeme teda skúmať závislosť priemerného zisku za jednu hru hráča od počtu hier. Priemerný zisk sa vypočíta ako podiel maximálneho zisku jedného hráča a počtu odohraných hier. Keďže mariáš je hra s nulovým súčtom, táto hodnota je vždy kladná, čo nám umožňuje znázorniť výsledky v logaritmickej škále.

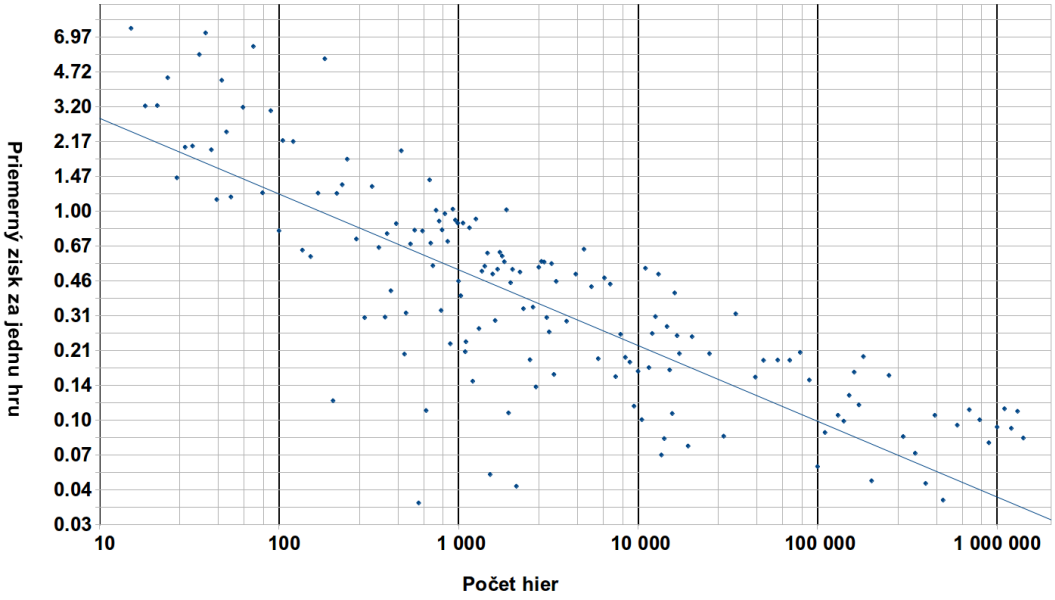
Druhý test by mal už overiť schopnosť MinimaxPlayera ryžovať na dvoch SmartPlayeroch. Ak predpokladáme, že v priemere je MinimaxPlayer lepší, tak pre vyšší počet hier by mal priemerný zisk za jednu hru konvergovať ku konkrétnemu číslu.

Tretí test bude porovnávať schopnosť spolupráce. Nastúpia v ňom dvaja MinimaxPlayeri proti SmartPlayerovi. Budeme skúmať priemerný súčet ich ziskov. Nemá zmysel posúvať rolu forhonta, lebo chceme testovať schopnosť spolupráce a nie prípady, keď hrajú proti sebe. Napriek tomu prebehli testy aj pre posúvanie forhonta. Priemerný zisk MinimaxPlayera bol približne o 30% nižší ako v druhom teste.

5.2.2 Výsledky testovania

Výsledky prvého testu sú zobrazené v grafe 5.1, ktorý má obidve osi v logaritmickej škále. Ako referenčného hráča sme určili SmartPlayera, lebo dokáže

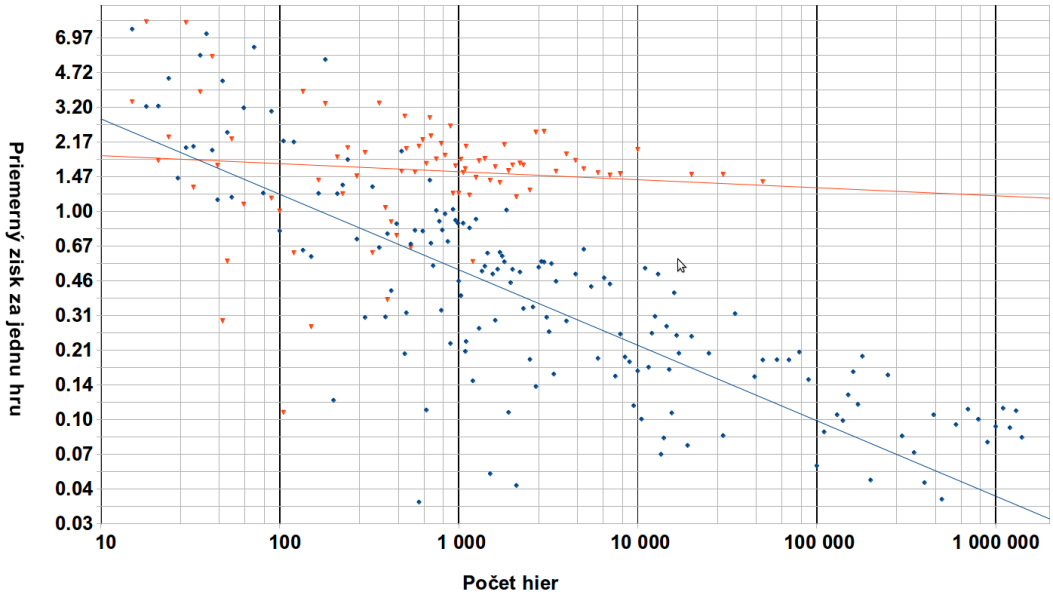
veľmi rýchlo odohrať veľa hier. Hodnoty jednotlivých meraní sú zaznačené modrou farbou. Pre daný počet hier vyznačujú očakávanú oblasť hodnôt, pre ktoré je testovaný hráč na rovnakej úrovni ako SmartPlayer. Očakávaný rozptyl zavisí od počtu hier p vzťahom $6.54p^{-0.37}$.



Obr. 5.1: Priemerný zisk za jednu hru troch rovnakých hráčov

V grafe 5.2 sú červenou farbou sú znázornené zisky MinimaxPlayera, ktorý hral proti dvom SmartPlayerom, pričom sa rola forhonta neposúvala. Záporné výsledky, ktoré sa nedajú zobraziť, nastali len trikrát, a to len pri počte odohraných hier menšom ako 200. Väčšinou boli kladné, čo sa potom odrazilo na priemernom zisku pri vyššom počte hier. Z grafu sa dá vyčítať, že ďalšie testy na porovnanie hráčov stačí robiť pre počty hier väčšie ako 1000.

Priemerný zisk MinimaxPlayera za jednu hru konverguje k číslu 1.5. Pre porovnanie - najmenší možný kladný zisk forhonta je 2, ak nikto nedá flek na hru.

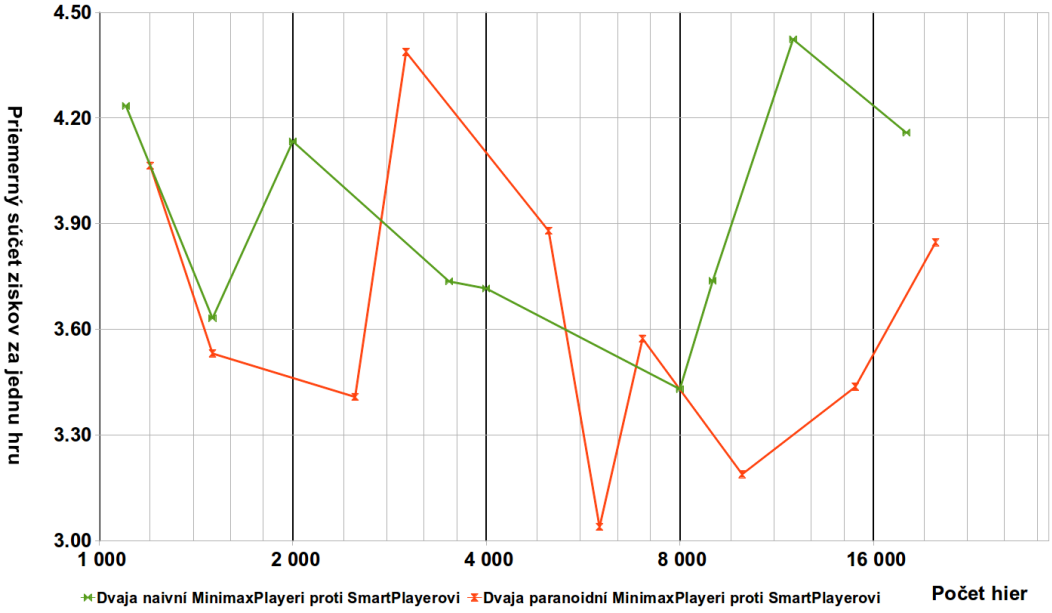


Obr. 5.2: Priemerný zisk za jednu hru MinimaxPlayera proti dvom SmartPlayerom

V grafe 5.3 sú červenou farbou znázornené zisky dvoch MinimaxPlayerov proti jednému SmartPlayerovi, ktorý bol stále na pozícii forhonta. Zelenou farbou sú znázornené zisky dvoch naivných MinimaxPlayerov[1]. Očakávalo sa, že dvaja hráči, keď si dôverujú, budú mať vyšší zisk, ako dvaja paranoidní hráči. Je to pravda iba čiastočne, lebo slepá dôvera tiež nie je správna cesta k najvyššiemu zisku. Mariáš je hra s nesymetrickou informáciou. Hráči, aj keď spolupracujú, majú k dispozícii rôzne informácie a preto pri metóde Monte Carlo budú prehľadávať rôzne stromy.

[PNS10] uvádza porovnanie medzi paranoidným a sebavedomým hráčom, ktorý predpokladá, že súper hrá náhodne. Vo všetkých hrách s neúplnou informáciou bol lepší sebavedomý hráč. Rozdiel sa zvyšoval pri hrách z nižšou informovanosťou hráčov. Slepá dôvera je niečo iné ako sebavedomie. Napriek tomu stojí za pozornosť zvážiť aj tento model pri budúcom vývoji.

[1]Naivný MinimaxPlayer je kópia MinimaxPlayera, len slepo dôveruje svojmu spojencovi a predpokladá, že vyberie ťah, ktorý maximalizuje spoločný zisk.



Obr. 5.3: Priemerný súčet ziskov dvoch MinimaxPlayerov proti SmartPlayerovi

Záver

V práci sme čitateľa oboznámili s pravidlami mariášu. Takisto sme zhrnuli známe techniky používané pri tvorbe počítačových hráčov. Hráč, ktorý prehľadáva stavy sa nespolieha na rôzne finty, ktoré vymysleli ľudskí hráči mariášu, preto vie "predvídať", čo sa stane niekoľko ťahov dopredu, čo mu dáva istú výhodu.

Free OpenSource implementácia hry umožňuje komukoľvek sa podieľať na ďalšom vývoji. Prostredie pre vývoj hráčov poskytuje jednoduchú implementáciu vlastného hráča a následné porovnávanie s už existujúcimi hráčmi. Výsledky testovania potvrdili úspech vyvíjaného algoritmu založeného na prehľadávaní stavov.

Algoritmus sa dá ďalej vyvíjať. Napríklad lepšie nastaviť kompromis medzi šírkou a hĺbkou prehľadávania, skúsiť kompromis medzi paranoidným a naivným minimaxom. Dajú sa aj zahrnúť techniky ako Trappy minimax ([GR07]).

[RN03] navrhuje aj úplne iný prístup - neprehľadávať stavy pre možné rozdania, ale prehľadávať stavy, ktoré zodpovedajú poznatkom ostatných hráčov o možných rozdaniach. Pre každé rozdanie a každého hráča by sme si potrebovali pamätať množinu rozdaní, o ktorých vie, že môžu byť aktuálne.

Stavový priestor by sa tak enormne zväčšil.

Úspech pri porovnávaní s už existujúcimi počítačovými hráčmi avšak nezaručuje vyšší úspech aj proti ľudským hráčom. Preto do budúcnosti navrhujem verziu, ktorá po každej hre by odoslala výsledky na server, kde by sa spracovávali. Tak by sa dala otestovať úspešnosť proti ľudským hráčom.

Literatúra

[Lip08] Marek Lipták. "Mariáš pro čtyři hráče". Česká technická univerzita. Bakalárska práca. 2008. https://dip.felk.cvut.cz/browse/details.php?f=F3&d=K13136&y=2008&a=liptam1&t=bach

[Kub06] Miroslav Kubík. "Mariáš". Česká technická univerzita. Bakalárska práca. 2006. https://dip.felk.cvut.cz/browse/details.php?f=F3&d=K13136&y=2006&a=kubikm1&t=bach

[St04] Martin Staufčík. "Mariáš 2.7". 2004. http://marias.webz.cz/

[Fia09] Jan Fiala. "Becherovka Mariáš". 2009. http://www.falasoft.com/misc/cs/cards

[TS01] Theodore L. Turocy, Bernhard von Stengel. "Game Theory". CDAM Research Report LSE-CDAM-2001-09. 2001. http://www.cdam.lse.ac.uk/Reports/Abstracts/cdam-2001-09.html

[Fong05] Jason Fong. "Developing an Artificial Intelligence for Whist". University of California, Los Angeles. 2005 http://www.fongboy.com/programs/Whist/whist_paper.pdf

[FBM98] Ian Frank, David Basin, Hitoshi Matsubara. "Finding Optimal Strategies for Imperfect Information Games". 1998. http://portal.acm.org/citation.cfm?id=295724

[RN03] Stuart J. Russel, Peter Norvig. "Artificial Intelligence, A Modern Approach - Second Edition". ISBN: 0-13-080302-2. Prentice Hall Series. Pearson Education Inc. 2003.

[HN09] Thomas Dueholm Hansen, Helene Damgaard Nielsen. "Hardness of minimax computation in 3-player games". Computational game theory. University of Aarhus. 2009. http://www.daimi.au.dk/~bromille/CGT09/lecture8.pdf

[Smith06] Michael Smith. "Running the Table: An AI for Computer Billiards". University of Alberta. 2006. http://portal.acm.org/citation.cfm?id=1597695

[CG02] Tal Cohen, Joseph Gil. "Better Construction with Factories". Journal of Object Technology. Israel Institution of Technology. 2002. http://www.jot.fm/issues/issue_2007_07/article3/

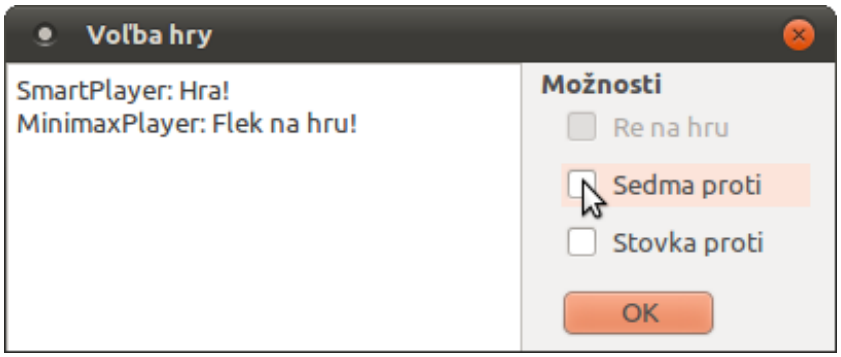
[Ras07] Eric Rasmusen. "Games and information: an introduction to game theory". ISBN-10: 1-4051-3666-9. 2007. http://www.amazon.com/Games-Information-Introduction-Game-Theory/dp/1405136669

[GR07] V. Scott Gordon, Ahmed Reda. "Trappy Minimax- using Iterative Deepening to Identify and Set Traps in Two-Player Games". CSU, Sacramento. ISBN: 1-4244-0464-9 . 2006. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.87.3383

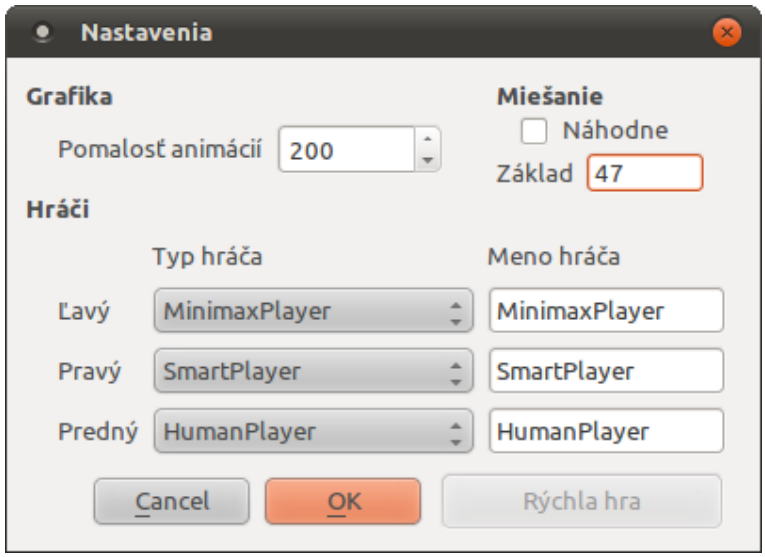
[PNS10] A. Parker, D. S. Nau, and V. S. Subrahmanian. "Paranoia versus overconfidence in imperfect-information games". University of Maryland. 20010. `http://www.cs.umd.edu/~nau/papers/parker10paranoia.pdf`

Screenshoty

Obr. 4: Screenshot: Intro



Obr. 5: Screenshot: Licitovanie



Obr. 6: Screenshot: Nastavenia

Obr. 7: Screenshot: Ukážka hry

Register