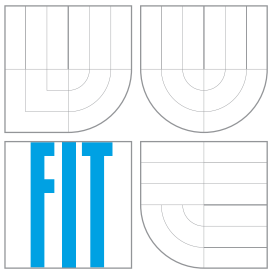
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

HRA VYUŽÍVAJÍCÍ METODU MTD(F)

COMPUTER GAME BASED ON MTD(F) METHOD

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MATEJ JANÁČEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ TECHET
SUPERVISOR

BRNO 2007

Zadání

- Seznamte se se základy umělé inteligence, především pak s metodami pro hraní her (alphabeta).
- Seznamte se s metodou MTD(f).
- Zamyslete se nad hrou, která by vhodně demonstrovala výše uvedenou metodu.
- Implementujte tuto hru.
- Diskutujte přednosti a nedostatky vašeho řešení a možné další rozšíření vámi navrženého systému.

Licenční smlouva

Abstrakt

Táto bakalárska práca demonštruje výhody a nevýhody metódy MTD(f) na jednoduchej implementácii hry Dáma. Stručne popisuje rozdiely medzi touto a inými metódami používanými na vyhľadávanie najlepšieho ťahu v hrách.

Klíčová slova

Dáma, Alfa-Beta, Alfa, Beta, Umelá Inteligence, Umelá Inteligencia

Abstract

This bachelor's thesis demonstrates pros and cons of MTD(f) method on simple implementation of checkers game. Briefly describes differences between this and other methods used for the best move search in games.

Keywords

MTD(f), Checkers, Draughts, Alpha-Beta, Alpha, Beta, Artificial Intelligence

Citace

Matej Janáček: Hra využívající metodu MTD(f), bakalářská práce, Brno, FIT VUT v Brně, 2007

Hra využívající metodu MTD(f)

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jiří Techeta

.....

Matej Janáček

15. května 2007

Obsah

Seznam obrázků

Kapitola 1

Úvod

Už oddávna sú ľudia fascinovaný svojou schopnostou rozmýšlat. Schopnosťou reagovať na prejavy a podnety prostredia, analyzovať ich a využívať ich vo svoj prospech. Schopnosťou, ktorá ako niektorí tvrdia ich odlišuje od nižších živočíšnych foriem. Možno to nie je úplne najvhodnejšia definícia, ale je pravdou, že u človeka sa táto schopnosť v rámci živočíšnej ríše rozvinula najviac.

Postupom času sa schopnost myslieť, inak nazývaná aj inteligencia stala zdrojom štúdia mnohých filozofov staroveku, stredoveku a aj novoveku. Neskôr po ére industrializácie a rozmachu strojov v každodennom živote človeka, sa postupne začala vynárať otázka, či je možné neživému stroju dat schopnosť mysliet tak ako je to u človeka. Dnes túto otázku skúma vedná disciplína nazývaná príhodne: Umelá Inteligencia.

Zámerom vytvorenia strojov schopných samostatného myslenia je tak ako pri všetkých vynálezoch uľahčenie alebo nahradenie ľudskej práce. Vytvorenie dokonalej kópie ľudskej inteligencie ako prvotný a hlavný problém umelej inteligencie sa však čoskoro ukázal ako príliš komplexný a v rámci blízkeho časového horizontu, aj neriešitelný. Preto sa tento problém postupne rozdelil na množstvo menších, užšie špecifikovaných problémov, ako napríklad:

- rozpoznávanie reči
- rozpoznávanie obrazu
- triedenie vzoriek
- hranie hier
- navigácia v známom a neznámom teréne
- riadenie robotov
- riadenie a plánovanie napr. výrobných procesov
- analýza a predpovedanie štatistických postupností (napr. obchodných , burzových informácií)

- spracovanie a analýza dát z prostredia (geológia, meteorológia, ...)
- reprezentácia zozbieraných dát
- data mining

Výskum v oblasti umelej inteligencie však stále napreduje a spolu so vznikom modernejších strojov sa aj jej význam stále zvyšuje. Kórejské ministerstvo informácií a komunikácií dokonca prednedávnom (marec 2007) zverejnilo správu,ktorá predpovedá, že roboty budú bežne vykonávat operáciu do roku 2018. [5])

Kapitola 2

Umelá Inteligencia v hrách

Hry boli vždy jedným z hlavných stavebných kameňov vo výskume umelej inteligencie.

Veľkú časť výskumu v tejto oblasti zohrala hra Šach, ktorá významne pomohla k pokroku v tejto oblasti. V súčasnosti sú najlepšie šachové programy schopné hrať vyrovnané partie s tými najlepšími ľudskými protivníkmi. Ako príklad je možné uviesť priekopnícky šachový počítacový systém od firmy IBM ,,Deep Blue", prípadne programy Shredder 8, alebo Deep Fritz 8. Existuje však aj rada amatérskych šachových programov schopných poraziť profesionálnych hráčov šachu.

Väčšina týchto programov je založená na tzv. Hrubej sile. To znamená, že algoritmus pri hľadaní najvýhodnejšieho ťahu prehľadáva čo najväčšiu časť stavového priestoru.

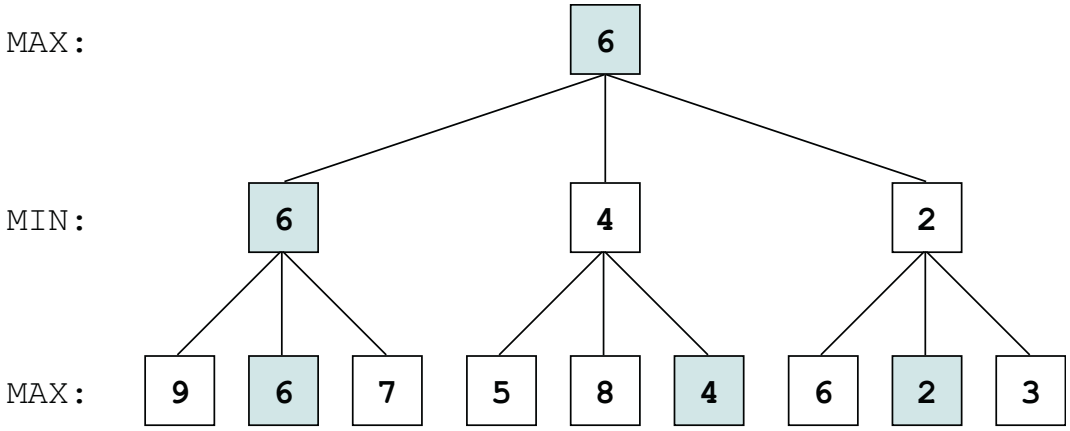
2.1 MinMax algoritmus

Medzi najznámejšie algoritmy na prehľadávanie stavového priestoru v hrách patria určite algoritmy MinMax a jeho vylepšenie Alfa-Beta.

Algoritmus MinMax je metóda vyhľadania toho najlepšieho herného ťahu v hrách pre dvoch hráčov, z ktorých každý smeruje k opačnému cieľu, teda k vlastnému víťazstvu. Metóda je založená na jednoduchej myšlienke, že každý z protihráčov si vo svojom ťahu vyberie to najlepšie možné riešenie.

Jednotlivé ťahy sú reprezentované tzv. herným stromom. Tento herný strom prehľadávaný v algoritme MinMax je využívaný aj v ostatných algoritmoch popisovaných v tejto práci. Predstavuje všetky možné ťahy hry vychádzajúce zo súčasnej pozície do určitej, vopred určenej budúcnosti, teda hĺbky stromu. Jednotlivé úrovne stromu predstavujú striedavo možné ťahy jednotlivých hráčov. Hodnoty stromu v jeho listoch predstavujú vyhodnotenie stavu hry po každej sekvencii herných pohybov. V prípade obmedzenej hĺbky vyhľadávania predpovedajú tieto výsledky aj možné smerovanie hry, ale vždy platí, že prehladávanie do väčšej hĺbky vráti presnejšiu predpoved najlepšieho možného výsledku, nakoľko je schopné odhaliť prípadné herné pasce, alebo iné nepredpokladané situácie.

Po získaní hodnoty stavu hry z listu prehľadávaného stromu sa algoritmus vracia



Obrázek 2.1: Vyhľadanie najlepšej hodnoty algoritmom MinMax

po strome a po vyhodnotení všetkých potomkov určitého uzlu, určí jeho hodnotu. Hodnoty v skutočnosti predstavujú veľkosť výhody pre jedného, resp. druhého hráča v danej hernej situácii. Napríklad v mojej implementácii predstavujú nižšie hodnoty väčšiu výhodu pre bieleho hráča a vyššie hodnoty naopak výhodu pre čierneho hráča.

```
function minimax(node, depth)
    if node is a terminal node or depth = 0
        return the heuristic value of node
    if the adversary is to play at node
        let A := +infinity
        foreach child of node
            A := min(A, minimax(child, depth-1))
    else {we are to play at node}
        let A := -infinity
        foreach child of node
            A := max(A, minimax(child, depth-1))
    return A
```

Algoritmus MinMax zapísaný v pseudokóde. [4]

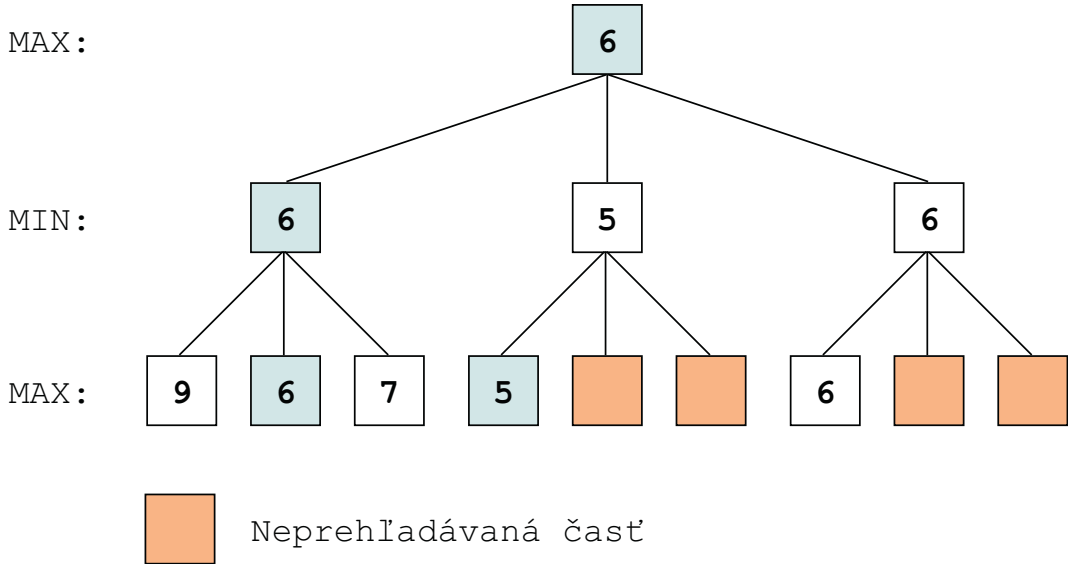
2.2 Alfa-beta agoritmus

Algoritmus alfa-beta je účinná modifikácia algoritmu MinMax. Veľkosť práce algoritmu MinMax sa pri zväčšujúcej hĺbke vyhľadávania zvyšuje exponenciálne a prehľadávajú sa aj možné vetvenia hry, pri ktorých je jasné, že sú nevýhodnejšie ako už prehľadané varianty. Algoritmus alfa-beta tento problém odstraňuje tým, že sa tieto spomínané vetvenia stromu neprehľadávajú. Hráč nespraví na prvý pohľad dobrý ťah, ktorý by ale umožnil jeho protihráčovi zahrať lepší ťah ako sú doterajšie výsledky vyhľadávania v strome a tak ak

čiastočné výsledky prehľadávania v potomkoch smerujú k tomuto stavu, tak prehľadávanie ďalších potomkov už nebude prebiehať, nakoľko sa predpokladá že hráč radšej vyberie predchádzajúci výsledok, ktorý mu dával väčšiu výhodu.

Týmto sa relatívne urýchli prehľadávanie vzhľadom na algoritmus MinMax, aj ked veľkosť ušetrenej práce výrazne závisí na jednotlivých hodnotách listov stromu. V najhoršom možnom prípade musí algoritmus vyhodnotiť rovnaké množstvo prvkov stromu ako v prípade algoritmu MinMax.

Algoritmus si pri pohybe stromom prenáša informácie o najlepšom ťahu pre obidvoch hráčov. Tieto dve hodnoty boli nazvané alfa a beta a z toho vlastne vyplýva aj názov pre tento algoritmus.



Obrázek 2.2: Vyhľadanie najlepšej hodnoty algoritmom Alfa-Beta

2.3 MTD(f) modifikácia Alfa-Beta algoritmu

MTD(f) je považovaný v súčasnosti za jednu z najefektívnejších modifikácií algoritmu alfa-beta popísaného vyššie pri hraní programov ako šach, dáma alebo piškvorky (othello). V testoch na turnajových hrách vykazuje lepšie výsledky ako iná, veľmi rozšírená modifikácia algoritmu alfa-beta nazývaná NegaScout.

Skutočný algoritmus je pritom veľmi jednoduchý a v pesudokóde predstavuje len pár riadkov.[2]

```
int MTDF(root : node_type; f : integer; d : integer) {
    g := f;
    upperBound := +INFINITY;
    lowerBound := -INFINITY;
```

```
    repeat
          if (g == lowerBound) then
            beta := g + 1 else beta := g;
          g := AlphaBeta(root, beta - 1, beta, d);
          if (g < beta) then
            upperBound := g
          else lowerBound := g;
    until (lowerBound >= upperBound);
    return g;
}
```

Program pracuje s funkciou AlphaBeta ktorá predstavuje algoritmus Alfa-Beta volaný s parametrami beta - 1 a beta, teda s nulovým rozsahom prehľadávania. Každé toto volanie vracia jednu z krajných hodnôt, ktorá sa následne uloží do premenných lowerBound prípadne upperBound, ohraničujúcich skutočnú hľadanú hodnotu stromu.

Plus a mínus INFINITY, teda nekonečno, predstavujú vlastne medze ohraničujúce hodnoty listov stromu. Hľadaná hodnota algoritmu sa vráti, keď sa hodnoty lowerBound a upperBound prekryjú.

Efektivita metódy MTD(f) spočíva práve vo viacnásobnom volaní algoritmu Alfa-Beta s nulovým rozsahom hodnôt, ktorý je príčinou toho že algoritmus síce bude neúspešný vo vyhľadávaní presného výsledku ako by to bolo pri bežnom volaní s rozsahom od mínus do plus nekonečno, ale každé volanie vráti buď horné alebo dolné ohraničenie pre hľadanú hodnotu. Algoritmus teda síce vracia menej informácií ale na druhú stranu maximálne využije vynechávanie neefektívnych variant Alfa-Beta vyhladávania.

Algoritmus ale pre efektivitu potrebuje dobrý ,,odhad" konečnej hľadanej hodnoty. Čím lepší bude tento odhad, tým menejkrát prebehne cyklus volajúci AlphaBeta funkciu a tým efektívnejší teda algoritmus bude. Funkcia AlphaBeta sa však zavolá vždy aspoň dvakrát, tj. v prípade, že by sme odhadom presne trafili hľadanú hodnotu stromu, algoritmus prebehne raz aby zistil dolné a raz aby zistil horné ohraničenie.

Kapitola 3

Výber a Koncepcia hry

Hier, ktoré by vhodne demonštrovali algoritmus MTD(f) je veľmi veľa. Určite najznámejšia z nich je šach, ktorý bol a stále je hlavnou inšpiráciou pri vývoji hernej umelej inteligencie. Napriek tomu som vo svojej práci dal prednosť hre Dáma (v anglickom origináli ,,Checkers" alebo ,,Draughts"), ktorá je z programátorského hľadiska jednoduchšia na implementáciu, a na druhej strane prednosti algoritmu MTD(f) sú rovnako dobre demonštrovatelné aj na tomto príklade.

3.1 Dáma a jej história

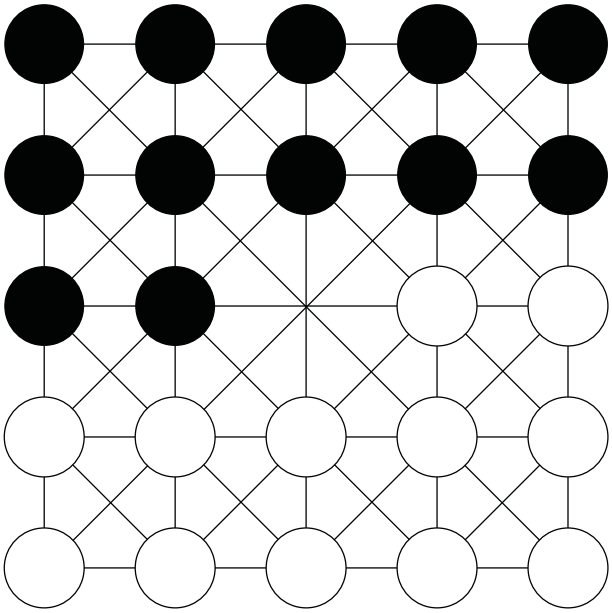
Dáma (v anglickom origináli ,,Checkers" alebo ,,Draughts") má svoje korene hlboko v minulosti.

Predpokladá sa, že prvé formy Dámy sa hrali už v roku 3000 pred naším letopočtom. Dokazujú to archeologické nálezy z vykopávok v meste Ur v dnešnom Iraku. Táto staroveká verzia dámy sa však hrala na odlišnej hernej ploche a s odlišným počtom kameňov a samozrejme len na základe vykopávok nie je ani možné určit presné pravidlá tejto starej verzie.

V starovekom Egypte existovala ďalšia podobná hra, známa ako Alquerque, ktorá sa hrala na hernej ploche o rozmeroch 5 krát 5 políčok. Jej korene siahajú do roku 1400 pred naším letopočtom. Táto hra bola vo svojej dobe veľmi obľúbená a rozšírila sa po celom vtedy známom ,,západnom" svete.

Okolo roku 1100 nášho letopočtu sa vo Francúzsku ujala myšlienka hrať túto hru na vtedajších šachovniciach, čím sa hracia plocha značne rozšírila. Táto verzia sa nazývala ,,Fierges" alebo ,,Ferses". Čoskoro niekto prišiel aj s myšlienkou spraviť skoky pre jednotlivé kamene (hracie figúrky) povinné a tak vznikla oveľa zaujímavejšia hra, podobná tej dnešnej a známa ako ,,Jeu Force".

Staršie verzie Jeu Force boli považované viac za spoločenskú hru pre ženy ako pre mužov a preto napríklad vo Francúzsku ľudia túto hru nazývali ,,Le Jeu Plaisant De Dames" a v Španielsku ,,Damas". Na základe týchto názvov pravdepodobne vzniklo aj naše pome-



Obrázek 3.1: Herná plocha hry Alquerque

novanie ,,Dáma".

Pravidlá Jeu Force sa čoskoro rozšírili do okolitých krajín. V pätnástom a šestnástom storočí vznikli v Španielsku knihy popisujúce túto hru a v roku 1756 v Anglicku, kde nazývali túto hru ,,Draughts", o nej napísal rozpravu matematik William Payne. Hra si stále udržiavala svoju popularitu a to natoľko, že už v roku 1847 sa konali prvé majstrovstvá sveta.

Postupom času, tak ako sa zisťovalo, že určité rozohranie dávalo jednej strane výhodu, tak sa aj modifikovali pravidlá pre expertov. V dnešnej dobe sa používajú tri obmedzenia pre turnajovú verziu dámy.

Dáma sa prvýkrát dostala do povedomia programátorskej verejnosti ešte pred Druhou svetovou vojnou. Hoci počítače boli v tej dobe ešte len v štádiu počiatočného vývoja, známy počítačový priekopník, matematik, logik a kryptograf Alan Turing vytvoril jednoduchý program pre hranie dámy, ktorý však vyžadoval robiť všetky výpočty na papier, pretože samotné počítače neboli vtedy ešte vhodné na vykonávanie takýchto činností.

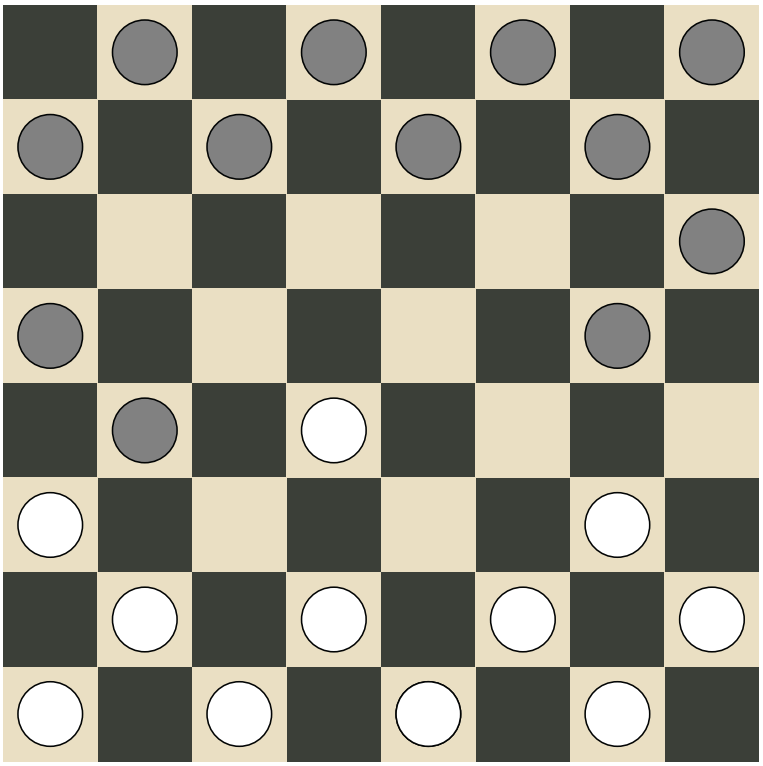
Prvý, kompletne počítačový program pre hranie dámy vytvoril v roku 1952 Arthur L. Samuel.

A ako roky ubiehali, tento program bol neustále vylepšovaný tak ako sa vylepšovala aj počítačová kapacitná a rýchlostná technika. V súčasnosti sú dobré programy na hranie dámy obohatené o databázu všetkých možných kombinácií ťahov pri zostávajúcich ôsmich až desiatich hracích kameňoch, a predpokladá sa, že sa toto číslo bude stále zväčšovať. To znamená, že samotná stratégia programu má stále menší a menší význam na úkor prehľadávania databázy možných ťahov.

A takéto programy už nemajú problém porážať alebo prinajmenšom aspoň remizovať aj s tými najlepšími hráčmi na svete.[3]

3.1.1 Základné pravidlá

Dáma je hra pre dvoch hráčov, z ktorých každý má svoju sadu hracích kameňov. Jeden hráč hraje s tmavými hracími kameňmi a druhý hráč má naopak svetlú sadu. Hráč s tmavými hracími kameňmi začína hru, pokiaľ sa neuvádza inak (V mojej implementácii je naopak začínajúcim hráčom hráč so svetlou sadou kameňov). Hra prebieha na šachovnicovej ploche, so striedavo tmavými a svetlými štvorcami, definujúcimi jednotlivé herné pozície.



Obrázek 3.2: Typická hracia plocha pre Dámu

Hracie kamene sa môžu pohybovať diagonálne na okolné voľné polia, prípadne s nimi hráč môže zajať súperov hrací kameň tým, že ho ,,preskočí". V niektorých verziách dámy je ,,preskakovanie" súperových hracích kameňov povinné (ak sa takýto ťah ponúka) a v niektorých naopak nie. Z defínicie pohybu vyplýva, že hra prebieha len na jednej farbe štvorcov. Ktorej, to záleží od konkrétnej verzie hry.

Naopak vo všetkých pravidlách ale platí, že hráč prehral, ak už nemá žiadne hracie kamene, prípadne nemá žiadnu možnosť pohybu so svojimi súčasnými kameňmi.

V dáme existujú dva typy hracích kameňov. Prvý typ je jednoduchý hrací kameň, nazývaný ,,Pešiak". Každý hráč začína svoju hru len so sadou pešiakov. Pešiak môže

používať ťahy popísané v predchádzajúcom odstavci, t.j. pohyb diagonálne o jeden krok na voľné políčko. Opäť záleží na verzii dámy, či môže byť tento pohyb len v smere útoku alebo aj opačným smerom. Okrem toho môže pešiak aj zajať súperove hracie kamene ich preskakovaním. Pešiak môže zajať viacero súperových kameňov, vykonaním viacerých nadväzujúcich skokov v jednom ťahu.

Druhým typom hracích kameňov je tzv. ,,Král". Taktiež sa tento typ nazýva ,,Kráľovná" prípadne ,,Dáma". Pre jednoduchosť budem používať označenie dáma, ktoré je rozšírené u nás. Dáma vznikne tým, že sa niektorý z pešiakov dostane na posledný rad (z jeho pohľadu) hernej plochy a tým povýši na dámu. Dáma je vždy určitým spôsobom odlíšená (v stolnej verzii sa dámy väčšinou vytvárajú položením dvoch pešiakov na seba) od pešiakov a má aj špeciálne vylepšené schopnosti pohybu, ako pohyb dozadu, prípadne krok ľubovoľnej dĺžky.

3.1.2 Verzie dámy

V súčasnosti existuje veľa verzií dámy a tu sú popísané tie najznámejšie z nich.

Medzinárodná verzia - International draughts

Medzinárodná, alebo tiež Poľská verzia je najrozšírenejšia v určitých častiach Európy a Afriky.

Hraje sa na hernej ploche o rozmeroch 10 krát 10 štvorcov a každý hráč začína hru so sadou 20–tich pešiakov. Hráč musí vždy pohybovať tým hracím kameňom, ktorý mu umožní zajať čo najviac súperových hracích kameňov. Dámy majú schopnosť pohybovať sa a prípadne skákať akokoľvek dlhým skokom dopredu aj dozadu.

Anglická verzia - English draughts - Checkers

Ako názov napovedá, táto verzia sa hraje hlavne vo Veľkej Británii a taktiež je rozšírená na americkom kontinente.

Hracia plocha má rozmery 8 krát 8 štvorcov a hráči disponujú na začiatku sadou 12–tich pešiakov. Pešiaci sa môžu pohybovať len dopredu, dáma sa môže pohybovať všetkými smermi, ale dĺžka jej pohybu a skokov je rovnaká ako u pešiakov. Pri možnosti viacerých skokov nemusí hráč nutne vybrať ten ťah, ktorý vyradí najviac súperových herných kameňov, ale môže si vybrať ľubovoľný z nich.

Ďalšie verzie

Existuje ešte veľké množstvo ďalších väčšinou národných verzií, ktoré sa od horeuvedených dvoch líšia, buď veľkosťou hernej plochy a počtom hracích kameňov, alebo modifikáciami pravidiel o pohybe a skokoch pešiakov a dám.

3.1.3 Implementovaná verzia

Verzia dámy, ktorá som implementoval vychádza z Anglickej verzie, ale obsahuje dve menšie a jednu väčšiu modifikáciu. Hracia plocha má velkost 8 krát 8 štvorcov, ale farby štvorcov sú opačné oproti verzii anglickej, teda hracie kamene sa pohybujú po svetlých štvorcoch. Pre počítačové zobrazenie bola verzia s pohybom po tmavých štvorcoch trochu nevhodná na zobrazenie. Ďalšia zmena je tiež v podstate estetická. Hráč ktorý začína má nie čierne, ale biele kamene. Z pohľadu užívateľa sú teda biele kamene na začiatku v dolnej časti hernej plochy a hraje s nimi užívateľ. Počítač hraje s čiernymi hracími kameňmi, ktoré začínajú v hornej časti hernej plochy. Táto zmena nemá logické zdôvodnenie, vznikla na základe mojich osobných skúseností s dámou ešte predtým ako som sa oboznámil s oficiálnymi pravidlami. Posledná a najvýraznejšia zmena je zrušenie viacnásobných skokov. Zostávajú tak len jednoduché skoky ponad jeden nepriateľský herný kameň. Toto obmedzenie vyplynulo pri tvorbe hry a malo poslúžiť na zjednodušenie implementácie vyhľadávania najlepšieho ťahu. Hra ale aj napriek tomuto obmedzeniu dokáže dostatočne demonštrovať vyhľadávací algoritmus, ako aj ostatné časti implementácie. Zrušenie tohto prvku vyplynulo aj z čiastočne nevhodného návrhu generácie možných ťahov pre herný kameň.

Kapitola 4

Implementácia hry

Popis implementácie programu vytvorenému k tejto práci obsahuje detailný popis postupu pri vytváraní hlavných častí, ktoré budú stručne načrtnuté v nasledujúcom odstavci. Tieto jednotlivé časti tvoria kostru celého programu a preto je popis orientovaný práve na ne. V popise realizácie je minimum zdrojového textu a celá kapitola je zameraná hlavne na vysvetlenie princípu a predstavenie výhod a nevýhod môjho riešenia.

4.1 Návrh spracovania

V tejto časti je stručne načrtnuté základné teoretické členenie implementácie jednoduchých hier s umelou inteligenciou, ktoré som neskôr použil aj pri vytváraní implementácie pre túto prácu. Každá z nasledujúcich častí je pre pre hry s umelou inteligenciou dôležitým a neopomenuteľným prvkom. Jednotlivé časti sú popri tom vzájomne prepojené a samotný algoritmus vyhľadávania MTD(f) je len malou časťou programu.[1]

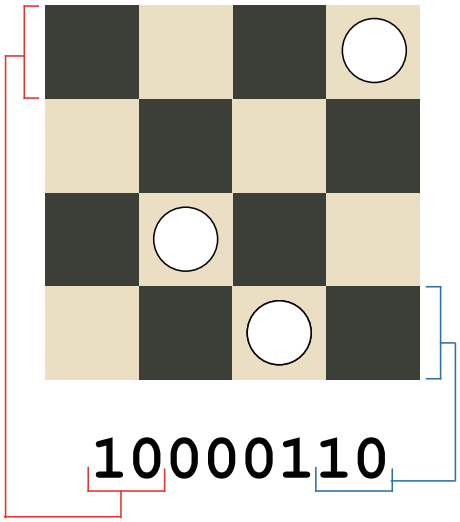
- Reprezentácia hernej plochy. Potrebujeme určitú dátovú štruktúru, prípadne dátové štruktúry, ktoré by uchovávali informáciu a stave hernej plochy a herných kameňov.
- Generovanie možných ťahov. Predstavuje určitú množinu algoritmov umožňujúcich pre každý herný kameň generovať množinu legálnych ťahov.
- Vyhľadávanie najlepšieho ťahu. V tejto časti implementácie nájde uplatnenie samotný MTD(f) algoritmus, ktorý bude slúžiť na výber najlepšieho možného ťahu pre danú hernú situáciu.
- Funkcia vyhodnocovania pozície. Táto funkcia, prípadne skupina algoritmov, je dôležitá časť implementácie, slúžiaca hlavne pre algoritmy vyhľadávania. Umožňuje určiť najvýhodnejší stav, pomocou určitých hodnotiacich kritérií.

4.2 Reprezentácia hernej plochy

Aj keď sa to na prvý pohľad nezdá, reprezentácia hernej plochy je pre výslednú implementáciu veľmi dôležitá a môže vo výraznej miere ovplyvniť náročnosť práce s programom. V prvotnej verzii programu bola moja herná plocha reprezentovaná jednorozmerným poľom záznamov typu byte, v ktorom každá položka predstavovala stav na jednom hernom políčku. Veľkosť takéhoto riešenia však nebola ideálna a nakoniec som v implementácii zvolil ako reprezentáciu hernej plochy minimalistické riešenie, 3 premenné typu *integer*. Každá z týchto premenných predstavuje jeden aspekt hernej plochy.

```
int whiteBR - predstavuje pozíciu bielych herných kameňov na ploche
int blackBR - predstavuje pozíciu čiernych herných kameňov na ploche
int queenBR - predstavuje pozíciu dám na ploche
```

Takýto štýl premennej záznamu hernej plochy sa zvykne nazývať ,,bitboard", alebo bitová tabuľka, čo znamená, že každý bit tejto premennej predstavuje jedno políčko hernej šachovnice. V tomto prípade som využil priaznivý fakt, že premenná typu *integer* má veľkosť presne 32 bitov, presne ako počet využiteľných polí hernej plochy anglickej verzie dámy. Hodnota bitu následne reprezentuje hodnotu premennej pre jednotlivé políčka.



Obrázek 4.1: Uloženie pozícií bielych herných kameňov

Napríklad hodnota premennej whiteBR 100111011011 bude znamenať, že na políčkach číslo 0, 1, 3, 4, 6, 7, 8 a 11 budú biele herné kamene. Buď bieli pešiaci alebo biele dámy. Podobne pracujeme aj so zvyšnými dvomi premennými.

Výhodou takejto reprezentácie hernej plochy je popri pamäťovej nenáročnosti aj jednoduchosť spôsobu práce s týmito hodnotami. Použitím základných bitových operácií ako je

zjednotenie, prienik alebo bitové posuny, je možné zostrojiť aj zložitejšie operácie potrebné pre prácu môjho programu.

Príklady použitia:

```
(whiteBR & queenBR) - pozície bielych dám na ploche
!(whiteBR | blackBR) - prázdne herné políčka plochy
(blackBR & !queenBR) - pozície čiernych pešiakov na ploche
```

4.3 Generovanie možných ťahov

Generácia možných ťahov je v implementácii rozdelená na dve samostatné časti. V prvej časti algoritmus pracuje len s jednoduchým pohybom a pre daný herný kameň vyhľadá všetky možné jednoduché pohyby v okolí. Druhá časť algoritmu pracuje podobne ako prvá, s tým rozdielom, že vyhľadáva iba skoky daného kameňa. V celej implementácii je generácia funkčne rozdelená na tieto dve časti, pretože každá má dosť odlišnú implementáciu. Na druhú stranu oddelenie jednoduchých pohybov a skokov často spôsobuje zneprehľadnenie zdrojového kódu.

Konkrétna realizácia vyhľadania možných ťahov je vďaka použitiu vyššie spomínaných bitových tabuliek jednoduchá a pozostáva z kombinácie bitového posunu a kontroly prekrytia záznamu voľných polí na hernej ploche so záznamom herných kameňov.

Príklad získania herných kameňov schopných pohybu:

```
    movPieces = (emptyFields << 4) & blackBR;
movPieces |= ((emptyFields & maskL3) << 3) & blackBR;
movPieces |= ((emptyFields & maskL5) << 5) & blackBR;
```

Princíp algoritmu je zjavný a posledné dva riadky sú v podstate len korekciou, ktorá vyplýva z podstaty diagonálneho pohybu po hernej ploche.

Výsledná dátový model generovaných ťahov pozostáva z indexu odkazujúceho na pohybujúci sa skáčuci herný kameň, následne z indikátoru smeru, ktorým sa môže kameň pohybovať, a nakoniec premennou rozlišujúcou obyčajný pohyb a skok.

Celá generácia ťahov je samozrejme závislá na použitých herných pravidlách, ale v prípade hry dáma, je situácia pri generovaní možných ťahov značne zjednodušená, nakoľko existujú len dva druhy pohybov, jeden pre pešiakov a jeden pre dámy. Pohyb dám je navyše ľahko spracovaný už existujúcimi metódami na spracovanie pohybu pešiakov jednotlivých farieb.

4.4 Vyhľadávanie najlepšieho ťahu

Vyhľadávanie najlepšieho ťahu je spolu s Funkciou vyhodnocovania základom prejavu a štýlu hry počítačového protivníka. Počítače nedokážu myslieť rovnako ako ľudský hráč,

ktorý pri hre často ,,vidí" najlepšie a najhoršie ťahy hneď na prvý pohľad. Záleží od skúsenosti hráča, či sú tieto ťahy najlepšie z krátkodobého alebo z dlhodobého hľadiska. Počítač však má narozdiel od ľudského protivníka tú výhodu, že dokáže pracovať rýchlo. Tak rýchlo, že mu nerobí problém vygenerovať si v pamäti neuveriteľné množstvo herných kombinácií do budúcnosti a tak určiť, ktorý ťah bude pre neho najvýhodnejší.

V implementácii sa o vyhľadávanie stará trieda *Search*, v ktorej sú popri metóde MTD(f) implementované aj ďalšie použiteľné algoritmy ako MinMax alebo Alfa-Beta.

Princíp práce týchto algoritmov bol popísaný v teoretickej časti a na rovnakom princípe pracujú tieto algoritmy aj v mojej implementácii.

4.5 Funkcia vyhodnocovania pozície

Funkcia vyhodnocovania pozície bola pri tvorbe implementácie jednou z najzložitejších častí na vytvorenie. Samotný kód funkcie nie je príliš náročný, ale celá jej podstata vyplýva z herných pravidiel a môže vo veľkej miere ovplyvniť úspešnosť vyhľadávania a tým aj kvalitu hry počítačového protivníka. V programe je implementovaných viacero verzií vyhodnocovacej funkcie, ktoré vznikli pri postupnom testovaní, ale hlavnú váhu vo všetkých má pomer počtu herných kameňov jednotlivých protivníkov.

Jednotlivé položky, na ktoré sa kladie dôraz pri určení hodnoty herného stavu:

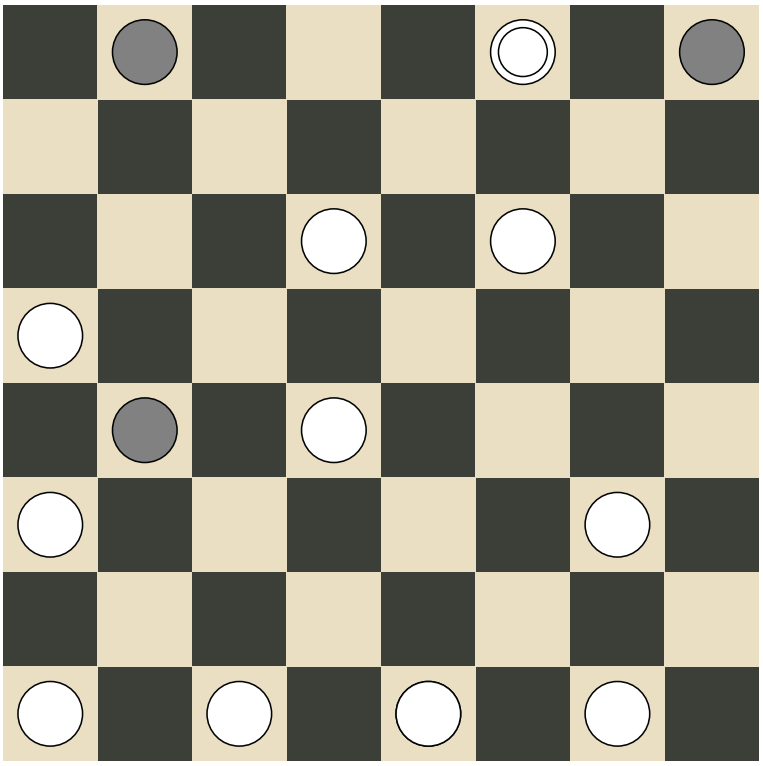
- počet pešiakov
- obrana s dámou v niektorom rohu hernej plochy
- obrana posledného radu a zabránenie súperovi povýšiť svoje herné kamene
- počet pešiakov, ktorí môžu povýšiť na dámu
- počet dám
- rozdiel počtu herných kameňov súperov

Funkcia vracia hodnotu vypočítanú podľa stavu hernej plochy, t.j. podľa počtu a pozícií jednotlivých herných kameňov. Ako už bolo uvedené, využívaná je táto funkcia hlavne v algoritme vyhľadávania najlepšieho ťahu. Tento algoritmus používa vyhodnocovaciu funkciu ako indikátor výhodnosti a nevýhodnosti možných ťahov.

4.6 Uživateľské rozhranie

Uživateľské rozhranie programu je jednoduché, bez zbytočných prídavných vymožeností, tak aby program prezentoval hlavne použitie algoritmu MTD(f) a vyhľadávacích algoritmov všeobecne v praxi.

Rozhranie programu bolo vytvorené v jazyku Java za použitia balíčku SWT, teda ,,Standard Widget Toolkit", ktorý slúži na vytváranie grafických uživateľských rozhraní v tomto



Obrázek 4.2: Vyhodnocovacia funkcia sa blíži k víťazstvu bieleho hráča

jazyku. V hlavnom okne dominuje herná plocha, ktorá je doplnená párom kontrolných prvkov. Všetky funkcie programu sú taktiež dostupné z menu, umiestneného v hornej časti obrazovky.

Konkrétna funkcia jednotlivých kontrolných prvkov je popísaná v krátkom návode na obsluhu umiestnenom v prílohe.

Kapitola 5

Výsledky implementácie

5.1 Nedostatky riešenia

Pre poskytnuté riešenie existuje veľmi veľa možných vylepšení a zjednodušení, z ktorých niektoré boli zjavné už pri začiatku implementácie a niektoré vyplynuli po tvorbe a testovaní programu. Tu sú uvedené tie najvážnejšie z nich.

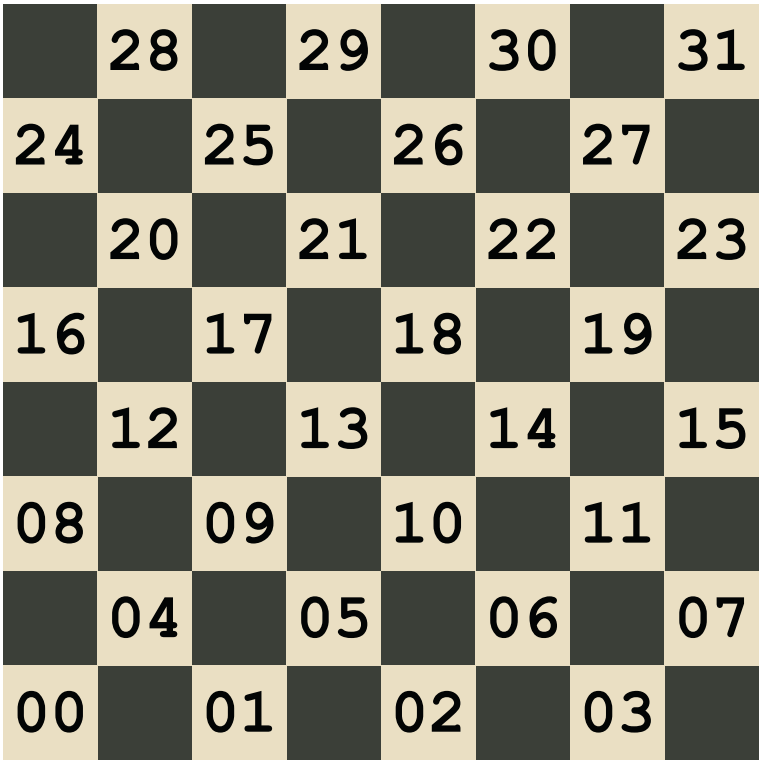
5.1.1 Problém reprezentácie hernej plochy

Jedným z najväčších problémov bol práve spôsob reprezentácie hernej plochy. Moje konečné riešenie je pravdepodobne najúspornejšie možné, ale prináša aj nevýhody. Jedna z nich vyplýva zo spôsobu očíslovania polí hernej plochy. Ako je zjavné z obrázku, pri mojom číslovaní sú výpočty vzdialeností pre párne a nepárne riadky hernej plochy rôzne. Pre tento fakt sú mnohé časti programu až príliš zložité a neprehľadné. Riešením by bolo zaviesť alternatívne číslovanie, ktoré by dokázalo tento problém korigovať. V súčasnosti existuje množstvo rôznych číslovaní, ktoré sú v tomto smere výhodnejšie ako to implementované.

Ďalším problémom, ktorý plynie z reprezentácie hernej plochy, je nutnosť občasného prevodu binárnej reprezentácie hernej plochy na decimálnu a naopak. Tento problém sťažuje najmä to, že nejde o regulárny prevod binárnych na decimálne čísla, ale o určitú formu binárneho kódovania.

5.1.2 Optimalizácia vyhľadávacích algoritmov

Okrem reprezentácie hernej plochy sú nedostatky riešenia hlavne v implementácii vyhľadávacích algoritmov. Je možné výrazne zvýšiť ich rýchlosť dodatočnou optimalizáciou. Hlavne v prípade metódy MTD(f), ktorá využíva len jednoduchú verziu algoritmu Alfa-Beta bez pamäte, teda bez použitia transpozičnej tabuľky. Transpozičná tabuľka má na starosti ukladanie informácii o tých vetveniach v Alfa-Beta algoritme, ktoré už boli raz programom prehľadané. Tento spôsob je výhodný najmä pre metódu MTD(f), ktorá volá algoritmus Alfa-Beta veľmi často. Transpozičná tabuľka je taktiež použiteľná pri ukladaní údajov



Obrázek 5.1: Spôsob interného číslovania hernej plochy

o výhodnom zahájení hry, ako sú napríklad zoznamy otváracích ťahov najlepších profesionálnych hráčov. Samozrejme tento spôsob urýchlenia vyhľadávania nie je príliš markantný u hry akou je dáma.

Ďalší spôsob možného zefektívnenia práce vyhľadávacích algoritmov je tabuľka histórie, ktorá by uchovávala zoznam ťahov, ktoré priniesli neobvykle výhodný (prípadne nevýhodný) výsledok. Program by si pamätal, že viackrát odohral tento ťah a po analýze výsledkov by mohol určiť dôsledky odohrania tohto ťahu na všetky hry v budúcnosti.

5.1.3 Generovanie možných ťahov

Posledným nedostatkom a zrejme aj najväčším je nepríliš vhodný spôsob generovania možných ťahov, resp. dátové štruktúry reprezentujúce tieto ťahy. V implementovanej verzii sú jednotlivé ťahy rozdelené na obyčajné pohyby a skoky, takže akákoľvek práca s vygenerovanými ťahmi zvyčajne vyústila do zdvojnásobenia relevantnej časti zdrojového kódu. Každý ťah bol pritom reprezentovaný indexom pohybovaného kameňa a smeru ktorým sa kameň na ploche uberá, prípadne skáče. Táto reprezentácia sa tiež neukázala ako vhodná, nakoľko bolo často nutné dopočítať pozíciu, na ktorú sa pohybovaný kameň presúval, resp. skákal.

Tieto faktory spôsobili aj moje rozhodnutie obmedziť viacnásobné skoky, pretože ich

generovanie vo vyhľadávacích algoritmoch bolo na implementáciu príliš zložité. Program teda podporuje len jednoduché skoky s vyradením jedného nepriateľského kameňa.

Kapitola 6

Záver

Zložitosť a výskum u každej jednotlivej časti teórie tvorby hier s umelou inteligenciou ma dosť prekvapili, a preto je aj táto práca komplexnejším pohľadom na všetky časti tvorby hry a nielen zameraním na metódu vyhľadávania MTD(f). Jednotlivé časti sú natoľko previazané, že podcenenie ktorejkoľvek z nich prináša automaticky problémy aj pre tie ostatné. Funkcia ohodnotenia stavu môže byť jednoduchá a založená na najzákladnejšom pravidle, ale potom výrazne utrpí efektívnosť vyhľadávania. Rovnako je to aj s previazanosťou vyhľadávacieho algotimu a reprezentáciou hernej plochy, ktorá môže výrazne ovplyvniť pamäťovú náročnosť pri nesprávnom použití. Avšak rozpísať všetky tieto celky do detailu by zabralo oveľa viac strán ako má táto bakalárska práca.

Samotný vývoj implementácie bol na druhú stranu o to zaujímavejší a hodnotnejší, nakoľko ma nútil hľadať riešenia ku každému z týchto problémov a tak mi poskytoval stále nové podnety. V súčasnosti okrem toho pre danú tému existuje až neuveriteľne veľké množstvo programov, návodov a publikácií, a tak nebol problém získať odpovede na jednotlivé otázky. Dokonca naopak, niekedy bolo informácií k téme až príliš a ťažké bolo vybrať len tie hodnotnejšie. V každom prípade ma téma zaujala, a to aj napriek tomu, že sa mi nepodarilo dopracovať demonštračný program do takej podoby ako by som dúfal.

Literatura

[1] Francois-Dominic Laramée. Chess programming. http://www.gamedev.net/reference/programming/features/chess1/.

[2] Aske Plaat. Mtd(f). http://www.cs.vu.nl/ aske/mtdf.html.

[3] WWW stránky. History of checkers or draughts. http://www.indepthinfo.com/checkers/history.shtml.

[4] WWW stránky. Minimax method. http://en.wikipedia.org/wiki/Minmax.

[5] WWW stránky. Bbc news: Robotic age poses ethical dilemma. http://news.bbc.co.uk/2/hi/technology/6425927.stm, 2007.