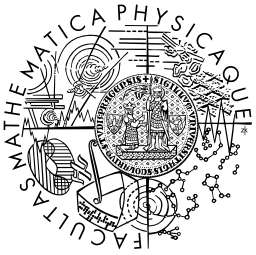


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

David Bělíček

Umělá inteligence pro deskovou hru dáma

Katedra teoretické informatiky a matematické logiky

Vedoucí bakalářské práce:	Mgr. Jiří Švancara
Studijní program:	Informatika
Studijní obor:	Programování a softwarové systémy

Praha 2020

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V Praze dne 30.7.2020 Podpis autora

Chtěl bych poděkovat svému vedoucímu Jiřímu Švancarovi za jeho cenné rady a pomoc při vytváření této práce. Dále bych chtěl poděkovat své rodině za podporu.

Název práce: Umělá inteligence pro deskovou hru dáma

Autor: David Bělíček

Katedra: Katedra teoretické informatiky a matematické logiky

Vedoucí bakalářské práce: Mgr. Jiří Švancara, Katedra teoretické informatiky a matematické logiky

Abstrakt: Dáma je desková hra, která se v různých formách hraje po celém světě. Cílem této práce je teoreticky popsat a implementovat algoritmus umělé inteligence, která bude schopna hrát dámu. Vysvětlíme si co je to algoritmus Minimax, jak ho zefektivnit pomocí Alfa-Beta prořezávání a jeho verzi s omezenou hloubkou, která využívá heuristických ohodnocení. Uvedeme si dvě konkrétní heuristická ohodnocení, jak tyto ohodnocení nahradit neuronovou sítí a jak tyto sítě vyvíjet pomocí evolučních algoritmů. Nakonec uděláme několik experimentů, ve kterých otestujeme vytvořené heuristiky a sítě. Práce je zakončená turnajem, jenž rozhodne který z vyvinutých algoritmů je nejlepší.

Klíčová slova: deskové hry, umělá inteligence, algoritmus minimax, heuristické funkce

Title: Artificial Inteligence for Draughts

Author: David Bělíček

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: Mgr. Jiří Švancara, Department of Theoretical Computer Science and Mathematical Logic

Abstract: Draughts is a board game that is played all around the world in various forms. The aim of this thesis is to describe and implement an artificial intelligence algorithm that will be able to play draughts. We will explain the working of Minimax algorithm, how to enhance it using Alpha-Beta pruning, and its limited-depth version, which uses heuristic evaluations. We will present two hand-crafted heuristic evaluations, how such heuristic evaluation can be replaced with a neural network, and how to develop these networks using evolutionary algorithms. Finally, we will perform experiments in which we will test the created heuristics and networks. At the end of the thesis, we present a tournament that decides which of the developed algorithms is the best.

Keywords: board games, artificial intelligence, minimax algorithm, heuristic functions

Obsah

Úvod

Dáma je tradiční desková hra pro dva hráče, která se hraje v mnoha zemích světa. Má mnoho různých verzí, které mají společné charakteristické prvky. Hraje se na šachovnicové desce s kameny bílé a černé barvy. Kameny se pochybují pouze diagonálně. V zahraničí je známa pod jménem Checkers, nebo taky Draughts. Pořádají se v ní soutěže a existuje „Světová federace dámy“.

Podobně jako u šachů, je ve světě snaha o to, vytvořit umělou inteligenci, která dokáže hrát proti člověku. V roce 2007 se povedlo vyřešit verzi dámy podle Anglických pravidel. 18 let trvající výpočet pomocí umělé inteligence „State-of-the-art“ prošel celý strom hry a dokázal, že při bezchybné hře je zaručena remíza (Schaeffer a kol., 2007). Vytvoření bezchybného algoritmu, který vždy najde optimální strategii, je zajímavé z teoretického pohledu, ale pro hráče to znamená, že si s ním nikdy nezahraje, protože nemůže vyhrát. Proto tu zbývá prostor pro nedokonalé algoritmy, které nehrají optimálně, ale hrají dostatečně dobře.

Cílem této práce je vytvořit program (framework), ve kterém si bude uživatel moct zahrát proti formě umělé inteligence. Jednou takovou formou je algoritmus Minimax s omezenou hloubkou, který si teoreticky popíšeme a pak i implementujeme. Tento algoritmus využívá heuristického ohodnocení stavu hry, které herní desce přiřadí číslo. Dvě takovéto ohodnocení navrhneme. Ukážeme si, že místo heuristických ohodnocení se dá použít i neuronová síť a jak takové sítě vyvinout pomocí neuroevoluce.

Nakonec provedeme několik experimentů, s cílem porovnat všechna ohodnocení a určit, které je nejlepší.

1. Pravidla hry Dáma

Dáma je desková hra pro dva hráče. Hraje se na čtvercové desce s pomocí pohyblivých kamenů. Každý hráč může pohybovat jen kameny své barvy. Cílem hry je sebrat, nebo zablokovat všechny kameny soupeře. Doba vzniku se uvádí okolo 15. století. Avšak hry velmi podobného charakteru se objevují už již 3000 let před naším letopočtem. Popíšeme si zde Obecná pravidla a dvě verze konkrétních pravidel, ke kterým se pak budeme odkazovat dále v práci.

1.1 Obecná pravidla

Pravidla Dámy mají mnoho verzí v různých zemích. Obecná pravidla se snaží popsat společné prvky všech verzí, avšak i ty jsou většinou upravována konkrétními verzemi.

Deska

Dáma se hraje na šachovnicové desce s 8x8, nebo 10x10 poli. Levé horní pole je vždy světlé a hraje se pouze na tmavých polích.

Kameny

Hráči hrají s bílými a černými kameny. Většinou začíná hráč s bílými kameny. Rozlišujeme dva typy kamenů. Obyčejný kámen a dáma.

Počáteční umístění

V počáteční pozici mají hráči umístěné obyčejné kameny na protějších stranách desky.

Pohyb kamenů

Všechny typy kamenů se mohou pohybovat pouze diagonálně. Obyčejné kameny se mohou pohybovat jen dopředu o jedno pole. Toto pole, na které chceme kámen posunout, musí být volné.

Dáma

Kámen, který dojde na druhý konec desky, je povýšen na dámu. Dáma se může pohybovat i dozadu. Konkrétní pravidla určují, jestli se dáma může pohybovat jen o jedno pole, nebo o libovolný počet volných polí.

Braní kamenů

Pokud se před kamenem nachází soupeřův kámen a je za ním volné pole, tak lze na toto volné pole skočit a soupeřův kámen tak sebrat a vyřadit ze hry. Pokud jde s tímto kamenem skákat znovu, tak je povinné skákat dále, dokud to jde. Některá pravidla umožňují i skok s obyčejným kamenem dozadu. Skákání je většinou povinné. Má-li hráč na výběr který kámen přeskočit, může si vybrat

který, avšak musí skákat dokud to jde. Pokud tedy přeskočí soupeřův kámen a je schopný tím samým kamenem udělat další skok, tak musí „skákat dál“. Soupeřovy kameny, které jsou přeskočeny, jsou odebrány ze hry až po tom, co je dokončena sekvence skoků. Žádný kámen nelze přeskočit vícekrát.

Konec

Hráč, který nemá žádné kameny, nebo jsou všechny jeho kameny zablokované, bez možnosti pohybu, prohrává.

Remíza

Pokud se hráči dostanou do situace, kdy žádný není schopen vyhrát, neudělají-li oba chybu, pak hra končí remízou. Protože to není exaktní definice a není lehké říct, že taková situace nastala, oba hráči se musí shodnout, že hrát dál už nemá smysl a zahlásí remízu.

1.2 Česká pravidla

Oficiální pravidla české verze určuje Česká federace dámy. Ta byla založena v roce 1964 a stala se členem Světové Federace Dámy.

- Hraje se na šachovnici 8x8. Každý hráč má 12 kamenů umístěných v prvních třech řadách desky. Začíná hráč s bílými kameny. (Obrázek 1.1)
- Kameny nemohou skákat dozadu. Dáma se může pohybovat o libovolný počet polí.
- Kámen, který dojde na konec desky je povýšen na dámu a jeho tah končí.
- Skákání je povinné a skákání s dámou má přednost před skákáním s normálním kamenem.
- Dáma může skočit na kterékoliv pole za kamenem, který přeskakuje, ale pokud některé z nich vede k dalšímu skoku, tak na takové pole musí a skákání pokračuje.

Pro zajímavost si zmíníme dvě další verze, které se u nás hrají. Od oficiální verze se liší jen trochu.

Česká dáma dle Zapletala

V této verzi se hraje pouze s osmi kameny, na dvou řadách.

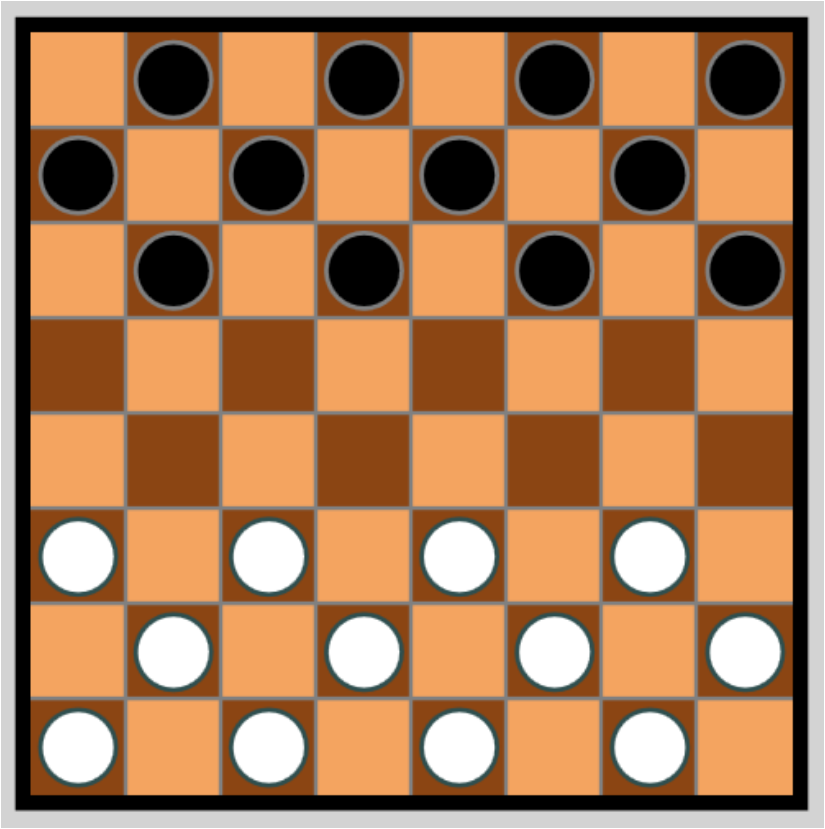
Žravá dáma

Tato verze má opačný cíl. Vyhrává ten, který není schopen hrát dál. Cílem je tedy nechat si sebrat, či zablokovat všechny kameny.

1.3 Anglická pravidla

Anglická pravidla se hrají v zemích jako Velká Británie a Spojené Státy Americké. Také se nazývají „Americká dáma“. Je to nejpopulárnější forma pravidel. Velmi často bývá předmětem zkoumání (Bosboom a kol., 2018; van Horssen, 2018).

- Hraje se na šachovnici 8x8. Každý hráč má 12 kamenů umístěných v prvních třech řadách desky. Začíná hráč s černými kameny. (Obrázek 1.1)
- Kameny se nemohou pohybovat dozadu, ani skákat dozadu. Dáma se může pohybovat pouze o jedno pole, jako obyčejný kámen, dopředu i dozadu.
- Skákání je povinné. Má-li na výběr, může si vybrat, se kterým kamenem skočí.



Obrázek 1.1: Startovní pozice podle českých i anglických pravidel.

2. Umělá inteligence

V této kapitole si ukážeme, jak vytvořit umělého agenta schopného hrát Dámu. Začneme tím, co je to vlastně agent. Agent je cokoliv, co vnímá prostředí a nějak ho ovlivňuje. V našem případě je to entita, která vidí stav herní desky a řekne, který tah zahrát. Formálně takového agenta můžeme popsat **agentovou funkcí**: $S \to T$, kde S je stav herní desky a T je tah ve hře.

2.1 Dáma v teorii her

Pro agenta je také důležité prostředí. Dáma odpovídá multiagentnímu prostředí. Přesněji je to dvouagentní prostředí, protože proti sobě hrají právě dva hráči. Je to hra s nulovým součtem. To znamená, že zisk jednoho hráče, odpovídá ztrátě druhého. Dále je to hra deterministická, protože v ní není žádný prvek náhody. Oba hráči mají spolu plně pod kontrolou, jak hra probíhá. A nakonec je to hra s úplnou informací, protože oba hráči vidí celý stav hry. Na rozdíl třeba od karetních her, kde nevíme, co má soupeř na ruce za karty a nemůže proto předvídat jeho tah.

Tyto informace jsou pro nás důležité, abychom se rozhodli jakého agenta zvolit. Pro takovéto hry, existuje algoritmus, který nám zaručuje nalezení optimální strategie. Nazýváme ho Minimax (Russell a Norvig, 2010).

2.2 Minimax

Nejprve si zadefinujeme pár pojmů, které budeme potřebovat znát. Hráče, který je na řadě a pro kterého chceme najít strategii, nazveme MAX a druhého MIN. Důvod uvidíme za okamžik. Budeme používat ohodnocující funkci, která koncovým stavům hry přiřadí číselnou hodnotu. Říká se jí také funkce zisku, nebo utilitní funkce. Výherním stavům pro hráče MAX, přiřadí hodnotu 1. A výherním stavům pro hráče MIN, přiřadí hodnotu -1. Remízovým stavům přiřadíme hodnotu 0.

Algoritmus teoreticky

1. Vygenerujeme strom hry.
2. Ohodnotíme listy podle ohodnocující funkce.
3. Ohodnotíme vnitřní vrcholy nad listy následovně: Je-li vrchol v sudé hloubce (na tahu je hráč MAX), pak vybereme nejvyšší hodnotu z potomků. Je-li vrchol v liché hloubce (na tahu je hráč MIN), pak vybereme nejnižší hodnotu z potomků. Kořen stromu indexujeme jako hloubku 0.
4. Takto ohodnotíme další vrcholy nad nimi a postupujeme až do kořene.
5. V kořeni vybereme tah, který vede do stavu s nejvyšší hodnotou. Tento tah zahrajeme.

Ohodnocování vrcholů jsme si pro snazší pochopení popsali směrem od listů do kořene. Pro takový algoritmus, bychom ale potřebovali mít v paměti uložený celý strom hry, ale to je zbytečné. Ve skutečnosti strom procházíme od kořene do listů a ohodnocení vrcholů provádíme rekurzivně. To znázorňuje Algoritmus 1. Tento způsob procházení se nazývá průchod do hloubky. V paměti udržujeme pouze $O(bd)$ vrcholů. Kde b je index větvení a d je hloubka stromu. Index větvení stromu hry, odpovídá tomu, kolik má hráč možných tahů.

Algoritmus 1 Algoritmus minimax.

```
1: function MINIMAX(vrchol v, boolean maximalizujeme)
2:     if Koncovy(v) then
3:         return Eval(v)
4:     end if
5:     if maximalizujeme == TRUE then
6:         ohodnoceni = −∞
7:         for each potomek ∈ Potomci(v) do
8:             ohodnoceni = max(ohodnoceni, minimax(potomek, FALSE))
9:         end for
10:    else
11:        ohodnoceni = ∞
12:        for each potomek ∈ Potomci(v) do
13:            ohodnoceni = min(ohodnoceni, minimax(potomek, TRUE))
14:        end for
15:    end if
16:    return ohodnoceni
17: end function
```

Největší nevýhoda Minimaxu, je jeho časová složitost. To brání jeho nasazení u komplexnějších her. Nejsme totiž schopni projít celý strom až do listů, jelikož takový strom bývá příliš velký. U her jako jsou piškvorky to možné je, hrajeme-li na menší ploše. Avšak u her jako je Dáma, nebo Šachy to možné není. Takový strom je dokonce nekonečný. Později si ukážeme, jak tohle vyřešit.

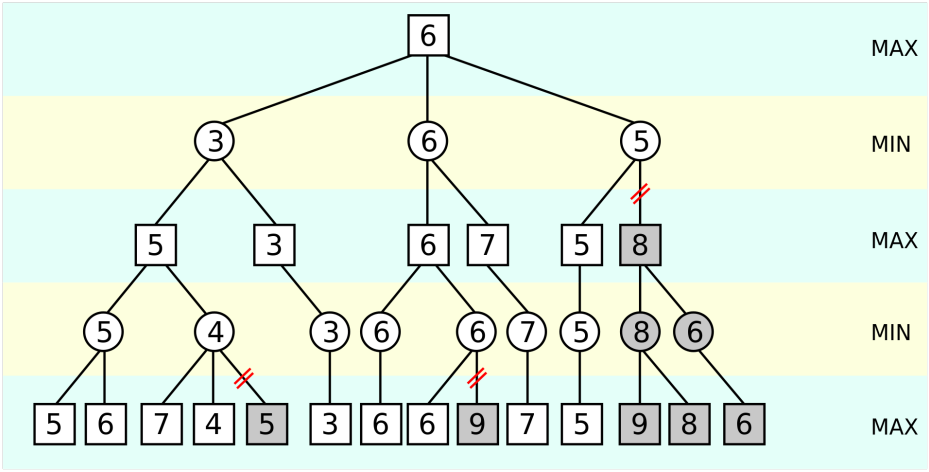
2.2.1 Alfa-Beta prořezávání

Minimax je možné vylepšit. Některé větve stromu hry, totiž vůbec nemusíme procházet. Popíšeme si to na obrázku 2.1. Strom procházíme zleva doprava. Zaměřme se na zašedlou větev na pravé straně obrázku. Nacházíme se ve stavu, kdy jsme v hloubce 1 a vybíráme za hráče MIN. Prošli jsme jednu větev, která je ohodnocena číslem 5. A teď bychom se měli vydat do té zašedlé větve. Ukážeme si, že to není třeba.

Jsme hráč MIN a chceme co nejnižší hodnotu. Momentálně udržujeme nejnižší hodnotu 5 a vyšší už nevybereme. Hráč o úroveň výš je MAX a jeho nejlepší hodnota je 6. Ten si hodnotu 5 ani nižší nevybere, takže tuto větev vůbec zkoumat nemusíme.

Stačí k tomu, abychom znali dosavadní nejlepší hodnotu „hráče nad námi“. Pak jsme schopni udělat porovnání, jestli má smysl zkoumat další větev. Záleží na pořadí, v jaké větve procházíme. Projdeme-li je ve špatném pořadí, pak to musíme projít vše. Bohužel nejsme schopni vědět předem, jaké pořadí je správné.

Tomuto algoritmu se říká Alfa-Beta prořezávání (Knuth a Moore, 1975; Russell a Norvig, 2010), ale ani ten nestačí na to, aby vyřešil problém s časovou náročností u netriviálních her.



Obrázek 2.1: Příklad alfa-beta prořezávání. Zašedlé větve nemusíme procházet. Obrázek převzat z (Wikipedia contributors, 2020a).

2.2.2 Minimax s omezenou hloubkou

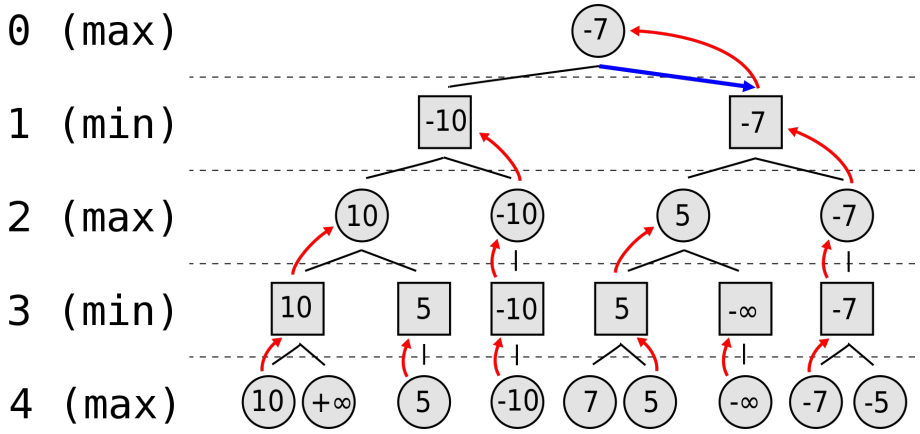
Existuje úprava minimax algoritmu, která nám vyřeší problém s časem, ale za cenu toho, že nebudeme mít zaručenou optimální strategii (Campbell a Marsland, 1983; Russell a Norvig, 2010). Namísto abychom procházeli celý strom až do listů, projdeme strom jen do předem určené hloubky. Vrcholy v té hloubce ohodnotíme podle nějaké heuristické ohodnocující funkce a tyto hodnoty pak zpětně propagujeme až do kořene, stejně jako v klasickém minimax algoritmu. Také této funkci můžeme zkráceně říkat heuristika. Příklad stromu hloubky 4 vidíme na obrázku 2.2. Na začátku si pevně zvolíme, do jaké hloubky půjdeme. Buď podle dostupného výkonu, a nebo podle požadované obtížnosti. Větší hloubka stromu je lepší, protože algoritmus „vidí“ více tahů dopředu a je schopen plánovat.

Protože výherní stav, je ten nejlepší tah z pohledu vítěze a ten nejhorší tah z pohledu poraženého, pak výherní stavy pro hráče MAX vždy ohodnotíme $+\infty$ a výherní stavy pro hráče MIN vždy ohodnotíme $-\infty$. Ostatní stavy musíme analyzovat a poté ohodnotit podle toho, který hráč je ve výhodě. Tato analýza, je ukrytá v heuristické ohodnocující funkci. Jakou heuristickou ohodnocující funkci zvolíme je velmi důležité. Čím lépe jsme schopni ohodnotit stav, tím lepší „rozhodnutí“ dokáže minimax udělat. Předtím nám minimax našel tah, který zaručeně vede k nejlepšímu stavu. Ale teď nás vede jen k nejlépe ohodnocenému stavu.

V sekci 3.1.5 se budeme zabývat ohodnocením stavů v Dámě.

Problém s horizontem

Horizont jsou všechny vrcholy v hloubce, kterou jsme omezeni. Jsou to listy našeho omezeného stromu hry. Přítomnost tohoto horizontu, přináší problém.



Obrázek 2.2: Minimax s omezenou hloubkou 4. Obrázek převzat z (Wikipedia contributors, 2020b).

Uvažme následující situaci. Máme nevyhnutelný tah, který nás bude stát dost bodů v ohodnocení. Může to být dokonce tah, který nám přinese prohru. Nazveme ho nechtěný tah. Není to ale jediný tah, který můžeme zahrát. Existují jiné tahy, které v tu chvíli nutí soupeře reagovat hned, ale nepřináší žádnou dlouhodobou výhodu. Tyto tahy nazveme zdržovací tahu. Pokud je zdržovacích tahů dost na to, aby se nechtěný tah posunul až za horizont, pak bude algoritmus preferovat tuto větev a tím odsouvat nevyhnutelný stav.

2.3 Neuronová síť jako ohodnocující funkce

Přijít s dobrou heuristikou není jednoduché. S tím nám pomohou neuronové sítě. Ukážeme si, jak je použít jako heuristickou ohodnocující funkci a jak najít, respektive vyvinout takovou síť pomocí neuroevoluce. Začněme tím, co to vůbec je.

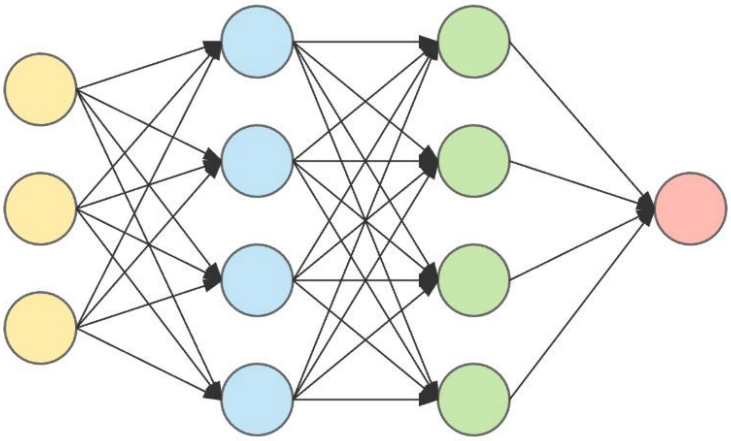
2.3.1 Vrstevnatá neuronová síť

Neuronové sítě, jak už název napovídá, jsou inspirované v přírodě nervovou soustavou (Soltoggio a kol., 2018).

Neuronová síť je orientovaný graf, jehož vrcholy nazýváme neurony (Haykin, 1999). Ke každé hraně je přiřazené číslo, které udává váhu hrany mezi neurony. Neurony rozdělujeme na vstupní, skryté a výstupní.

Vrstevnatá síť je zobrazena na obrázku 2.3. Je specifická tím, že její neurony jsou rozděleny do lineárně uspořádaných skupin (vrstev). Každý neuron je spojen hranou s každým neuronem v předchozí a následující vrstvě a s žádným jiným. Hrany jsou orientovány směrem do vyšších vrstev. První vrstvu tvoří vstupní neurony a proto ji také nazýváme vstupní vrstva. Stejně tak výstupní neurony jsou v poslední vrstvě, kterou nazýváme výstupní vrstva. Zbývají skryté vrstvy se skrytými neurony.

Samotná síť pak funguje jako funkce. Vstupním neuronům přiřadíme vstupní data v podobě číselných hodnot. Podobně jako argumenty funkce. Poté provádíme výpočet po vrstvách a nakonec hodnoty ve výstupní vrstvě jsou výstupní data.



Obrázek 2.3: Vrstevnatá neuronová síť.

Zadefinujeme si to formálně. Nechť $a_{i,j}$ je hodnota neuronu j ve vrstvě i a $w_{i,j,k}$ je váha hrany mezi neuronem j ve vrstvě i a neuronem k ve vrstvě $i+1$. Pak

$$a_{i+1,k} = f(\sum_{j=0}^{n} a_{i,j} w_{i,j,k}),$$

kde f je přechodová funkce. Příkladem přechodových funkcí jsou Sigmoid, Tanh, ReLu, Lineární funkce (Sharma, 2017).

2.3.2 Neuroevoluce

Je mnoho způsobů, jak vyvinout neuronovou síť, aby dělala to co po ní chceme. Slovem „vyvinout“ síť, myslíme najít ideální topologii sítě a váhy hran.

V našem případě nemáme přímou odezvu, jestli stav hry ohodnotila dobře. Jeden ze způsobů, jak vyvinout neuronovou síť je pomocí evolučních algoritmů. Zkráceně tomu říkáme neuroevoluce (Stanley a kol., 2019).

Evoluční algoritmy jsou optimalizační metody, inspirované Darwinistickou evolucí (Vikhar, 2016; Sloss a Gustafson, 2019). Pracují s množinou jedinců, které říkáme populace. Jedinci jsou reprezentováni genomem. Populace se mění každou generací. Na jedince v populaci se aplikují genetické operátory a tím vznikne nová další generace jedinců. Genetické operátory jsou Selekce, Křížení a Mutace (Sloss a Gustafson, 2019).

Algoritmus 2 Algoritmus Evoluce.

```
1: p0 = InicializacePopulace()
2: populace = SpočteníFitness(p0)
3: for each i ∈ [1, ..., počet_generací] do
4:     p = Selekce(populace)
5:     p = Křížení(p)
6:     p = Mutace(p)
7:     populace = SpočteníFitness(p)
8: end for
```

Popíšeme si operátory obecněji, protože jsou pro každý problém jiné. V sekci 3.2.2 si pak uvedeme konkrétní implementace, které jsme zvolili pro náš problém.

Fitness

Fitness hodnocení je číselná hodnota, která říká, jak je jedinec dobrý. Je určena otestováním jedince.

Selekce

Selekce je výběr několika jedinců, které postoupí do další generace. Výběr je určen jejich fitness ohodnocením.

Elitismus

Při selekci je šance, že se do další generace dostanou jen horší jedinci a tím ztratíme všechny dobré jedince. Proto se často dělá, že několik nejlepších jedinců vybereme automaticky.

Křížení

Vybraní jedinci mají nějakou šanci na křížení. Při křížení se vezmou dva jedinci, kterým říkáme rodiče a promíchají si genetickou informaci, pro stvoření potomků.

Mutace

Mutace je menší náhodná změna v genomu jedince.

3. Framework

V této kapitole probereme strukturu kódu, technické detaily, použité algoritmy a způsob implementace samotné hry. Celý framework je rozdělen do dvou projektů Draughts a Controller.

V příloze práce je popsáno, jak s projekty pracovat.

3.1 Projekt Draughts

Hlavním je projekt Draughts. Je to okenní aplikace, ve které sídlí jádro samotné hry. Stará se o průběh a vizualizaci hry. Název vychází z britského pojmenování této hry.

V projektu jsou implementována verze Českých a Anglických pravidel, které jsme si popsali v kapitole 1. V příloze A.1.1 nalezneme popis ovládacích prvků výsledného programu.

3.1.1 Reprezentace kamene

Začneme popisem nejmenšího prvku a tím je jedno herní pole. K tomu abychom popsali stav jednoho pole, potřebujeme vědět:

- Jestli se na poli nachází kámen
- Barvu kamene
- Status kamene (jestli je to dáma, nebo obyčejný kámen)

Každá z těchto tří informací se dá zakódovat do jednoho bitu. Dohromady nám vychází, že na jedno pole potřebujeme 3 bity informace.

Může nás napadnout, že není-li na poli žádný kámen, pak je nám informace o barvě a o statusu zbytečná. To je pravda a jedno pole nabývá jen pěti stavů, namísto osmi. Ovšem 5 stavů nelze reprezentovat méně než třemi bity. Protože dvěma bity můžeme reprezentovat pouze 4 stavy. A nic mezitím není.

3.1.2 Reprezentace herní desky

Stav herní desky je klíčový prvek ve hře. Chceme-li, aby implementace algoritmu minimax byla efektivní, toto je jedno z míst na které je potřeba se zaměřit. Při prohledávání stromu hry do hloubky 9, totiž projdeme řádově stovky tisíců stavů. Potřebujeme objekt herní desky rychle stvořit a smazat.

K uložení stavu herní desky použijeme bitové kódování. Už víme, že k reprezentaci jednoho kamene, nám stačí jen 3 bity. Samotná herní deska mívá 64 (8x8), nebo 100 (10x10) polí. Sice existují verze dámy, které se hrají na herní desce velikosti 12x12, ale není jich mnoho a pro zjednodušení tyto verze nebudeme podporovat. Z pravidel víme, že se hraje pouze na černých polích desky. Díky tomu, nám stačí uchovávat stav jen poloviny polí na desce. Pro větší desku 10x10 to znamená, že se dá hrát jen na padesáti polích. Už víme, že na jedno pole potřebujeme 3 bity, takže k uložení celé herní desky tedy potřebujeme 150 bitů. Zakódování provedeme do tří 64-bitových celočíselných proměnných.

3.1.3 Generování tahů

Pro přesné popsání tahu kamene, nám nestačí pozice odkud a kam se pohybuje. V případě skoku, je nutné vědět všechny pozice, po kterých kámen skáče a také pozice kamenů, přes které skáče. Hodí se mít uložený i příznak, jestli se jedná o skok, nebo ne.

Projdeme všechny kameny hráčovi barvy a prozkoumáme, jestli pole na které by se mohl posunout jsou volná. U skoků je potřeba řešit to, že v sekvenci skoků se cesta větví a můžeme si vybrat, kterým směrem se vydat.

Při hledání tahů je potřeba myslet na to, že některé tahy mají přednost a proto se jiné nedají zahrát. Například v českých pravidlech, rozdělíme tahy do kategorií, seřazené podle priority sestupně:

1. Brací tah dámou.
2. Brací tah obyčejným kamenem.
3. Tah obyčejným kamenem, nebo dámou.

A vybereme jen tahy z první neprázdné kategorie.

3.1.4 Minimax

Co je to minimax jsme si popsali v kapitole 2. Minimax algoritmus jsme implementovali s malou úpravou, která ale neovlivní jeho výsledek. Změna spočívá v tom, že hráč MAX a MIN není určen tím, za koho agent hraje, ale je pevně dáno, že hráč s bílými kameny je MAX a hráč s černými kameny je MIN. To umožní ohodnocení implementovat nezávisle na tom, za koho agent hraje a každou desku ohodnotí úplně stejně. Fungování algoritmu to nijak neovlivní, protože Dáma je hra s nulovým součtem.

Algoritmus není deterministický. Pokud existuje více vrcholů se stejným optimálním hodnocením, pak vybere jeden z nich náhodně. Bez toho by každá hra agentů, používajících minimax algoritmus, dopadla stejně.

Ukážeme si dvě implementované optimalizace, které nám pomůžou zefektivnit algoritmus.

Opakování stavů

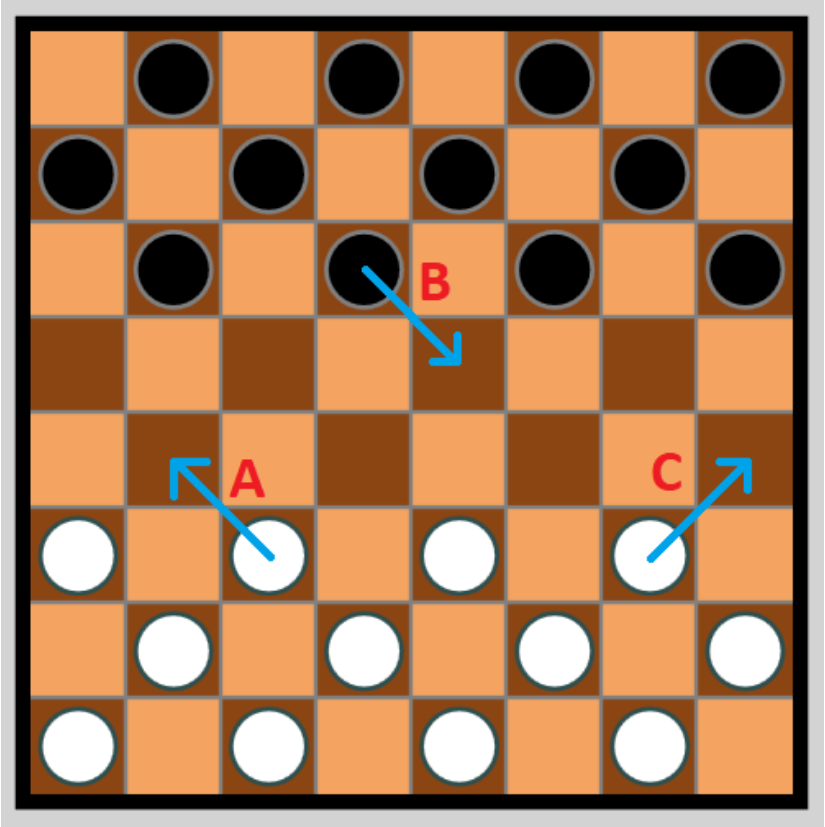
Dokud hrajeme pouze s obyčejnými kameny, bez dámy, tak jsou všechny tahy pouze dopředné. Nemůžeme se dostat do pozice, ve které jsme byli o pár tahů zpátky. Příchod dámy, která se pohybuje o libovolný počet polí i dozadu, přináší velké zpomalení pro minimax algoritmus, protože dáma má mnohem více tahů, které může zahrát. To výrazně zvýší index větvení stromu hry.

Také to přináší možnou situaci, kdy oba hráči zahrají nějaký tah s dámou a hned další tah se oba vrátí zpátky, kde stáli předtím. Tyto tahy nijak nevedou k rozvoji hry a neposunou se blíže k cíli. Při prohledávání proto takové tahy nebudeme brát vůbec v potaz.

Využijeme toho, že minimax je prohledávání do hloubky a pamatujeme si historii stavů až do kořene. Narazíme-li na tah, který vede do stavu, ve kterém jsme už byli, pak tento tah ignorujeme.

Cachování prozkoumaných stavů

Do některých stavů, se můžeme dostat více způsoby. Například hráč 1 zahraje nějaký tah, nazveme ho tah A, poté hráč 2 zahraje nějaký tah B a nakonec hráč 1 zahraje tah C. Pokud se tyto tahy navzájem nijak neovlivňují, pak existuje možnost je zahrát v pořadí C, B, A a dostaneme se do stejného stavu, jako prvním případě. Příklad znázorňuje obrázek 3.1.



Obrázek 3.1: Příklad tahů, které vedou do stejného stavu, nehledě na to, jestli se zahrají v pořadí A-B-C, nebo C-B-A.

Proto se hodí mít uložené již prozkoumané stavy a u nich výsledek prohledávání. Před prozkoumáním každého stavu se podíváme jestli ho nemáme uložený a pokud ano, pak použijeme uložený výsledek.

3.1.5 Heuristická ohodnocující funkce v Dámě

Díky úpravě minimaxu nemusíme počítat s informací, který hráč je na tahu. Ukážeme dvě jednoduchá ohodnocení, které implementujeme. To jak dobré jsou uvidíme v kapitole 4. Budeme hovořit o bodech. Počet bodů na konci tvoří číslo, kterým ohodnotíme stav hry.

Ohodnocení Basic

Prvním je ohodnocení Basic. Název je anglický výraz pro „základní“. To proto, že toto ohodnocení dělá jen úplný základ a sice to, že jen počítá kameny. Za

každý bílý kámen body přidává a za černý kámen body ubírá. Obyčejný kámen má přiřazenou hodnotu 1 bod a dáma má hodnotu 5 bodů.

Myšlenka za tímto ohodnocením je ta, že čím více kamenů hráč má oproti soupeři, tím lépe na tom je. Protože cílem hry, je sebrat soupeřovi kameny a nenechat si sebrat svoje.

Ohodnocení Progressive

Ohodnocení Progressive je vylepšený Basic. Snaží se do hodnocení brát v potaz i postavení kamenů. Myšlenka je taková, že je dobré rozehrávat všechny kamenu dopředu. Stejně jako basic, za každý bílý kámen body přidává a za černý kámen body ubírá. Za dámu dává 5 bodů, ale ohodnocení obyčejného kamene je složitější. V základu dá za kámen 2 body a k tomu přidá hodnotu danou následujícím výrazem $(p-1)^3+1$, kde

$$p = \frac{\textit{řádka}}{(\textit{celkový počet řádek} - 1)}$$

p vyjadřuje pozici na desce. Pokud je kámen úplně vzadu, tak $p = 0$ a pokud je úplně vepředu na konci desky, tak $p = 1$. Hodnota p je vždy počítána z pohledu hráče, kterému kámen patří.

Hodnoty p jsou zvyšovány lineárně s každým posunutím kamene dopředu. Chceme motivovat posouvání kamenů vzadu, před kameny vepředu a k tomu slouží výraz $(p-1)^3+1$, který křivku udělá konkávní na intervalu $[0,1]$, ve kterém se nachází hodnoty p. Díky tomu posun kamene vzadu desky, přidá více bodů, než posun kamene vepředu desky.

3.1.6 Detekce remízy

Podle oficiálních pravidel, remíza nastává, když se oba hráči dohodnou, že ani jeden není schopen vyhrát, pokud oba neudělají žádnou chybu. Například když oběma zůstane jen dáma. Protože to není přesná definice, je obtížné rozhodnout, zda remíza nastala. Dva hráči řízení algoritmem minimax neudělají chybu, která by jim prohrála hru, když se mohou neustále vyhýbat soupeři. Budou takto hrát do nekonečna.

Z vlastních pozorování her mezi těmito virtuálními hráči, jsme zjistili, že ty nejdelší hry, podle českých pravidel, trvají přibližně 100 až 120 tahů. Hry, které trvaly déle, už neskončili. Proto jsme se rozhodli nastavit limit na počet tahů na 150. Všechny hry, které dosáhnou tohoto počtu tahů, skončí jako remíza.

U anglických pravidel, jsme limit tahů nastavili na 200. Hry trvají déle, protože dáma má omezenější pohyb a to prodlouží hru u konce.

3.1.7 Uživatelské rozhraní

Jedním z cílů práce je, aby uživatel měl možnost hrát proti počítači. K tomu je potřeba interaktivní deska, pomocí které se dají posouvat kameny. Způsob ovládání je popsán přímo ve hře (Stisknutím tlačítka „Help“).

Aby hráč nemohl udělat tah, který je v rozporu s pravidly, je potřeba si nejdříve vygenerovat všechny tahy, které může udělat v souladu s pravidly. Kód, který se stará o uživatelský vstup, pak pracuje s touhle informací.

3.2 Projekt Controller

Druhým je projekt Controller. Je to konzolová aplikace, která se stará o řízení simulací a evoluce. K tomu využívá projekt Draughts.

Tento projekt není koncipovaný jako interaktivní program. Všechny parametry evolucí a simulací se nastavují v kódu. Po zkompilování a spuštění udělá jen jednu věc, na kterou ji zrovna nastavíme a uživatel už nijak neinteraguje s programem za běhu, jen sleduje výsledky, které jsou zároveň zapisovány do souboru. Konkrétní popis programu nalezneme v příloze A.1.2.

3.2.1 Simulace her

Simulací rozumíme to, že vezmeme dva agenty a necháme je hrát proti sobě několik her a zaznamenáme výsledky. Toho využíváme při porovnání dvou agentů a v evoluci, při Hráči se po každé hře střídají v tom, kdo začíná, aby ani jeden z nich neměl výhodu.

Detaily o pouštění simulací popisujeme v příloze A.1.2.

3.2.2 Neuroevoluce v dámě

V sekci 2.3.2 jsme si popsali neuroevoluci teoreticky a tady se zaměříme na její implementaci.

Tvar sítí

Vyhneme se vyvíjení topologie sítě a její tvar předem stanovíme. Tvar nemůže být jakýkoliv. Výstupní vrstva se musí skládat z jednoho neuronu a velikost vstupní vrstvy je dána polovinou velikosti herní desky, jelikož se hraje pouze na tmavých polích. Jeden neuron odpovídá jednomu poli desky. Skryté vrstvy už mohou být jakékoliv.

Použití

Síť dostane na vstup hodnoty podobně, jako ohodnocení Basic. Každé tmavé pole má jeden odpovídající neuron, kterému je pak přiřazena hodnota podle toho, v jakém stavu pole je. Za kámen dostane hodnotu 1, za dámu 5 a za bílé to zůstane kladné, ale za černé je to -1 a -5.

Na výstupním neuronu se pak nachází hodnota, kterou přiřadíme hernímu stavu, který hodnotíme.

Inicializace sítí

Jedinci v první generaci populace jsou vygenerování náhodně. Všechny váhy v síti jsou nastaveny náhodně pomocí Normálního rozdělení se střední hodnotou 1 a rozptylem 0,2. Přechodová funkce u všech skrytých neuronů je Sigmoid funkce a u výstupních to je Lineární funkce.

Mutace

S určitou pravděpodobností projde jedinec mutací. Při tom je k některým vahám přičtena náhodná hodnota z normálního rozdělení se střední hodnotou 0 a rozptylem, který je dán jako parametrem „variace mutace“.

Křížení

Křížení probíhá po neuronech. Každý neuron má k sobě přiřazené váhy na neurony z předchozí vrstvy. Tyto váhy pak zůstávají s ním. Dvě sítě v jedné evoluci mají stejnou topologii, takže každý neuron v jedné síti má odpovídající neuron v druhé síti. Pro každý neuron, se určí s pravděpodobností 50% jestli se prohodí s jeho odpovídajícím neuronem v druhé síti. Tímto procesem dostaneme dvě nové sítě.

Selekce

Pro selekci jedinců využíváme ruletovou selekci. Ruletová selekce je, jak název napovídá, založená na točení ruletou. Každému jedinci je přiřazena výseč o velikosti dané poměrem:

$$\frac{\textit{jeho fitness}}{\textit{součet všech fitness}}.$$

Díky tomu jedinec, který má vyšší fitness má i větší výseč a tím i vyšší šanci na výběr. Při výběru se pak „zatočí ruletou“.

Fitness

K získání fitness hodnocení sítě, ji otestujeme v praxi. Necháme ji odehrát proti danému oponentovi. Jako oponenta zvolíme ohodnocení Basic. Dva agenti s algoritmem minimax s hloubkou 3 odehrají 100 her. Jeden agent využívá k ohodnocení stavů testovanou neuronovou síť a druhý agent využívá ohodnocení Basic. Počet výher prvního agenta bude fitness sítě.

4. Experimenty

V sekci 3.1.5 jsme si ukázali dvě heuristiky a také jsme si popsali jak použít a vyvinout neuronovou síť jako hodnotící funkci. V této kapitole porovnáme heuristiky mezi sebou, vyvineme několik sítí pomocí evolučních algoritmů a pak je otestujeme proti našim heuristikám.

K porovnání použijeme simulace her popsané v sekci 3.2.1 a výsledky simulací pak nalezneme ve výstupu programu a v souborech. Umístění souborů popisujeme v příloze A.2.

4.1 Hráč Rand

Bude se nám hodit hráč, který hraje jen náhodné tahy. Samozřejmě jen tahy v souladu s pravidly. Implementace takového agenta, je velmi jednoduchá. Vygenerujeme si všechny validní tahy, jak jsme si popsali v sekci 3.1.3 a jeden z nich náhodně vybereme.

Hráč Rand, reprezentuje velmi hloupého agenta a můžeme ho brát jako takovou referenci. Dobrý agent by ho měl ve většině her, ideálně ve všech, porazit.

4.2 Parametry soubojů

Parametry všech soubojů, jsou stejné. Abychom ohodnocení otestovali ve více hloubkách minimax stromu, jsou jednotlivé souboje mezi agenty rozděleny do šesti běhů. V prvním běhu je hloubka nastavena na 1. V druhém běhu na 2, a tak dále až do šestého běhu, kde je hloubka nastavena na 6. Oba agenti tak hrají pokaždé na stejné hloubce.

V každém běhu se hraje 1000 her. Oba agenti se každou hru střídají v tom, kdo začíná, takže oba hrají 500 her s bílými kameny a 500 her s černými kameny. Ani jeden nemá častěji výhodu začínajícího.

Součástí výstupu soubojů, jsou i data o tom, kolikrát každý hráč vyhrál za bílé a za černé kameny. Avšak žádná data nevykazují, že by v některém souboji měl někdo výhodu v nějaké barvě kamenů, respektive v tom, jestli začínal, nebo ne. Jinými slovy, všichni agenti měli počet výher za bílé kameny, přibližně stejný jako počet výher za černé kameny. Proto jsou v tabulkách s výsledky uvedena pouze data o celkovém počtu výher, proher a remíz. Kompletní výsledky pak najdeme v elektronické příloze práce.

4.3 Souboj heuristik

Zde si ukážeme, jak si vedou heuristické ohodnocení Basic a Progressive. Pro jednoduchost budeme-li mluvit o agentovi, který k rozhodování využívá algoritmus minimax a heuristiku Basic, nazveme ho hráč Basic. Stejně tak nazveme hráče Progressive.

4.3.1 Basic vs Rand

Začneme tím, že proti sobě postavíme hráče Basic a Rand. Basic je velmi jednoduchý typ ohodnocení, ale hra proti hráči Rand ukazuje, že má smysl. V tabulkách 4.1 a 4.2 vidíme, že krom prvního běhu, vyhrál téměř všechny hry v každém běhu a žádnou neremizoval.

Běh 1 musíme brát s rezervou, protože není moc vypovídající. Síla minimax prohledávání je v tom, že prozkoumá několik tahů dopředu a vybere takový, který vede k nejlepšímu stavu. Když vidí jen jeden tah dopředu, tak nemůže moc „plánovat“.

Běh	***1***	***2***	***3***	***4***	***5***	***6***
Basic	561	995	995	1000	1000	1000
Rand	439	5	5	0	0	0
Remíza	0	0	0	0	0	0

Tabulka 4.1: Tabulka výher. Basic vs Rand. Česká pravidla.

Běh	***1***	***2***	***3***	***4***	***5***	***6***
Basic	553	983	993	997	997	1000
Rand	444	11	5	1	2	0
Remíza	3	6	2	2	1	0

Tabulka 4.2: Tabulka výher. Basic proti Rand. Anglická pravidla.

4.3.2 Progressive vs Rand

Výsledky v tabulkách 4.5 a 4.6 ukazují, že hráč Progressive poráží hráče Rand bez obtíží. Zajímavým pozorováním je, že v prvním běhu má lepší výsledky, než hráč Basic proti hráči Rand. Naznačuje to, že Progressive je lepší ohodnocení.

Běh	***1***	***2***	***3***	***4***	***5***	***6***
Progressive	604	998	991	1000	999	1000
Rand	396	2	9	0	1	0
Remíza	0	0	0	0	0	0

Tabulka 4.3: Tabulka výher. Progressive vs Rand. Česká pravidla.

Běh	***1***	***2***	***3***	***4***	***5***	***6***
Progressive	625	961	998	1000	999	1000
Rand	365	35	2	0	1	0
Remíza	10	4	0	0	0	0

Tabulka 4.4: Tabulka výher. Progressive proti Rand. Anglická pravidla.

4.3.3 Basic vs Progressive

Souboj obou heuristik proti sobě je ta zajímavá část. Ohodnocení Progressive je vylepšené ohodnocení Basic, takže doufáme v lepší výsledky. Tabulky 4.5 a 4.6 potvrzují naší domněnku.

Rozdíl je nejvíce vidět v bězích 4 až 6, protože jak už víme, v těchto hloubkách je minimax efektivnější.

Nelze si nevšimnout, že agenti odehráli velké množství remíz. To je očekávané, vzhledem k tomu, že oba nedělají triviální chyby. Dostanou-li se do stavu, kdy mají oba proti sobě jen dámy, které nejsou nijak uvězněné, pak je remíza téměř jistá. Dámy mají volný pohyb a je těžší je nalákat do pasti.

Běh	***1***	***2***	***3***	***4***	***5***	***6***
Basic	427	257	354	210	293	206
Progressive	573	272	476	504	455	540
Remíza	0	471	170	286	252	254

Tabulka 4.5: Tabulka výher. Basic vs Progressive. Česká pravidla.

Běh	***1***	***2***	***3***	***4***	***5***	***6***
Basic	405	184	242	110	192	84
Progressive	592	182	617	561	595	409
Remíza	3	634	141	329	213	507

Tabulka 4.6: Tabulka výher. Basic proti Progressive. Anglická pravidla.

4.4 Důležitost hloubky stromu

Několikrát jsme zmínili, že minimax dosahuje lepších výsledků, pokud prochází strom hry do větší hloubky. Nyní to ověříme experimentálně. Vytvoříme hráče Basic3, který je shodný s hráčem Basic, akorát ve všech bězích prochází strom do hloubky 3. Poté ho necháme hrát s hráčem Basic. Oba se liší pouze v hloubce stromu.

V tabulkách 4.7 a 4.8 vidíme, že ve třetím běhu, mají oba hráči přibližně stejný počet výher, protože oba jsou ten běh identičtí. Basic3 pak vyhrál v prvních dvou bězích a prohrál v posledních třech, jak jsme předpokládali. Všimněme si větší rozdíl v hloubce se projeví na větším rozdílu v počtu výher.

Běh	***1***	***2***	***3***	***4***	***5***	***6***
Basic	6	102	410	744	787	823
Basic3	994	758	420	118	98	65
Remíza	0	140	170	138	115	112

Tabulka 4.7: Tabulka výher. Basic proti Basic3. Česká pravidla.

Běh	*1*	*2*	*3*	*4*	*5*	*6*
Basic	1	116	464	651	753	770
Basic3	997	622	408	149	95	71
Remíza	2	262	128	200	152	159

Tabulka 4.8: Tabulka výher. Basic proti Basic3. Anglická pravidla.

4.5 Evoluce

Přesuneme se od heuristik k neuronovým sítím a ukážeme si několik pokusů o vytrénování sítě pomocí evoluce. Pro odlišení pokusů jsme jim přiřadili unikátní identifikátor (zkráceně id), v podobě „evoluceXX“, kde XX je dvouciferné odlišující číslo. Ukážeme si tu výsledky tří evolucí, které přinesly zajímavé výsledky. Z každé evoluce jsme vzali nejlepšího jedince z poslední generace a nechali ho hrát proti Basic, stejně jako v sekci 4.3. Tohoto jedince, respektive neuronovou síť, jsme nazvali „NetworkXX“, kde XX odpovídá číslu v identifikátoru.

4.5.1 Parametry

Sítě jsme trénovali pouze pro česká pravidla hry. Všichni agenti jsou trénováni na hloubce 3, na stejné hloubce počítá i oponent. Tabulka 4.9 zobrazuje parametry evolucí. Do teď nezmíněný parametr je parametr „náhrada oponenta“. Cílem je zjistit vliv výběru oponenta na evoluci sítě. Pokud je jeho hodnota 1, pak se nic nemění a po celou dobu je oponent hráč Basic. Pokud ale zvolíme jinou hodnotu z intervalu $[0, 1)$, poté pokud nejlepší jedinec v generaci má poměr počtu výher ku celkovému počtu her vyšší, než je hodnota tohoto parametru, pak se stane oponentem pro další generace, dokud není nahrazen novým. Výjimkou je první generace, kde nejlepší jedinec je vybrán automaticky, ale jen pokud je hodnota parametru jiná, než 1.

Parametr	*evoluce01*	*evoluce02*	*evoluce06*	*evoluce07*
pr. mutace jedince	0,2	1	0,2	0,2
pr. mutace váhy	0,2	0,05	0,05	0,05
variace mutace	2	2	2	2
pr. křížení	0,2	0,2	0,2	0,2
velikost populace	30	30	30	20
náhrada oponenta	1	1	0,7	1
počet generací	20	20	20	100

Tabulka 4.9: Tabulka parametrů evolucí

4.5.2 evoluce01

Evoluce s identifikátorem „evoluce01“ je jedna těch, které se povedlo vyvinout síť, která hodnotí lépe, než Basic. V tabulce 4.10 vidíme, že i přesto, že byla síť trénovaná hrát na hloubce 3, tak se jí daří i v ostatních hloubkách.

Běh	***1***	***2***	***3***	***4***	***5***	***6***
Basic	388	328	372	281	274	265
Network01	612	408	455	516	499	538
Remíza	0	264	173	203	227	197

Tabulka 4.10: Tabulka výher. Basic proti Network01. Česká pravidla.

4.5.3 evoluce02

Zde máme ukázku neúspěšné evoluce. Mutace byla příliš častá. Tabulka 4.11 ukazuje, že síť prohrává téměř všechny hry.

Běh	***1***	***2***	***3***	***4***	***5***	***6***
Basic	482	966	872	920	804	938
Network02	517	19	25	23	31	29
Remíza	1	15	103	57	165	33

Tabulka 4.11: Tabulka výher. Basic proti Network02. Česká pravidla.

4.5.4 evoluce06

Myšlenka za tímto pokusem byla taková, že budeme postupně nahrazovat oponenta lepšími jedinci. Ale i přesto, že se oponent několikrát vyměnil, tak hra nejlepšího jedince z poslední generace proti Basic, nedopadla podle očekávání. Jak je vidět v tabulce 4.12, nedopadl sice tak špatně jako Network02, ale souboj prohrál ve všech bězích, krom běhu 1.

Běh	***1***	***2***	***3***	***4***	***5***	***6***
Basic	469	827	623	581	510	529
Network06	531	90	201	245	230	320
Remíza	0	83	176	174	260	151

Tabulka 4.12: Tabulka výher. Basic proti Network06. Česká pravidla.

4.5.5 evoluce07

Evoluci07 je podobná jako evoluce01, akorát jsme nechali běžet 100 generací, ve snaze zjistit, jestli to pomůže k vyvinutí lepší sítě. V tabulce 4.13 vidíme, že Network07 vyhrává proti Basic, ale není to nijak velký rozdíl oproti hře Network01 vs Basic.

Běh	***1***	***2***	***3***	***4***	***5***	***6***
Basic	437	314	321	272	272	249
Network07	563	394	472	540	504	570
Remíza	0	292	207	188	224	181

Tabulka 4.13: Tabulka výher. Basic proti Network07. Česká pravidla.

4.6 Turnaj

Nakonec nám zbývá určit tu nejlepší heuristiku, nebo síť. K tomu jsem udělali turnaj, kde hrál každý s každým. Turnaje se účastnili hráči Basic, Progressive, Network01 a Network07. Všichni spolu hráli 1000 her na hloubce 4.

Tabulka 4.14 zobrazuje výsledky turnaje. Zobrazuje počet výher hráče na řádku, proti hráči ve sloupci. Počet remíz není zobrazen, kvůli stylu tabulky, avšak kdybychom ho chtěli znát, můžeme ho dopočítat tak, že odečteme počet výher obou hráčů od 1000.

	Basic	***Progressive***	***Network01***	***Network07***
Basic	-	237	276	285
Progressive	512	-	357	407
Network01	525	334	-	694
Network07	536	289	180	-

Tabulka 4.14: Tabulka výher v turnaji mezi heuristiky a nejlepšími sítěmi.

Pro vyhodnocení turnaje každý hráč dostane 2 body za výhru, 1 bod za remízu a žádný bod za prohru. Celkové skóre pak dostaneme sečtením bodů za všechny odehrané hry. Výsledné bodování, včetně počtu výher, remíz a proher je vidět v tabulce 4.15.

	Výhra	***Remíza***	***Prohra***	***Skóre***
Basic	798	629	1573	**2225**
Progressive	1276	864	860	**3416**
Network01	1553	634	813	**3740**
Network07	1005	609	1386	**2619**

Tabulka 4.15: Tabulka získaných bodů za všechny odehrané hry.

Z výsledků je vidět, že nejlépe si vedla síť Network01 těsně následovaná heuristikou Progressive. Naopak nejhůře dopadla základní heuristika Basic, která bere v potaz pouze počet kamenů na herní desce. Je tedy patrné, že má smysl používat chytřejší heuristiky pro ohodnocování stavů hry.

Závěr

Uvedli jsme co je to Dáma, vysvětlili obecná pravidla a dvě konkrétní verze pravidel – Česká a Anglická. Pak jsme si uvedli algoritmus minimax s omezenou hloubkou. Udělali jsme ho efektivnější pomocí alfa-beta ořezávání a pamatováním si prošlých stavů. Vytvořili jsme si dvě heuristické ohodnocení herní desky, které minimax s omezenou hloubkou využívá – Basic a Progressive.

Basic je jednoduché ohodnocení, které pouze počítá počet kamenů, zatímco Progressive je o něco chytřejší heuristika, která pracuje i s pozicí kamenů.

Ukázali jsme si jak tyto heuristická ohodnocení nahradit neuronovou sítí. Vysvětlili jsme si co jsou to evoluční algoritmy a ukázali si, jak pomocí nich vyvinout neuronovou síť, schopnou ohodnotit herní desku.

V poslední kapitole 4 jsme experimentálně hodnotili všechny heuristiky a sítě. Porovnali jsme mezi sebou heuristiky Basic a Progressive, kde jsme si potvrdili, že Progressive je lepší. Ukázali si, že větší hloubka minimax prohledávání pracuje lépe, čímž jsme si dokázali to, že je lepší uvažovat co nejvíce tahů dopředu. Nakonec jsme udělali Turnaj, ve kterém jsme nechali hráče Basic, Progressive, Network01 a Network07 hrát každý s každým. Zjistili jsme, že si celkově nejlépe vedla síť Network01.

Možnosti rozšíření práce

Framework je vytvořen tak, že není náročné ho rozšiřovat o nové prvky. Jednou možností rozšíření, může být implementace dalších pravidel dámy, implementace jiného algoritmu, než je minimax nebo implementací dalších heuristik.

Seznam použité literatury

Bosboom, J., Congero, S., Demaine, E. D., Demaine, M. L. a Lynch, J. (2018). Losing at checkers is hard. *CoRR*, **abs/1806.05657**.

Campbell, M. a Marsland, T. A. (1983). A comparison of minimax tree search algorithms. *Artif. Intell.*, **20**(4), 347–367. doi: 10.1016/0004-3702(83)90001-2.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Prentice Hall.

Knuth, D. E. a Moore, R. W. (1975). An analysis of alpha-beta pruning. *Artif. Intell.*, **6**(4), 293–326. doi: 10.1016/0004-3702(75)90019-3.

Russell, S. J. a Norvig, P. (2010). *Artificial Intelligence - A Modern Approach, Third International Edition*. Pearson Education. ISBN 978-0-13-207148-2.

Schaeffer, J., Burch, N., Björnsson, Y., Kishimoto, A., Müller, M., Lake, R., Lu, P. a Sutphen, S. (2007). Checkers is solved. *science*, **317** (5844), 1518–1522.

Sharma, S. (2017). Activation functions in neural networks. *Towards Data Science*, **6**.

Sloss, A. N. a Gustafson, S. (2019). 2019 evolutionary algorithms review. In Banzhaf, W., Goodman, E. D., Sheneman, L., Trujillo, L. a Worzel, B., editors, *Genetic Programming Theory and Practice XVII [GPTP 2019, Michigan State University, East Lansing, Michigan, USA, May 16-19, 2019]*, pages 307–344. Springer. doi: 10.1007/978-3-030-39958-0_16.

Soltoggio, A., Stanley, K. O. a Risi, S. (2018). Born to learn: the inspiration, progress, and future of evolved plastic artificial neural networks. *Neural Networks*, **108**, 48–67.

Stanley, K., Clune, J., Lehman, J. a Miikkulainen, R. (2019). Designing neural networks through neuroevolution. *Nature Machine Intelligence*, **1**. doi: 10.1038/s42256-018-0006-z.

van Horssen, J. (2018). Complexity of checkers and draughts on different board sizes. *J. Int. Comput. Games Assoc.*, **40**(4), 341–352. doi: 10.3233/ICG-190077.

Vikhar, P. A. (2016). Evolutionary algorithms: A critical review and its future prospects. In *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, pages 261–265.

Wikipedia contributors (2020a). Alpha–beta pruning. URL `https://en.wikipedia.org/wiki/Alpha-beta_pruning`.

Wikipedia contributors (2020b). Minimax. URL `https://en.wikipedia.org/wiki/Minimax`.

Seznam obrázků

Seznam tabulek

A. Přílohy

Příloze se nachází zdrojové kódy obou projektů Draughts a Controller a výstupy experimentů. Programy jsou napsány v jazyce C# s .NET Framework 4.8.

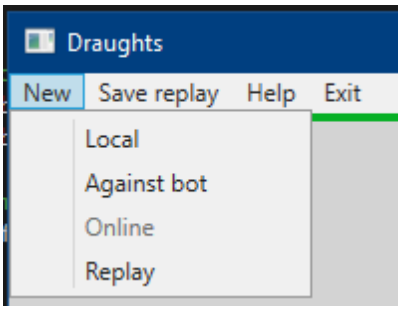
A.1 Manipulace s programy

V této sekci si stručně popíšeme jak používat oba programy.

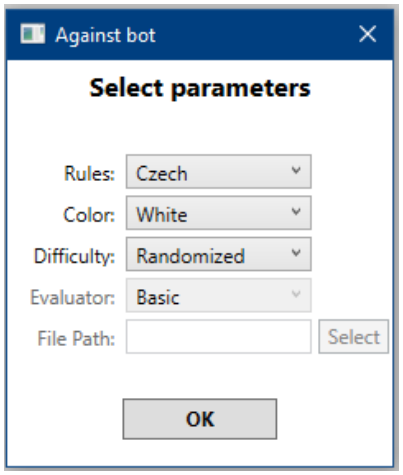
A.1.1 Draughts

Projekt Draughts je okenní aplikace typu WPF (Windows Presentation Foundation). Je opatřena jednoduchým rozhraním, ve kterém uživatel může hrát proti všem agentům, popsaným a implementovaným v této práci.

Na obrázku A.1 vidíme menu, ve kterém si zvolíme jaký typ hry hrát. Na výběr máme hru dvou hráčů (Local), proti počítači (Against bot), nebo přehrání uloženého záznamu hry (Replay).



Obrázek A.1: Obrázek menu v herním rozhraní.



Obrázek A.2: Obrázek menu s výběrem agentů.

Po vybrání hry proti počítači, vyskočí okno, ve kterém zvolíme pravidla hry, barvu kamenů, za které chceme hrát a protihráče (Obrázek A.1). Výběr protihráče je rozdělen do 2-3 částí. Nejprve vybereme obtížnost. Randomized je hráč Rand, který dělá pouze náhodné tahy. Zbylé obtížnosti jsou algoritmy minimax, s různou hloubkou prohledávání. Easy (hloubka 3), Medium (hloubka 6) a Hard (hloubka 9).

Při vybrání obtížnosti, která reprezentuje minimax algoritmus se nám odemkne možnost vybrat ohodnocující metodu – Basic, Progressive a Neurová síť (NeuralNetwork). V posledním případě se odemkne poslední vstupní pole, do kterého musíme zadat cestu k souboru s příponou *.nn*, ve kterém je serializovaná síť. Takový soubor je jeden z výstupů evolucí.

Po dohrání hry, můžeme stisknout tlačítko „Save replay“, kterým uložíme záznam do nového souboru s připonou *.drep*.

A.1.2 Controller

V projektu Controller spouštíme simulace her a evoluce sítí. Pro spuštění simulací, nebo evolucí je potřeba upravit třídu Program, zkompilovat a spustit. K tomu slouží metody *RunSimulation()* a *TrainEvA()*, které jsou zavolány z hlavní metody Main. Volána je vždy jen jedna z nich a volání druhé metody je zakomentováno.

Simulace

Metoda *RunSimulation()* je komentáři rozdělena do sekcí. Ne všechny sekce je potřeba upravovat.

V sekci „Setup bots“ jsou nastaveni všichni hráči. V sekci „Choose bots“ jsou pak vybráni dva, kteří proti sobě budou hrát. Následuje sekce „Setup game“, kde nastavíme pravidla a počet her.

Klíčová sekce „Run“ obsahuje vnořenou metodu „run(runID, depth)“, která pouští běh s danou hloubkou. V této sekci také voláme metodu run přímo, nebo v cyklu.

Evoluce

V metodě *TrainEvA()* jsou specifikované parametry evoluce a spuštěn běh evoluce. Pro úpravu parametrů stačí přepsat zadané hodnoty.

A.2 Výstupy experimentů

Projekt Controller je naprogramován k tomu, aby sám vypisoval výsledky do souborů.

A.2.1 Simulace

Soubory s výstupy simulací nalezneme ve složce *local/sim*. Nalezneme tam několik souborů, které jsou pojmenované podle toho, kdo spolu hrál. Například v souboru *minimax_basic_vs_minimax_progressive.txt* jsou všechny simulace mezi hráči Basic a Progressive.

A.2.2 Evoluce

Ve složce *local/eva* nalezneme složky, pojmenované identifikátory evolucí. V nich se nachází data o každé evoluci.

V souboru *settings.txt* jsou parametry evoluce. V souboru *log.txt* jsou výsledky simulací pro fitness ohodnocení sítí. A dále tam nalezneme soubory s názvy ve tvaru *genA_netB.nn*. V každém z nich se nachází serializovaná neuronová síť. *A* nahrazuje číslo generace a *B* nahrazuje pořadové číslo sítě v generaci.