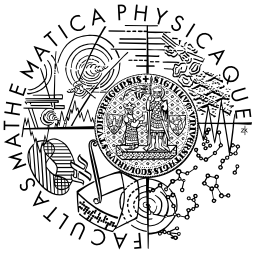


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Adam Šmelko

Vytváranie herných stratégií pre hru PuppetWars pomocou neuroevolúcie

Katedra teoretické informatiky a matematické logiky

Vedoucí bakalářské práce: Mgr. Martin Pilát, Ph.D.

Studijní program: Informatika

Studijní obor: Softwarové a datové inženýrství

Praha 2018

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 ..... dne ..... Podpis autora

Na tomto mieste chcem poďakovať svojmu vedúcemu bakalárskej práce, Mgr. Martinovi Pilátovi, PhD., za výborné rady a príjemné slová pri písaní tejto práce.

Název práce: Vytváranie herných stratégií pre hru PuppetWars pomocou neuroevolúcie

Autor: Adam Šmelko

Katedra: Katedra teoretické informatiky a matematické logiky

Vedoucí bakalářské práce: Mgr. Martin Pilát, Ph.D., Katedra teoretické informatiky a matematické logiky

Abstrakt: V posledných rokoch nastal v hernom priemysle rozmach. Pre udržanie konkurencieschopnosti sú herné spoločnosti nútené vyvíjať stále viac príťažlivé počítačové hry, čo implikuje i prítomnosť čo najvernejšej umelej inteligencie ovládajúcej herné prvky, na čo sa naša práca zameriava. Implementovali sme jednoduchú 2D programovaciu hru, na ktorej sme predviedli sadu pokusov učiac umelú inteligenciu v nej, snažiac sa vytvoriť stratégie konkurujúce tým ľudským. Preskúmali sme niekoľko variácií učenia pomocou evolučnej stratégie aplikovanej na neurónové siete a vytvorili sme herné postavičky hodné bytia rovnocenným protivníkom užívateľom hry.

Klíčová slova: počítačová hra, spätnoväzbové učenie, umelá inteligencia

Title: Creating the Game Strategies for PuppetWars using Neuroevolution

Author: Adam Šmelko

Department: Department od Theoretical Computer Science and Mathematical Logic

Supervisor: Mgr. Martin Pilát, Ph.D., Department od Theoretical Computer Science and Mathematical Logic

Abstract: In recent years the gaming industry has been on increase. In order to maintain competitiveness gaming companies are required to develop more and more compelling computer games what implies the presence of the very responsive artificial intelligence controlling the game elements, on which our work focuses. We have implemented a simple 2D programming game where we have experimented with the artificial intelligence in it trying to create a strategy beeing able to compete with human. We have explored several variations of learning through the evolutionary strategy applied to neural networks and we have created game characters worthy of being an equal opponent to the game user.

Keywords: computer games, reinforcement learning, artificial inteligence

Obsah

Úvod

Umelá inteligencia má v dnešnej dobe nespočetného využitia. Od pomoci ľuďom v medicínskych diagnózach cez finančnú správu v elektronických aukciách až po zábavu v počítačových hrách. My sa v tejto práci zameriame práve na jej hoci zábavný, no nie triviálny podiel v už spomínaných počítačových hrách.

Práve herný priemysel zažíva v posledných rokoch enormný rozkvet, celkové zisky dosahujúc obrovských súm. Rastúca výpočtová sila a nízka cena osobných počítačov vydláždila cestu zabezpečujúcu prístup počítačových hier masám ľudí, širokej verejnosti. To umožnilo oblasti počítačových hier zažiť revolúciu a ich popularita medzi verejnosťou neustále stúpa. Tento trend nasýtil fakt, že sprvu nemotorné a málo príťažlivé hry sa stali hrami, ktoré sú zábavou, pôžitkom, skvelým strávením voľného času a niekedy i životným štýlom.

Ako ale trend „hier ako voľného času“ rastie, užívatelia počítačových hier, inak zvaní *hráči*, sú prirodzene zavalení kvantom herných titulov saturujúc ich neustále viac náročné požiadavky. Každým dňom baží hráč po väčšom zážitku, akcií, vernosti hranej hry. Čo najlepšie uspokojenie týchto potrieb prináša do herného priemyslu predmet, ktorý mu bol na počiatku neznámy, a to predmet konkurencie. Tvorcovia hier sú nútení prichádzať s novými, nevídanými nápadmi, ich výtvory nech sú čo najživšie, najreálnejšie, najautentickejšie.

Ako príklad bude určite dostatočný údiv hráča bojujúceho v strieľacej hre, ktorého porazí protivník ovládaný počítačom, pretože sa nespráva ako robot, ale ukazuje známky ľudského premýšľania, zaťahujúc hráča hlbšie do hry. A práve tu vidíme dôležitosť umelej inteligencie v hrách. Dopomáhajú hráčovi prežiť autentickejší zážitok z hrania vyúsťujúc vo väčšom profite herných spoločností dobre si uvedomujúcich tento fenomén.

Ciele tejto práce sú nasledovné:

- Vytvoriť hru, programovaciu 2D strieľačku, v ktorej sa užívateľ snaží písaním umelej inteligencie poraziť počítačom ovládaných nepriateľov.
- Snažiť sa v práve spomenutej hre vycvičiť nepriateľov tak, aby sa javili ako rovnocenní súperi užívateľom, alebo ich predčili.

Snažiac sa docieliť vytýčené ciele, vyskúšame si prácu s neurónovou sieťou ako nástrojom na reprezentáciu stratégie herných postavičiek (našich nepriateľov) a pokúsime sa preskúmať a následne porovnať rôzne spôsoby učenia neurónových sietí založených na jednej z foriem optimalizačných algoritmov, a síce Evolučných stratégiách.

Štruktúra práce bude dodržiavať ideu prvotného zoznámenia sa s problematikou programovacích hier a umelej inteligencií používanej v nich. Ďalej si vysvetlíme dôležité prvky spomenutej hry a pojmy, ktoré sú nutné na následné pochopenie experimentov vykonávaných pri učení herných postavičiek. Na záver si rôzne metódy učenia porovnáme a zhrnieme výsledky.

V prvej kapitole si ukážeme súvisiace práce, konkrétne hru vytvárajúcu základy pre veľký rozsah programovacích hier a tiež predstavíme niekoľko spôsobov, ktorými sa uberali pri učení umelej inteligencie iní.

V druhej kapitole sa zoznámime s programovacou hrou zvanou *PuppetWars*, ktorá nám poslúži na skúmanie a testovanie učenia umelej inteligencie. Vysvetlíme si základné pravidlá hry a zásady pri programovaní vlastnej umelej inteligencie.

Tretia kapitola bude venovaná pochopeniu termínov ako *evolučné programovanie* a *neurónové siete*, ktoré sa budú hojne používať v ďalších častiach práce a využijeme ich pri vytváraní a učení umelej inteligencie.

Štvrtá kapitola sa zameria na problémy spojené s evolúciou umelej inteligencie v hre *PuppetWars* a detailne si popíšeme parametre s evolúciou spojené.

V piatej kapitole si predvedieme experimenty zamerané na učenie neurónových sietí. Vyskúšame prácu evolučných stratégii použijúc ich na či už populárnych viacvrstvových neurónových sieťach, tak i na menej známom modely ako sú rekurentné neurónové siete.

1. Súvisiace práce

V tejto kapitole si priblížime programovaciu hru, ktorá dala vznik hre *PuppetWars*, teda hre použitej v tejto práci. Ďalej si priblížime niektoré zaujímavé spôsoby učenia umelej inteligencie.

1.1 Programovacie hry

Tento možno trochu vágny termín, *programovacia hra*, definuje hry, ktoré sa môžu líšiť žánrom, ovládaním, či spracovaním, no majú jednu spoločnú vlastnosť. A síce nás hra núti programovať k dosiahnutiu ultimátneho cieľa hry. Typicky programovaním v týchto hrách ovládame bojovníka alebo bábku, všeobecne objekt, ktorý je vypustený do herného prostredia. To ako si v hre počína sa priamo odzrkadľuje na kvalite napísaného kódu.

Základné prvky, ktoré má veľká väčšina programovacích hier sú:

- *Programovací jazyk* so sadou inštrukcií ovládajúci programovaný objekt.
- *Prekladač*, alebo (typickejšie) *interpret* tohto jazyka.
- *Herné prostredie* so sadou pravidiel, zabezpečujúce požadovaný chod hry.

V súčasnej dobe existuje mnoho programovacích hier. Ich rozmach ale zapríčinil titul *Core Wars* [1]. Od tejto hry sa odvíjala väčšina programovacích hier, preto sa im aj zvykne hovoriť *„corewars-like“ hry*. Ako iným hrám, Core Wars bola inšpiráciou aj k vzniku hre, ktorú používame v tejto práci, preto je len správne tento titul predstaviť.

1.1.1 Core Wars

Core Wars, parafrázujúc [2], je *programovacia hra*, v ktorej dva alebo viac programov beží na simulovanom počítači s cieľom zneškodniť ostatné programy a prežiť ako najdlhšie môže. Známi ako *Bojovníci*, programy sú písané v asemblérskom jazyku zvanom *Redcode*.

Prostredie, v ktorom programy medzi sebou bojujú je tzv. „pamäť“. Pamäť je konečná postupnosť ďalej nedeliteľných jednotiek do ktorých sa zmestí po jednej inštrukcii. Je cyklická s iba relatívnym adresovaním (voči inštrukciám programov).

Programy sa skladajú z postupnosti inštrukcií[1]. Na začiatku hry je každý program načítaný do pamäti (teda s celou jeho postupnosťou inštrukcií) na náhodné miesto. Striedajúc sa, každý program vykoná jednu svoju inštrukciu a posunie inštrukčný pointer o pamäťovú jednotku vpred[2]. Programy sa snažia zlikvidovať protivníkov prepísaním nimi vykonávaných inštrukcií na neplatné. Ak program vykoná neplatnú inštrukciu, zomiera. Takto sa programy snažia prežiť, kým ostanú ako posledné ovládnuc celú pamäť.

[1] V základnej verzií Redcode je 8 rôznych inštrukcií. Najnovšie štandardy ich majú 16.

[2] Vpred sa pointer posunie samozrejme len za predpokladu, že nenarazil na inštrukciu skoku.

Imp

Prvý a najjednoduchší, no nie nedôležitý program sa nazýva *Imp*. Skladá sa len z jedinej inštrukcie:

```
MOV 0, 1
```

`MOV` kopíruje inštrukciu. Vysvetľujúc jeho chovanie, *Imp* vykoná svoj krok skopírovaním inštrukcie na adrese 0 (vďaka relatívnemu adresovaniu je to inštrukcia práve vykonávaná) na adresu o 1 ďalej. Následne posunie svoj pointer tiež o adresu ďalej, na ktorej je práve skopírovaná inštrukcia. *Imp* teda zahlcuje pamäť svojím telom, potenciálne prepisujúc inštrukcie iných programov. Keďže negeneruje neplatné inštrukcie (`DAT`), nedokáže nepriateľov zabiť, prinúti ich len vykonávať jeho inštrukciu, meniac každý program na ktorý narazí na *Impa*.

Tento formát hier, *programovacie hry*, nás zaujala natoľko, že sme sa rozhodli vytvoriť vlastnú hru — hoci do iného prostredia zasadenú, ale v nemalej miere inšpirovanú hrou *Core Wars* — ktorá dala počiatok tejto práci.

1.2 Učenie umelej inteligencie v hrách

Prvé náznaky umelej inteligencie boli k nahliadnutiu už v antike, kde podla mýtu grécky boh ohňa, zbrojár bohov Héfaistos vytvoril inteligentného robota menom Talós. Tento medený muž mal za úlohu tri krát za deň obísť pobrežia Kréty a na lode nežiaducich cudzincov hádzať balvany. A ak sa i nejaký votrelec dostal na súš, Talós nechal svoje medené telo rozžeraviť a nepriateľa zahubil pekelným objatím. Parafrázujúc Pamelu McCorduckovú, umelá inteligencia začala ako staroveké prianie ukuť bohov [3].

Nižšie si ukážeme práce ktoré zaujímavým spôsobom používajú neurónové siete a evolučné algoritmy v hrách.

V práci [4] sú použité neurónové siete na hranie hry GO. Autori použili špecifický druh evolúcie. Tu sa nezameriavali na evolúciu samotných neurónových sietí, tento problém radšej rozdelili na evolúciu populácie *neurónov* a populácie *plánov* (vraviace, aké neuróny treba skombinovať na vznik siete). Týmto spôsobom dekomponovali prehľadávaný priestor prehlasujúc, že táto metóda nebude konvergovať k suboptimálnym riešeniam tak ľahko ako iné.

Práca [5] pojednáva o vytvorení hráča do hry Othello pomocou evolúcie neurónových sietí. Autori zvolili zaujímavý spôsob zakódovania siete, tzv. *Marker-based encoding* inšpirovaný štruktúrou DNA. Sieť kódovali do postupnosti celých čísel používajúc niektoré čísla ako významové (markery), vymedzujúc v postupnosti bloky ktoré reprezentovali konkrétne neuróny v sieti. Tento postup prichádza s výhodami ako možnosť súčasnej evolúcie topológie a váh siete a tiež, že samotná interpretácia alely nie je závislá na jej pozícií v chromozóme.

Jednou z úspešných a široko používaných metód v neuroevolúcií je *NEAT* [6] (skratka pre NeuroEvolution with Augmenting Topology). NEAT je zameraná na evolúciu sietí bez akejkoľvek apriórnej znalosti o jej topológii. Je to implikácia jej troch hlavných ideí. Prvá, evolúcia začína od neurónovej siete s najjednoduchšou topológiou, typicky so žiadnymi skrytými vrstvami. Ďalšia, NEAT si zaznamenáva, ak v evolúcii došlo k vzniku siete s novou topológiou. Toto využíva pre

jednoduché zistenie podobnosti štruktúry sietí, čo je užitočné pri rekombinácii. Nakoniec, NEAT chráni novovzniknuté topológie pred skorým zánikom, dávajúc im čas na evolúciu. Táto metóda bola použitá v práci [7] na učenie kontroly áut v pretekárskych hrách, dosahujúca cielené výkony.

2. Hra

Súčasťou tejto práce bolo vytvorenie hry, 2D strieľačky, v ktorej je možné herné postavičky svojvoľne naprogramovať, následne ich vypustiť do hry a sledovať, ktorá postavička má stratégiu tak unikátnu, že v bojovom poli ostane ako jediná.

V tejto kapitole sa zameriame práve na túto časť práce. Vysvetlíme si dôležité prvky hry nutné na pochopenie ako hry samotnej, tak i nasledujúcich častí práce. Popíšeme si základné pravidlá hry a predstavíme si postavičky a objekty objavujúce sa v nej.

2.1 PuppetWars

Hra *PuppetWars* je programovacia 2D strieľačka. Je hraná v mriežkovej ploche o rozmeroch $m \times n$ vytvárajúcej bludisko. V tomto bludisku stoja proti sebe postavičky – *hráčske figúrky*. Postavičky v hre bojujú ako jednotlivci alebo v tímoch. Na boj používajú strelné zbrane snažiac sa eliminovať nepriateľa, teda postavičku v tíme inom ako je ten ich. Naviac sa v bludisku nachádzajú miesta, na ktorých sa objavujú vylepšenia, poskytujúce bonusy, alebo rôzne zbrane. Hra skončí, keď v bludisku ostane jediný — teda i posledný — tím hráčov, respektíve posledný individuálny hráč.

Čo ale rozlišuje túto hru od iných „generických 2D strieľačiek“ je fakt, že pohyb a akcie postavičiek nie sú užívateľom hry priamo ovládané pomocou kláves, či myši. Ak má užívateľ záujem vyhrať, je nútený napísať zdrojový kód v jazyku *JavaScript*, ktorý bude reprezentovať stratégiu herných postavičiek. Táto **stratégia** ich bude následne riadiť v bojovom poli a vďaka nej potenciálne prídu k zaslúženému víťazstvu.

2.2 Puppets

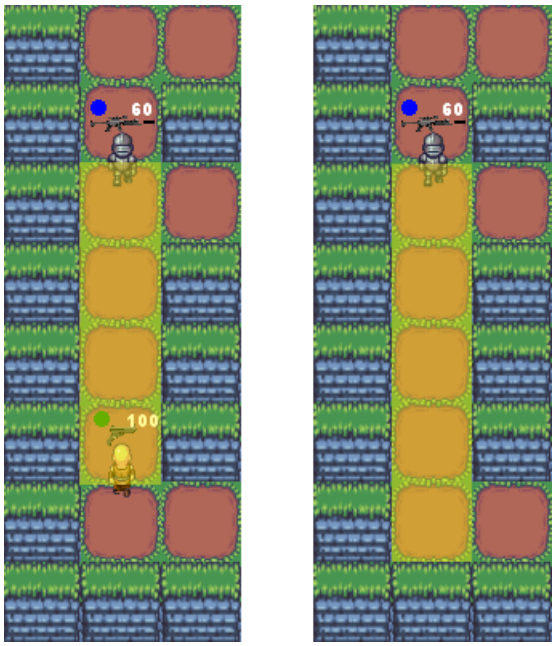
Názov hry sa skladá z dvoch anglických slov **Puppet** a **Wars**. Hoci použitie druhého slova v názve je kvôli žánru hry pochopiteľné, prvé si zasluhuje objasnenie. Toto slovo odkazuje na herné postavičky. Symbolizuje ich nemožnosť rozhodovania nad ich činmi, teda ako bábky[1] sú ťahané za nitky, tak postavičky sú ovládané im pridelenou **stratégiou**.

Bábky sú významnou súčasťou hry. Teda je dôležité vedieť ich vlastnosti a zákonitosti, ktorými sa riadia.

- **Pohyb** — Bábka sa vie pohybovať štyrmi smermi, a síce *hore, dole, doprava a doľava.*
- **Výhľad** — Inými slovami *okolie*, ktoré postavička v danom momente vidí. Registruje objekty na políčkach priamo pred ňou až po prvú stenu, kde sa jej výhľad končí. Ten sa ale môže naviac znížiť, ak na jednom zo spomenutých políčok stojí iná postavička. Za ňou výhľad opäť končí. Viď obr. 2.1.

[1] Bábka, z anglického slova *puppet.*

- **Zdravie** — Je reprezentované celým kladným číslom. Vieme rozlíšiť *zdravie* **aktuálne** a **maximálne**, pričom platí, že *aktuálne zdravie* počas hry neprekročí *maximálne zdravie*. Ak bábku zasiahne strela, jej *aktuálne zdravie* sa zmenší. Dosiahnúc hodnoty menšie alebo rovné nule, bábka zomiera a v hre končí.
- **Rýchlosť** — Nariaďuje postavičke, ako rýchlo sa smie pohybovať v bojovom poli, teda akú vzdialenosť je schopná prejsť v jednom kroku hry.
- **Zbraň** — Nástroj, ktorým bábka bojuje. Viz 2.3.1 **Zbrane**.
- **Tím** — Každá bábka univerzálne patrí do nejakého tímu[2]. Bábky v rovnakom tíme bojujú za víťazstvo spoločne, neznamená to ale, že sa nevedia zraniť.



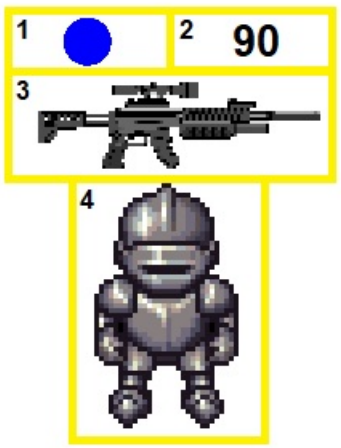
Obr. 2.1: Dva snímky simulácie hry predvádzajúce *voľný* (vpravo) a *zahataný* (vľavo) **výhľad** bábky (zvýraznený žltou farbou).

Základná verzia hry obsahuje tri typy postavičiek odlišujúce sa pomerom *zdravia* a *rýchlosti*. Sú to pomalý, no silný **Knight**, na opačnej strane rýchly ale slabý **Skeleton** a medzi nimi je kompromisný **Monk** (viď Tabuľka 2.1).

Bábka	**Rýchlosť**	**Zdravie**
Knight	1	140
Monk	1.5	100
Skeleton	2.5	80

Tabuľka 2.1: Líšiace sa vlastnosti základných druhov bábok.

[2]Jednotlivec je v tíme tiež, mimo neho ale už prázdnom.



Obr. 2.2: Snímka bábky zo simulácie hry s jej vlastnosťami zvýraznenými v žltých rámčekoch (1–*Tím*, 2–*Aktuálny život*, 3–*Zbraň*, 4–*Bábka*).

2.3 Objekty

V hre sa vyskytujú dva druhy objektov, ktoré môže bábka vziať a následne použiť. Sú to **zbrane** a **vylepšenia**. Zdieľajú spoločnú vlastnosť, a síce ich bábka dokáže vziať z na to určených miest.

2.3.1 Zbrane

Tieto objekty sú významnou súčasťou hry. Dovoľujú bábke zaútočiť na nepriateľa, zabrániť svojej smrti a potenciálne pomôcť k jej víťazstvu.

Dôležité vlastnosti zbraní sú tieto:

- **Sila** — Táto vlastnosť ukazuje, koľko bodov *života* bábke ubudne ak ju strela z danej zbrane zasiahne.
- **Rýchlosť streľby** — Vlastnosť, ktorá vraví, koľko herných *ťahov* sa musí od poslednej streľby vykonať pokiaľ bude zbraň schopná vystreliť opäť.

Definícia 1. *Majme zbraň Z so silou S a rýchlosťou streľby R. Potom pomer* $\boldsymbol{V = \frac{S}{R}}$ *nazveme* výkon *zbrane.*

V hre sú 3 základné zbrane zoradené podľa *výkonu*. Najnižšie v rebríčku sa nachádza **Pištoľ**, ďalej nasleduje **Puška** a na vrchole je **Brokovnica** s *výkonom* najväčším (viď Tabuľka 2.2, Obrázok 2.3).

Zbraň	**Sila**	**Rýchlosť streľby**
Pištoľ	10	24
Puška	8	12
Brokovnica	25	30

Tabuľka 2.2: Vlastnosti základných zbraní.

Obr. 2.3: Obrázky základných zbraní — (zľava) pištoľ, puška, brokovnica

2.3.2 Vylepšenia

Tieto typy objektov poskytujú na krátky čas rôzne bonusy bábke ktorá ich použije. Práve podľa toho aké bonusy sú bábke pridelené rozlišujeme tri druhy vylepšení (viď Obrázok 2.4):

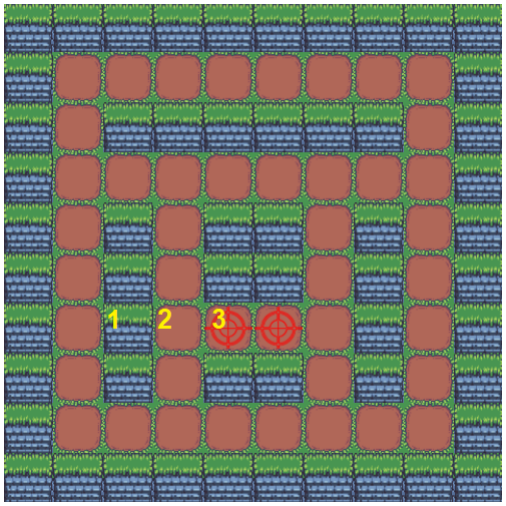
- **Vylepšenie zdravia** — Po použití bábke nastaví *Aktuálny život* na *Maximálny život*.
- **Vylepšenie rýchlosti** — Toto vylepšenie bábke dovolí pohybovať sa po bojovom poli rýchlejšie zdvojnásobením jej rýchlosti.
- **Vylepšenie výkonu zbrane** — Udelí bábke výhodu vo forme zdvojnásobenia *sily* jej zbrane.

Obr. 2.4: Obrázky vylepšení — (zľava) Vylepšenie zdravia, rýchlosti, výkonu zbrane

2.4 Prostredie

Hracia plocha hry je — ako už bolo spomenuté — mriežka políčok o veľkosti $m \times n$ formovaná ako bludisko. Hoci má hracia plocha charakter skôr diskrétny, bábky sa po nej takýmto charakterom neposúvajú a trvá im aj niekoľko krokov, kým prejdú jedno celé políčko. Spomínané políčka môžu byť vyplnené nasledujúcimi blokmi:

- **Cesta** — Po tomto bloku bábky smú chodiť. Slúži teda na vytváranie koridorov v bludisku hracej plochy.
- **Stena** — Na tento druh políčka bábka nedokáže stúpiť. Vymedzuje hranice medzi koridormi. Má ale vlastnosť typu *Život* a dá sa teda zničiť strelami zo zbraní bábok.
- **Zdroj** — S *cestou* zdieľa vlastnosť, že sa po nej dá chodiť. Má ale ešte ďalšiu vlastnosť, ktorá ho od *cesty* rozlišuje. A síce každý stanovený počet *kôl* sa na nej vytvorí (objaví) *objekt*. Podľa toho, aký je to druh *objektu* rozlišujeme **Zdroj vylepšení** a **Zdroj zbraní**. Každý takýto blok má predpísanú množinu *objektov*, ktoré vie vytvárať. Ak v hre nejaký *zdroj* vytvoril svoj *objekt*, bábka, ktorá na zdroji stojí, ho vie vziať. Vezmúc ho, zdroj opäť čaká stanovený počet *kôl* hry, kým nevytvorí ďalší.



Obr. 2.5: Snímka hracej plochy zo simulácie hry (1–*Stena*, 2–*Cesta*, 3–*Zdroj*).

2.5 Stratégia

Stratégia bábky je zodpovedná za každý jej krok v hre. Užívateľ ju vytvorí písaním zdrojového kódu v jazyku *JavaScript*.

Výber práve tohto programovacieho jazyka bol podložený potrebou mať jazyk, ktorý by mal *jednoduchú syntax*, kde by programátor vedel prepísať myšlienku na kód *rýchlo a bez námahy* a aby bol jazyk schopný byť *interpretovaný*. Spomínaný programovací jazyk všetky tieto vlastnosti spĺňa a je navyše vo verejnosti ohromne populárny, preto sme prišli s touto voľbou.

Keď sa užívateľ rozhodne začať písať stratégiu svojej bábky, začína s vygenerovaným kusom zdrojového kódu.

```
[1] function DoStep(puppet) {
[2]
[3] }
```

Hra prebieha v krokoch, *ťahoch*. V jednom hernom *ťahu* je každej bábke nariadené vykonať jeden krok, čo zodpovedá zavolaniu funkcie `DoStep`. Z toho nám vyplýva, že samotná **stratégia** bábky je práve táto funkcia.

Parameter `puppet` funkcie `DoStep` reprezentuje bábku ovládanú stratégiou. Spĺňa interface `IPuppet`, ktorý zaručuje na parametri volať tie dôležité funkcie, ktoré prikážu bábke vykonávať veci nápomocné k výhre na bojovom poli.

2.5.1 IPuppet

Tento interface spĺňa dôležitú rolu pri písaní stratégie bábok. Tvorí nám cestu k ovládaniu ich akcií. Obsahuje mnoho užitočných metód, ktoré dokážeme rozdeliť do dvoch skupín.

Metódy pozorovania

Tieto metódy umožňujú užívateľovi získať vnemy z prostredia bábky, či nejaké jej užitočné vlastnosti. Sú to napríklad[3]:

[3]Pre detailnejší popis interface viď užívateľskú dokumentáciu hry v priloženom CD.

- `CanGoForward()`, `IsCorridorRight()`, `IsCorridorLeft()` — Tieto metódy zisťujú možnosť pohybu bábky. Konkrétne, či dokáže ísť vpred, alebo či je naľavo, respektíve napravo od nej voľná cesta.

- `PlayerInSight()`, `ProjectileInSight()`, `WeaponSpawnInSight()` — Metódy priamo pracujúce s **výhľadom** bábky. Vracajú to, čo bábka vidí, alebo nevidí, vo svojom *výhľade*. Zisťuje, či vidí, po poradí, inú bábku, strelu zo zbrane, políčko s výskytom zbraní.

- `Health`, `MaxHealth`, `Weapon`, `Speed`, `Team` — Nakoniec sú v tejto skupine metódy na zistenie aktuálneho stavu na nie menej dôležitých vlastností bábky. Ich názvy činia vysvetlenie nepotrebným.

Hra dovoľuje v jednom kroku bábky volať neobmedzený počet metód z tejto skupiny[4].

Metódy akcií

Týchto metód je menšie množstvo, a to päť. Ich používanie je oproti 2.5.1 **Metódam pozorovania** prísne, a síce za jeden krok bábky sa môže vykonať práve jedna takáto metóda[5]. Zavolanie metódy tejto skupiny vo funkcii `DoStep` spôsobí okom pozorovateľné vykonanie akcie bábky, od čoho vznikol názov skupiny.

Sú tu tieto metódy:

- `GoForward()` — Bábka sa posunie vpred v závislosti od jej rýchlosti.

- `TurnLeft()`, `TurnRight()` — Prinúti bábku otočiť sa doľava, respektíve doprava.

- `Shoot()` — Bábka sa pokúsi vystreliť zo zbrane.

- `PickItem()` — Bábka sa pokúsi vziať objekt z políčka na ktorom stojí.

2.5.2 Príklad stratégie

Pre predstavenie ukážeme príklad jednoduchej stratégie. Bábka bude robiť tieto akcie (zoradené klesajúc podľa priority):

1. Ak vidíš bábku, strieľaj.

2. Ak stojíš na zdroji, vezmi objekt.

3. Pohni sa.

Kód tejto stratégie vyzerá nasledovne:

[4]Spomíname to pre vyzdvihnutie rozdielu medzi metódami druhej skupiny, viď 2.5.1 **Metódy akcií**.

[5]Ďalšie následné volanie takejto metódy bude hrou ignorované.

```
[1]  function DoStep(puppet) {
[2]    var player = puppet.PlayerInSight();
[3]    var weaponSpawn = puppet.WeaponSpawnInSight();
[4]    var powerUpSpawn = puppet.PowerUpSpawnInSight();
[5]    if(player != null)
[6]      puppet.Shoot();
[7]    else if (weaponSpawn != null || powerUpSpawn != null)
[8]      puppet.TakeItem();
[9]    else
[10]     move(puppet);
[11] }
[12] var turned = false;
[13] function move(puppet) {
[14]   if(turned) {
[15]     puppet.GoForward();
[16]     if(!puppet.IsCorridorLeft())
[17]       turned=false;
[18]   }
[19]   else  {
[20]     if(puppet.IsCorridorLeft()) {
[21]       turned = true;
[22]       puppet.TurnLeft();
[23]     }
[24]     else {
[25]       if(puppet.CanGoForward())
[26]         puppet.GoForward();
[27]       else
[28]         puppet.TurnRight();
[29]     }
[30]   }
[31] }
```

V riadkoch 2-4 získavame pomocou *metód pozorovania* neskôr potrebné informácie zisťujúce, čo má bábka vo *výhlade*. Následne v riadkoch 5-10 vykonávame už popísanú stratégiu. Funkcia `move` reprezentuje akciu „Pohni sa“, teda bábka kopíruje steny bludiska po jej pravej ruke.

3. Evolučné algoritmy a neurónové siete

V tejto kapitole sa budeme snažiť zoznámiť čitateľa s termínmi **Evolučný algoritmus** a **Neurónová sieť**. Cieľom je priniesť základnú vedomosť týchto pojmov, ktorá príde k úžitku pri ďalších kapitolách tejto práce.

Na začiatku kapitoly si priblížime biologickú motiváciu *Evolučných algoritmov*, kde sa neskôr presunieme na hlbšie detaily. Konkrétne si predvedieme jeden z modelov *Evolučných algoritmov* zvaný **Evolučná stratégia**. Rovnako začínajúc popíšeme i *Neurónové siete* a spomenieme niektoré z ich modifikácií, ako napríklad **Rekurentné neurónové siete**.

3.1 Evolučný algoritmus

Evolučný algoritmus je jednou z foriem optimalizačných algoritmov, pracujúci na princípe vyvíjania populácie riešení skrz opakovanú transformáciu. Len skutočnosť, že sa pracuje s populáciou riešení je významný rozdiel medzi evolučnými algoritmami a tradičnými optimalizačnými algoritmami. Ďalším dôležitým rozdielom je to, že operátory uskutočňujúce transformácie populácie sú inšpirované evolúciou v prírode [8]. Nie je teda prekvapujúce, že v oblasti evolučných algoritmov sa ako náhrada za množinu riešení používa termín *populácia* a jej prvkom sa hovorí *jedince*.

Formulujúc optimalizačný problém ako hľadanie globálneho extrému funkcie f na jej definičnom obore (tiež sa zvykne používať anglický termín *search space*, ďalej len prehľadávaný priestor), zadefinovali sme si všeobecný problém ktorým sa evolučné algoritmy vo veľkej miere zaoberajú.

Evolučné algoritmy sú založené na modeli prirodzenej, biologickej evolúcie, prvý krát sformulovanej Charlesom Darwinom [9]. *Darwinova teória evolúcie* vysvetľuje adaptívnu zmenu druhov princípom *prirodzeného výberu*. Ten uprednostňuje k prežitiu a ďalšej evolúcií tie druhy, ktoré sa dokážu najlepšie adaptovať podmienkam prostredia [10]. Nasledujúci text popisujúci základy evolučného algoritmu čerpáme z [11].

Reprezentácia jedinca

Biologickí, prirodzení jedinci sú v značnej miere obrazom svojej genetickej výbavy, **genotypu**[1]. Všetky informácie obsiahnuté v *chromozómoch*, nosičoch genetickej výbavy zakódovanej do molekuly DNA, predstavujú genotyp. To, aký má biologický jedinec genotyp je určené **génmi**, blokmi DNA. Tie kódujú istú základnú a ďalej nedeliteľnú informáciu o jedincovi, všetky súvisle tvoriaci jeho genotyp. Naviac, každý gén sa môže vyskytovať v rôznych variantách (napr. gén kódujúci farbu očí sa vyskytuje vo variante "modré", či "hnedé"). Tieto varianty sú nazývané **alely**.

[1] V biológií by sa tu nemalo zabudnúť na *fenotyp*, súhrn všetkých vonkajších znakov organizmu, no pre jednoduché uvedenie do problematiky o to nie je núdze.

V evolučných algoritmoch je táto idea do veľkej miery preberaná. Jedinci môžu byť napríklad tvorení jedným chromozómom, skladajúcim sa z pevnej postupnosti predom dohodnutých znakov, ktorý kóduje konkrétne riešenie optimalizačného problému. Tu by každá pozícia v postupnosti určovala *gén* a znak na nej by určoval *alelu*. Toto je jedna z najjednoduchších reprezentácií jedinca, o ktorej typoch si ešte povieme.

Konkurencieschopnosť

V prírode, aby jedinci prežili, potrebujú reagovať na svoje biologické potreby a charakter prostredia. Ak je jedinec hladný, musí si nájsť potravu. Ak je v nebezpečenstve, musí sa správne rozhodnúť, či utiecť alebo bojovať. Musí si vedieť nájsť úkryt. Prirodzene, niektoré jedince budú v niektorých oblastiach správania dokonalejšie, dominantnejšie. To im zaistí väčšiu šancu sa rozmnožovať — a preniesť svoj vzácny genetický materiál na ďalšiu generáciu — či už vďaka tomu, že nie dostatočne dobrých jedincov vytlačili z ich okolia, alebo že dokázali prekonať (teda i prežiť) nečakané zmeny prostredia.

V našej umelej reprezentácii by sme chceli taktiež ohodnotiť svojich umelých jedincov, nech vieme, ktorí si zaslúžia prežiť a poskytnúť svoju informáciu ďalšej generácií. Evolučné algoritmy preto pracujú s takzvanou **fitness**, teda *zdatnosťou*. Každý jedinec v populácií je ohodnotený zdatnosťou pomocou **fitness funkcie**, ktorej výstup je priamo závislý na jeho spôsobe riešenia optimalizačného problému. Hlavný cieľ zdatnosti jedinca je dať spätnú väzbu evolučnému algoritmu, nech vie, ktorý jedinec si zaslúži väčšiu šancu sa rozmnožiť a ktorý jedinec by mal mať zas väčšiu šancu byť z populácie odstránený. Príkladom zdatnosti môže byť počet nárazov do stien robota ovládaného evolučným algoritmom pri učení sa vyhýbaniu prekážkam, alebo suma peňazí získaná algoritmom ovládaným agentom v stávkovej hre.

Vývoj populácie

Populácia biologických jedincov sa postupom času vyvíja. Zdatné jedince sa rozmnožujú a menia charakter populácie, ideálne k lepšiemu. V evolučných algoritmoch prebieha takýto vývoj populácie, **evolúcia**, iteratívnou cestou, kde sa pri každej iterácií vytvorí nová populácia. Populáciu vytvorenú v i-tej iterácii nazveme i-tou **generáciou**. Typicky i-tá generácia vznikne za pomoci jedincov práve z generácie $i-1$. Evolučný algoritmus teda nedáva možnosť súčasnej existencií viac ako jednej generácie[2].

Hoci vieme, že v evolučných algoritmoch nastáva vývoj populácie iteratívne, ostáva nám zistiť, ako vznikne prvá generácia. Tu sa často používa označenie **nultá generácia** a typicky vzniká náhodným vygenerovaním jedincov.

Selekcia

Keď majú všetky jedince priradenú zdatnosť pomocou fitness funkcie, potrebujeme sa rozhodnúť, ktorí jedinci v populácií zostanú a stanú sá základom

[2]Tu sa už evolučné algoritmy vzďaľujú od prirodzených zákonitostí. Existuje mnoho prípadov, kde jedinec v prírode prežíva niekoľko generácií. Výnimkou tiež nie je možnosť sa rozmnožovať s jedincami z rôznych generácií.

pre vznik ďalšej generácie a ktorých dovolíme evolučnému algoritmu nahradiť. Táto úloha je zvaná **selekcia**. Hoci existuje mnoho selekčných operátorov a výber správneho je jeden z dôležitých krokov k úspešnému evolučnému algoritmu, ich ciele sú obdobné. Selekcia slúži na výber množiny jedincov z populácie a je riadená zdatnosťou jedinca, a to konkrétne preferuje výber najzdatnejších jedincov. Dôležité je ale nechať aj možnosť výberu takých jedincov, ktorým *fitness funkcia* nepriradila najlepšie hodnoty. A to práve pre zachovanie rozmanitosti a zabráneniu zmenšenia genetického fondu v populácií. Za zmienku stoja:

- **Ruletová selekcia** — Alebo aj *Selekcia úmerná zdatnosti*. Pri výbere k jedincov táto varianta pracuje postupne v k nezávislých krokoch, vyberajúc vždy po jednom jedincovi. V každom kroku vyberá z celej populácie a to v závislosti od pravdepodobnosti pridelenej každému jedincovi. Nech f_i je zdatnosť jedinca i a N je veľkosť populácie, tak pravdepodobnosť $p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$ je pravdepodobnosť výberu jedinca i. *Ruletová* sa jej hovorí, pretože keď si z pravdepodobností výberu jedincov celej populácie urobíme koláčový graf a predstavíme si ho ako ruletu s takým počtom a veľkosťou chlievikov aké sú výseky koláčového grafu, tak pri jej roztočení a hodení guľôčky jednoducho napodobňujeme vyššie popísanú pravdepodobnosť výberu.

- **Turnajová selekcia** — Táto varianta selekcie, na rozdiel od *Ruletovej*, nie je založená na konkurencii v celej populácii, ale len v jej podmnožine. Počet jedincov zvaný *veľkosť turnaja* sa vyberie uniformne náhodne a na tomto počte sa zvolí jeden najzdatnejší jedinec, teda víťaz turnaju[3].

Genetické operátory

Jedince vybraté selekciou evolučný algoritmus priamo neposiela do ďalšej generácie, takýto proces by uskutočnil evolúciu nemožnou. Preto sú na selekciou vybraných jedincoch aplikované **genetické operátory**. Tieto operátory transformujú populáciu modifikovaním jedincov. Ako vstup berú podmnožinu populácie (i jednoprvkovú) a typicky vrátia zmenených jedincov, ktorí už patria do ďalšej generácie. V jednej iterácií je často opakovane požívaných niekoľko genetických operátorov na vytvorenie novej generácie. To, ktorý sa v akej miere na vytváraní nových jedincov podieľa je určené operátoru priradenou pravdepodobnosťou. Poznáme tri základné operátory:

- **Kríženie** — Tento operátor kombinuje genetický materiál dvoch rodičov výmenou časti materiálu jedného rodiča s druhým. Operátor cieli nasledujúce. Majme populáciu jedincov, ktorých chromozómy obsahujú dva gény, každý riešiace istý navzájom nezávislý problém. Nech sa v populácií nachádzajú dva jedince. Jeden s variantou prvého génu perfektne riešiaci prvý problém, no s variantou druhého génu, ktorá nie je obstojná pri riešení druhého problému. Druhý jedinec majúci také varianty génu, nech, obrátene, dokáže vyriešiť druhý problém a v prvom neobstojí. Ak vezmeme týchto

[3]Vo všeobecnejšej variante tejto selekcie nemá najzdatnejší jedinec istú šancu zvíťaziť, selekcia dáva (typicky malú) šancu aj konkurenčným jedincom.

dvoch jedincov ako rodičov použijúc operátor kríženia, vznikne nám jedinec s unikátnym chromozómom riešiacim už oba problémy.

Tu môžeme nazrieť, prečo je dôležité pri selekcií dať šancu aj menej zdatným jedincom. Hoci si všeobecne nevedia poradiť s problémom tak dobre ako iní jedinci, môžu ale obsahovať vzácnu sekciu chromozómu, ktorá pri správnom krížení vytvorí veľmi zdatného jedinca.

Kríženie je do istej miery inšpirované biologickým *prekrížením*. Tento jav nastáva pri vzniku pohlavných buniek, kde telové bunky obsahujúce dvojicu chromozómov prekrížia tento pár a dajú vznik chromozómu s novým a potenciálne lepším genotypom. Je dôležité si ale všimnúť, že biologické *prekríženie* prebieha vrámci jedinca, pričom umelé kríženie kombinuje informáciu z rodičovských potomkov.

- **Mutácia** — Operuje len na jednom jedincovi a náhodne vytvára malé zmeny v jeho genetickej informácií. Stará sa o snahu priniesť do populácie rozmanitosť. Mohli sme si všimnúť, že operátor kríženia nedokáže vytvoriť jedinca s alelou génu, ktorá sa v populácií nenachádzala, pretože pracuje len s prehadzovaním genetického materiálu danej populácie. A ak sa v celej populácií nenachádza alela istého génu, ktorá by mohla zaistiť následné celkové zlepšenie zdatnosti populácie, tu prichádza operátor mutácie k veľkému úžitku.

  Je nutné ale dbať na okolnosť, že tento operátor dokáže priniesť do populácie i veľké škody zmenením variant génov na horšie. Preto je užitočné používať mutáciu sporadickejšie, teda nastaviť pravdepodobnosť jej používania na malé hodnoty.

- **Reprodukcia** — Operátor reprodukcie je najpriamočiarejší. Ako vstup vezme jedinca a bez zmeny ho prenesie do ďalšej generácie. Tento operátor do istej miery chráni evolučný algoritmus pred znehodnotením populácie zvyšnými genetickými operátormi.

  Reprodukcii je blízka technika zvaná **elitizmus**. Hoci reprodukcia dovolí selekciou vybranému jedincovi prejsť do ďalšej generácie bez zmeny, *elitizmus* zaručí, že pred začiatkom vytvárania novej generácie sa prvých N(parameter elitizmu) najzdatnejších jedincov bez zmeny skopíruje a vloží do novej generácie. Zvyšok generácie sa vytvorí selekciou a genetickými operátormi spomenutými vyššie.

Majúc zhrnuté základné pojmy, ukážeme jednoduchý algoritmus evolučnej stratégie.

```
[0]  inicializuj populáciu a priraď zdatnosť
[1]  while not ukončovacia podmienka
[2]    selektuj rodičov na kríženie
[3]    vytvor potomkov krížením
[4]    mutuj potomkov
[5]    priraď potomkom zdatnosť
[6]    vytvor z potomkov novú generáciu
```

V riadku 0 vytvárame populáciu jedincov, spomenutú nultú generáciu, typicky generovanú náhodne. V riadkoch 2-6 prebieha iterovaný vznik generácií, ktorý sa v každom evolučnom algoritme môže nejakým prvkom líšiť. To, koľko generácií sa vytvorí je ohraničené tzv. *ukončovacou podmienkou*[4], ktorá je nastavená tak, že evolúciu ukončí po dosiahnutí vytýčeného cieľa. Týmto cieľom môže byť napríklad hodnota fitness funkcie, kde lepšia hodnota znamená nájdenie dostatočne zdatného jedinca.

3.1.1 Modely evolučných algoritmov

Je mnoho spôsobov, ako zakódovať jedinca, či vybrať vhodné varianty selekcie a genetických operátorov. Preto je prirodzené, že existujú rôzne modely evolučných algoritmov líšiace sa práve v týchto vlastnostiach. Základné charakteristiky niektorých modelov si teraz zhrnieme.

Genetický algoritmus [12] sa najbližšie približuje pravej biologickej interpretácií evolúcie, pretože kóduje vlastnosti jedinca do množiny génov. Najbežnejšie kódovanie je pomocou bitového reťazca, reprezentujúceho spomínané gény.

Evolučná stratégia [13], na rozdiel od *Genetického algoritmu*, vlastnosti jedinca nekóduje, používa skôr ich pravú reprezentáciu. Typicky sa pracuje s vektorom čísel s pohyblivou desatinnou čiarkou. Týmto spôsobom je teda schopná efektívne prehľadávať cieľový priestor, používajúc najmä operátor mutácie.

Genetické programovanie [14] má podobnú schému evolúcie ako *Genetický algoritmus*. Pracuje ale so zložitejšou štruktúrou jedincov, reprezentuje ich ako stromy. Je teda schopným kandidátom k evolúcií programov.

Evolučné programovanie [15] je podobné *Evolučnej stratégií*, no nemá žiadne obmedzenie na dátový typ pri parametrov jedinca. V evolučnom programovaní je proces evolúcie zameraný na celé druhy jedincov, nie na jedince samotné.

3.1.2 Evolučné stratégie

Tu si detailnejšie popíšeme model evolučných algoritmov zvaný *Evolučná stratégia*, ukážeme si základné charakteristiky a zásadné rozdiely oproti ostatným evolučným algoritmom. Text o tomto modely je čerpaný z [16].

Ako bolo už naznačené, evolučné stratégie nepracujú s kódovanou reprezentáciou atribútov jedincov v podobe génov, ale s ich priamou hodnotou. Najmä pri atribútoch o reálnych hodnotách by bolo vhodné, keby operátor mutácie adaptívne menil veľkosť a smer mutácie v závislosti od topológie prehľadávaného priestoru. Parametre špecifikujúce vlastnosti mutácie sa nazývajú *strategické parametre* a atribúty riešenia (jedinca) voláme *genetické parametre*. Evolučná stratégia sa snaží optimalizovať oboje tieto parametre[5].

Pokračujúc značením, populáciu jedincov označme $\mathcal{P}$. Každý jedinec sa skladá z vektoru genetických parametrov $\boldsymbol{x} \in \mathbb{R}^n$ a strategických parametrov s a zdatnosti $f(\boldsymbol{x})$, teda ho označíme trojicou $(\boldsymbol{x}, s, f(\boldsymbol{x}))$.

Vystihnúc jeden z rozdielov od ostatných metód evolučných algoritmov, selekcia v evolučnej stratégií býva deterministická a rozlišujeme jej dva druhy:

[4] Z anglického *terminal condition*.

[5] Preto sa tomuto modelu evolučných algoritmov hovorieva i Evolúcia evolúcie

- **Environmentálna selekcia** slúži na zúženie veľkosti populácie na požadovanú veľkosť. V závislosti od zdatnosti jedinca $f(\boldsymbol{x})$ sa vyberie μ najlepších. Hneď si môžeme všimnúť deterministický prvok v tejto selekcií.

- **Selekcia párenia** vyberá jedincov z populácie za účelom nájdenia rodičov. Táto selekcia môže byť závislá od zdatnosti jedincov (výber je ale stále deterministickou cestou), alebo takto závislá byť vôbec nemusí (môže byť ako deterministická, tak i mať stochastický charakter). V prvom prípade smie byť environmentálna selekcia vynechaná, v druhom je ale nevyhnutná pre usmerňovanie evolúcie k lepším výsledkom.

Ďalší fakt k povšimnutiu je, že **Rekombinácia** (*kríženie*, ako poznáme z evolučného algoritmu) v evolučných stratégiách kríži niekoľko rodičov na vygenerovanie *jediného* potomka. Zvyčajne sú použití viac ako dvaja jedinci, čo sa nazýva *multi-rekombinácia*. Najdôležitejšie typy operátorov rekombinácie v evolučných stratégiách sú:

- **Diskrétna** — Pre každú premennú (parameter) z $\boldsymbol{x}$ je uniformne náhodne vybraný jeden jedinec z množiny rodičov (vytvorenej za pomoci selekcie párenia) aby táto premenná zdedila hodnotu daného jedinca.

- **Priemerová** — Vezme priemernú hodnotu zo všetkých rodičov, tým vytvorí nového potomka.

- **Vážená** – Je to zovšeobecnenie priemerovej rekombinácie. Namiesto priemernej hodnoty pracuje s váženým priemerom, kde váha zodpovedá zdatnosti rodiča.

Notácia Evolučných stratégií

Vieme, že evolučné algoritmy pracujú na iteratívnej báze. Preto nie je prekvapivé, že evolučné stratégie budú túto cestu dodržovať. Naviac, boli zavedené mnemotechnické notácie pre opis tejto iterácie. Notácia sa píše $(\mu/\rho^{+}_{,}\lambda)$-ES[6] a predstavuje tieto charakteristiky:

- μ označuje počet jedincov v populácií.

- ρ je význačné pri rekombinácií a hovorí, koľko jedincov má byť vybratých *selekciou párenia*. To znamená, že musí platiť $\rho < \mu$.

- λ hovorí, koľko potomkov sa má vygenerovať v jednej iterácií.

- $^{+}_{,}$ označuje, s akou množinou jedincov má environmentálna selekcia pracovať. V „plus“-selekcií sa vyberá μ najlepších z $\mu + \lambda$ jedincov, teda sa nevyberá len z vygenerovaných potomkov, ale aj z populácie, ktorá ich vytvorila. V „comma“-selekcii sa jedinci z aktuálnej generácie nedožijú selekcie a vyberá sa len z novovytvorených potomkov. Prirodzene tu teda musí platiť $\lambda \geq \mu$.

[6] ES značí Evolučná Stratégia.

Šablóny Evolučných stratégií

$(\mu/\rho\dagger\lambda)$-ES šablóna

```
[0]  initialize P
[1]  while not terminal_condition
[2]    for k = 1 to λ
[3]      (x_k, s_k) = recombine(select_mates(ρ, P))
[4]      s_k = mutate_s(s_k)
[5]      x_k = mutate_x(x_k, s_k)
[6]    P <- P ∪ {(x_k, s_k, f(x_k))|0 < k ≤ λ}
[7]    P <- select_by_age(P)
[8]    P <- select_μ_best(μ, P)
```

Táto šablóna zobrazuje algoritmus opisujúci základné chovanie evolučných stratégií, ktorou sa veľké množstvo ich variant riadi. Vysvetlíme si teda na nej ich prácu.

$\mathcal{P}$ je populácia jedincov veľkosti μ. $\boldsymbol{x}_k$ je vektor genetických parametrov a s_k obsahuje strategické parametre k-teho jedinca. V každej generácií sa vytvorí λ jedincov, každý z nich rekombináciou ρ vybraných rodičov (riadok 3). Nasleduje mutácia potomka začínajúc so s (riadok 4), ďalej $\boldsymbol{x}$ (riadok 5). Po mutácií sú potomkovia pridelení do $\mathcal{P}$ (riadok 6). Nasleduje selekcia populácie podľa veku jedincov (riadok 7), v „plus“-selekcií sa neučinia žiadne zmeny. Nakoniec deterministicky vyberieme z populácie μ najlepších (riadok 8).

Je dôležité spomenúť jednu konkrétnu variáciu tohto algoritmu, a síce keď $\rho = \mu$. Pretože za podmienky, že rekombinácia pracuje deterministicky, čo je častým zvykom, výsledok rekombinácie je v jednej generácií stále rovnaký a nie je nutné ho počítať pri tvorení každého z λ potomkov. Tento tzv. *rodičovský centroid* môžeme tým pádom vypočítať v jednej iterácií len raz mimo `for` cyklu zjednodušujúc algoritmus.

$(\mu/\mu\dagger\lambda)$-ES šablóna

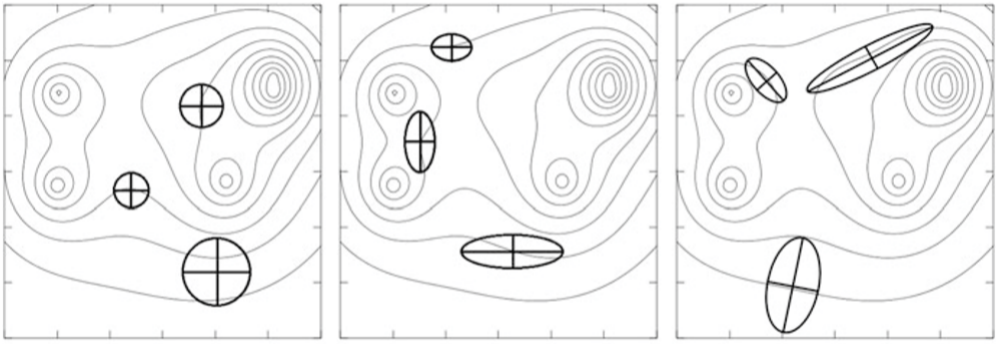
```
[0]  initialize P, x, s
[1]  while not terminal_condition
[2]    for k = 1 to λ
[3]      s_k = mutate_s(s)
[4]      x_k = mutate_x(x, s_k)
[5]      P <- P ∪ {(x_k, s_k, f(x_k))}
[6]    P <- select_by_age(P)
[7]    (x, s) = recombine(P)
```

V tejto šablóne iba *rodičovský centroid* $(\boldsymbol{x}, s)$ je vygenerovaný rekombináciou počas celej iterácie. Vytvorenie potomkov má teda na starosti len a práve mutácia (riadky 3-4). Prepočítanie rodičovského centroidu pre potrebu ďalšej generácie je presunuté za `for` cyklus, teda až po validnom vzniku súčasnej generácie (riadok 7).

Vďaka jej užitočným vlastnostiam ako jednoduchá analýza a formalizácia, či výkon v istých podmienkach sa stala varianta s jediným *rodičovským centroidom* široko populárna. Mnoho implementácií evolučnej stratégie pracuje práve na tomto princípe.

Mutácia a Evolúcia parametrov

Na **operátor mutácie** je v evolučných stratégiách braný nadmerný zreteľ, pretože, ako sme si mohli všimnúť v druhej šablóne, je vo veľkej miere zodpovedný za vytvorenie nových jedincov, tlačiac populáciu k lepším výsledkom. A ako už vieme, zavádza malú výchylku pridaním istej hodnoty k výsledku rekombinácie, teda vektoru $\boldsymbol{x}$. Táto hodnota je typicky vzatá z viacrozmerného normálneho rozdelenia $\boldsymbol{\mathcal{N}(0, C)}$ so strednou hodnotou rovnej nule a kovariančnou maticou $\boldsymbol{C} \in \mathbb{R}^{n \times n}$. V závislosti od viacrozmerného normálneho rozdelenia rozlišujeme tri operátory mutácie. **Sférická**, kde kovariančná matica je rovná matici identity. **Kolmá na osi**, kde je kovariančná matica diagonálna. **Všeobecná**, kde kovariančná matica je symetrická a pozitívne definitná (viď Obrázok 3.1).



Obr. 3.1: Tri 2-dimenzionálne viacrozmerné normálne rozdelenia používané pri mutácií (zľava) sférickej, kolmej na osi, všeobecnej. Tmavé elipsy ukazujú línie totožnej hustoty, svetlé vrstevnice naznačujú hodnoty fitness funkcie (zdroj [17]).

Evolúcia parametrov, alebo inak *ovládanie parametrov* mutácie je neodmysliteľná súčasť evolučnej stratégie. Ako príklad súc veľkosť kroku reprezentujúca faktor pridanej hodnoty k reálnemu vektoru $\boldsymbol{x}$. Očividne, táto veľkosť kroku do veľkej miery ovplyvňuje rýchlosť konvergencie evolučnej stratégie k očakávanému výsledku. Je teda len prirodzené, že existuje množstvo variácií evolučných stratégií zameriavajúcich sa práve na ovládanie parametrov. Jednoduché koncepty ovládania parametrov zachytáva tzv. **The 1/5 Success Rule** používajúci sférickú mutáciu, až najzložitejšia **Covariance Matrix Adaptation Evolution Strategy** pracujúca so všeobecnou mutáciou. Ich všeobecným cieľom je udržovať strategické parametre blízko k ich optimálnym hodnotám.

3.2 Neurónové siete

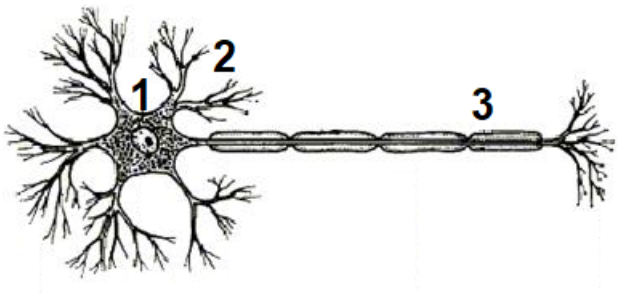
V tejto sekcii sa zameriame na jeden z dôležitých konceptov v oblasti umelej inteligencie a to na **neurónové siete**. Vysvetlíme si biologický pôvod tejto abstraktnej myšlienky, ďalej zadefinujeme pojem **formálny neurón**, toto všetko nakoniec vedúc k predstaveniu neurónových sietí. Ďalšie časti tejto sekcie sú čerpané z [18].

3.2.1 Biologická motivácia

Je účelné sa najprv zoznámiť so základnými poznatkami z neurofyziológie, ktoré nám umožnia pochopiť pôvodnú motiváciu umelých neurónových sietí. Preto si zopakujeme niektoré známe pojmy z biológie a vysvetlíme v danej oblasti používané termíny, ktoré nám pomôžu jednoducho vysvetliť ďalej spomenuté abstraktné pojmy.

Nervová sústava živých organizmov má na starosti sprostredkúvanie vonkajších vnemov, ich triedenie, uschovávanie a následné správne šírenie do výkonných častí organizmu. Toto zachytávanie vonkajších vnemov majú na starosti *receptory*, teda čidlá, ktoré dokážu vnímať jednotlivé podnety (napr. mechanické, tepelné, svetelné) a práve oni iniciujú šírenie vzruchu. Výkonné orgány, ku ktorým sa podnety šíria voláme *efektory*. Tieto vzruchy putujú po *projekčných dráhach* (už v ktorých prebieha predspracovanie informácie) až do najväčšieho centra nervovej sústavy, a síce *nervovej kôry*.

Neurón (viď Obrázok 3.2) je základný stavebný a funkčný prvok nervovej sústavy. Táto bunka je špecializovaná na získavanie, prenos, ukladanie a spracovanie informácií, ktoré prenáša vo forme elektrického signálu. Skladá sa z tela bunky, *neurocitu*, a z dvoch druhov výbežkov. Vstupných prenosových kanálov *dendritov* a jedného výstupného prenosového kanálu *axónu*. Axón je na svojom konci značne rozvetvený, spájajúci sa s dendritmi iných neurónov, čo zaručuje ich vzájomnú komunikáciu, prenos vzruchu.



Obr. 3.2: Obrázok neurónu (1 – neurocit, 2 – dendrity, 3 – axón).

Medzi dendritom a axónom susedných neorónov je takzvaná *synapsa* slúžiaca k preskoku informácie medzi neurónmi. Miera priepustnosti tejto synapsie určuje, či sa vzruch v nervovej sústave rozšíri, alebo utlmí.

Za určitých okolností dokáže neurón vyprodukovať elektrický impulz, ktorý reprezentuje prenášanú informáciu. Tento impulz je axómom prenesený do dendritov susedných neurónov cez ich synapsie, ktoré pomocou svojej priepustnosti rozhodnú o intenzite impulzu, a následnom podráždení neurónu. Ak je neurón dostatočne podráždený, prekonávajúci svoj *prah*, sám generuje elektrický impulz rozširujúc prenášanú informáciu ďalej po neurónovej sieti.

3.2.2 Formálny neurón

Formálny neurón preberá niektoré prvky z biologického neurónu, snažiac sa preformulovať jeho funkčnosť do matematickej podoby. Formálny neurón má n všeobecne reálnych vstupov $x_1, \ldots, x_n$, ktoré môžeme chápať ako dendrity. Ku vstupom sú zodpovedajúco pridelené *váhy* $w_1, \ldots, w_n$, teda reálne čísla, reprezentujúce ich priepustnosť. Možnosť zápornej váhy, prenesené do biologickej podoby, umožňuje existenciu synapsií tlmiacich prenášanú informáciu. Zvážená suma vstupných hodnôt neurónu predstavuje jeho *vnútorný potenciál* ξ (viď Výraz 3.1).

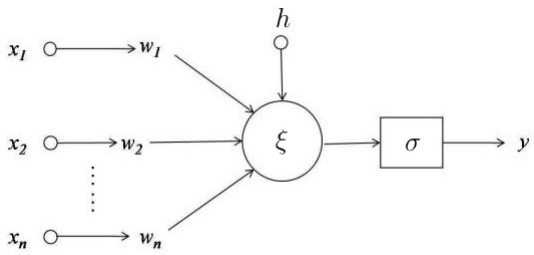
$$\xi = \sum_{i=1}^{n} w_i x_i \tag{3.1}$$

Výstup neurónu y je stanovený tým, či jeho vnútorný potenciál prekročí *prahovú hodnotu* h. Tento výstup nám určuje **aktivačná funkcia** modelujúca to, ako dostatočne je neurón podráždený. Najjednoduchšou aktivačnou funkciou je tzv. *ostrá nelinearita*, viď Výraz 3.2.

$$\sigma(\xi) = \begin{cases} 1, & \text{ak platí } \xi - h \geq 0 \\ 0, & \text{ak platí } \xi - h < 0 \end{cases} \tag{3.2}$$

Formálne sa typicky zavádza matematická formulácia, kde vnútorný potenciál neurónu ξ je vážená suma vstupov od ktorej sa naviac odčíta prah h, čo nám ďalej zjednoduší vyjadrovanie (viď Výraz 3.3).

$$\xi = \sum_{i=1}^{n} w_i x_i - h \tag{3.3}$$



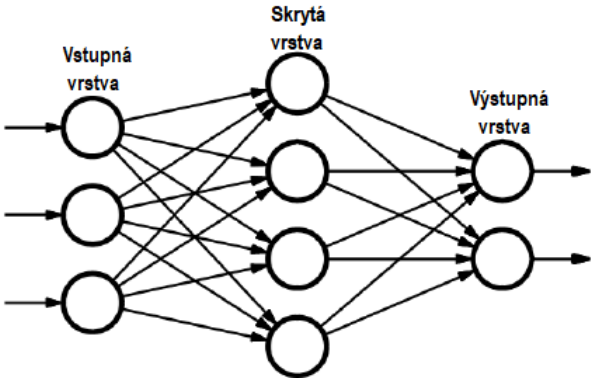
Obr. 3.3: Schematická štruktúra formálneho neurónu.

3.2.3 Neurónová sieť

Umelá neurónová sieť sa skladá z formálnych neurónov, kde výstup neurónu vedie do všeobecne niekoľkých vstupov ďalších neurónov. Toto symbolizuje biologické spojenie axónu cez synapsie s dendritmi iných neurónov, vytvárajúce sieť. To, ako sú neuróny medzi sebou prepojené určuje *topológiu* (*architektúru*) siete. V sieti rozlišujeme neuróny *vstupné*, *skryté* (alebo *pracovné*) a *výstupné*. Z biologickej analógie vstupné neuróny odpovedajú receptorom, výstupné efektorom

a skryté neuróny dráham medzi nimi, šíriace vzruchy. Hodnoty všetkých synaptických váh a prahov neurónov spoločne určujú *konfiguráciu* siete.

Rôznorodosť topológie neurónových sietí je typicky limitovaná len našou predstavivosťou. V zásade sa dá rozdeliť na dva typy, *cyklickú* a *acyklickú* topológiu. Cyklická topológia siete implikuje existenciu skupiny neurónov prepojených do kruhu. U acyklických sietí vieme neuróny rozdeliť do disjunktných *vrstiev* usporiadaných (napríklad vedľa seba) tak, že výstup neurónu vedie len z ľavých vrstiev do pravých, výstupy všeobecne prekračujúc i viac ako jednu vrstvu. Najčastejšia neurónová sieť vzhľadom na acyklickú topológiu je tzv. **Viacvrstvová neurónová sieť**. Pri nej platí, že nultá (ľavá) vrstva sa nazýva *vstupná*, tvorená zo vstupných neurónov a posledná *výstupná* vrstva tvorená výstupnými neurónmi. Ostatné, *skryté* vrstvy sú tvorené neurónmi skrytými. Dôležitým kritériom viacvrstvovej siete je fakt, že vstupy neurónov každej vrstvy (až na vstupnú) sú spojené s výstupmi vrstvy práve predchádzajúcej. Tiež platí, že medzi dvoma susednými vrstvami je každá dvojica neurónov v opisovanom spojení. Vďaka tomuto môžeme takúto sieť definovať len postupnosťou kladných prirodzených čísel (oddelených pomlčkou), každé číslo ukazujúce počet neurónov vo vrstve (napríklad 3-4-2, viď Obrázok 3.4).



Obr. 3.4: Príklad topológie viacvrstvovej neurónovej siete.

Práca, alebo aktivita, neurónovej siete spočíva v tom, že sa vstupným neurónom nastaví ich stav (výstup). Po tejto inicializácií nastáva výpočet siete, kde jeho výsledkom je stav neurónov vo výstupnej vrstve. Aplikujúc toto pravidlo, uvážme konkrétnu topológiu viacvrstvovej neurónovej siete s danou konfiguráciou. Na začiatku sú nasýtené neuróny vstupnej vrstvy stavmi v podobe všeobecne reálnych čísel. Výpočet siete prebieha v diskrétnom čase. V prvom časovom kroku sú aktualizované stavy prvej skrytej vrstvy, teda každý jej neurón vypočíta svoj vnútorný potenciál pomocou váženej sumy stavov zo vstupnej vrstvy, vracajúc svoj stav pomocou aktivačnej funkcie. V kroku druhom sú obdobne aktualizované stavy neurónov druhej skrytej vrstvy pomocou stavov neurónov z predchádzajúcej vrstvy. Tento postup prebieha, pokým nie sú aktualizované stavy neurónov z výstupnej vrstvy. Tu výpočet končí vracajúc výstup neurónovej siete.

Hlavné využitie neurónových sietí je aproximácia funkcií. Narozdiel ale od tradičných aproximačných metód pomocou Taylorovej, či Fourierovej rady neurónovej sieti nie je funkcia F na aproximovanie poskytnutá explicitne, ale implicitne vo forme vstupov a výstupov. Typicky ani nemáme k dispozícií celý obor hodnôt F, cielime ale k čo najlepšiemu priblíženiu sa správaniu funkcie F. To znamená,

že sieť dostávajúca vstup F sa snaží mať svoj výstup totožný s výstupom F, dosahujúc to škálovaním konfigurácie siete. Toto majú na starosti *learningové algoritmy*, najznámejší z nich zvaný *backpropagation* [19].

Mimo konfigurácie siete ovplyvňuje jej výstup i aktivačná funkcia použitá na výpočet stavu neurónov (typicky každý neurón v jednej sieti používa rovnakú aktivačnú funkciu). Okrem ukázanej, biologicky inšpirovanej aktivačnej funkcie (viď Výraz 3.2) sa obvykle stretávame so spojitými funkciami, napr. *sigmoida* (3.4), či *ReLU funkcia* (3.5). Každá aktivačná funkcia má iné vlastnosti a charakteristiky, rôzne pomáhajúc učeniu neurónových sietí.

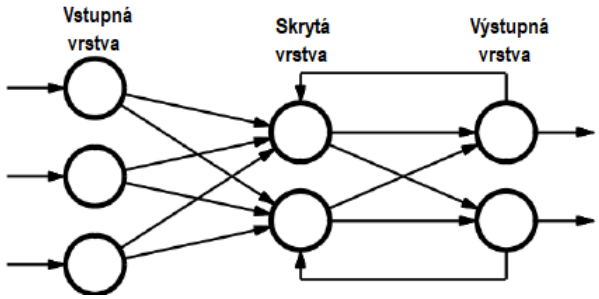
$$\sigma(\xi) = \frac{1}{1 + e^{-\xi}} \qquad \text{Sigmoida} \tag{3.4}$$

$$\sigma(\xi) = max(0, \xi) \qquad \text{ReLU funkcia} \tag{3.5}$$

Rekurentná neurónová sieť

Jedna z nevýhod *viacvrstvovej neurónovej siete* je, že očakáva vzájomnú nezávislosť prichádzajúcich vstupov. Táto vlastnosť nie je vždy žiadaná, napríklad ak sa snažíme pomocou siete predpovedať ďalšie slovo vo vete, alebo snažiť sa rozhodnúť aký nasledujúci krok má postavička v hre vykonať.

Na toto nám pomáhajú tzv. **Rekurentné neurónové siete** (viď Obrázok 3.5), ktoré hoci vypočítavajú každý vstup rovnako, no výstup závisí na predchádzajúcom výpočte. Inak povedané, tieto siete obsahujú „pamäť“ uchovávajúcu si predchádzajúce výpočty. Sú to cykly, čo v topológii rekurentných neurónových sietí umožňujú pamätať si a znovu použiť vypočítané signály na určitú dobu času.



Obr. 3.5: Príklad rekurentnej neurónovej siete, kde je skrytej vrstve sprostredkúvaná informácia posledného výpočtu výstupnej vrstvy.

4. Analýza problému učenia v hre PuppetWars

Táto kapitola je venovaná problémom a úskaliam spojenými s učením neurónových sietí reprezentujúcich stratégie hry PuppetWars, nad ktorými sa oplatí pred začatím predvádzania experimentov zamyslieť a popísať si ich. Na začiatku si zhrnieme, aké druhy učenia umelej inteligencie použijeme a ďalej sa budeme vo veľkej miere venovať parametrom, ktoré budú toto učenie ovplyvňovať.

4.1 Učenie pomocou neuroevolúcie

Stratégie herných postavičiek budeme vytvárať pomocou evolučného algoritmu. Ten bude učiť dva rôzne druhy neurónových sietí a to *viacvrstvové* a *rekurentné*.

Pri učenú sietí bude ich topológia statická, nemeniaca sa počas evolúcie. Cielime teda po najideálnejšej konfigurácií siete využívajúc schopnosti evolučných algoritmov.

Výber pevnej topológie neurónových sietí bol motivovaný snahou implementovať čo najvýkonnejšiu reprezentáciu sietí použitím vlastnosti jazyka *C++* zvanej *Variadic Template*. Pevná topológia nám taktiež umožňuje vyťažiť z konkrétnej instancie architektúry čo najviac snažiac sa dosiahnuť až jej limity, ukazujúc vrchol schopností istých architektúr.

Pre snahu zjednodušiť implementáciu reprezentujeme sieť iba ako súbor synaptických váh, nezahrňujúc prahy neurónov. Rovnako však predpokladáme, že siete s absenciou prahov neurónov obstoja v experimentoch dostatočne dobre a prítomnosť prahov by bola len zbytočnou komplikáciou.

Siete budú používať *ReLU* [20] aktivačnú funkciu, ktorá sa v mnohých prípadoch osvedčila ako vhodná funkcia na trénovanie neurónových sietí a je výpočtovo nenáročná.

Na evolúciu použijeme evolučné stratégie, konkrétne *Covariance Matrix Adaptation Evolution Strategy* [21] (ďalej len CMA-ES). Evolučné stratégie sme vybrali ako hlavný používaný model pri učení neurónových sietí práve kvôli prítomnosti pevných topológií sietí. Vďaka tejto vlastnosti sa evolúcia môže sústrediť len na hľadanie najlepších synaptických váh siete, čo implikuje nadbytočnosť komplikovaného kódovania celej siete. Preto sme zvolili evolúciu pravých hodnôt atribútov (váh) jedinca (siete), k čomu sa nám evolučné stratégie hodia perfektne. Neurónovú sieť sme sa teda rozhodli reprezentovať reálnym vektorom naplneným jej synaptickými váhami.

CMA-ES sme vybrali pretože má z mnohých variácií evolučných stratégií najvýkonnejšiu evolúciu strategických parametrov, čo môže viesť ku rýchlemu smerovaniu k žiadaným zdatnostiam jedincov.

4.2 Hyperparametre

Naše experimenty obsahujú nemalé množstvo *hyperparametrov*, teda parametrov v experimentoch sa ďalej nemeniacich, definujúcich ich chovanie, smerovanie. Táto vlastnosť nie je práve žiadaná, pretože z nej vzchádza nutnosť ladiť hyperparametre ručne, typicky metódou pokus - omyl. Preto je užitočné si ich rozdeliť na jednoduchšie stráviteľné množiny so snahou mať experimenty čo najviac pod kontrolou a neodplašiť čitateľa zahltením prílišným množstvom informácií.

4.2.1 Hyperparametre neurónových sietí

Hyperparameter experimentov zahrňujúci neurónové siete je ich **topológia**. Hoci znie jednoducho, zahrňuje ale priveľké množstvo variácií, preto si ho rozdelíme na disjunktné časti, dosahujúc ľahšiu analýzu podproblémov:

1. **Počet neurónov vo výstupnej vrstve**
2. **Počet neurónov vo vstupnej vrstve**
3. **Topológia siete tvorená skrytými vrstvami**

1. Výstupná vrstva

Začnúc od najmenej komplikovaného, bod **1** vieme zanalyzovať jednoducho. Výstupná vrstva nám bude reprezentovať rozhodovanie, akú akciu má postavička našej hry vykonať. A keďže sme si v Kapitole 2 spomenuli, že v hre sa vyskytuje päť *Metód akcií* (viď 2.5.1), rozhodli sme sa mať vo výstupnej vrstve päť neurónov. Každému z nich bude prislúchať jedna akcia.

Tu sa môžeme rozhodnúť o **determinickosti** neurónovej siete, čo môžeme brať ako ďalší hyperparameter spojený s neurónovými sieťami. Akciu môžeme vybrať nasledujúcimi spôsobmi:

- *Deterministicky* — Vyberieme tú akciu, ktorá prislúchala najviac aktivovanému (podráždenému) neurónu vo výstupnej vrstve, teda neurónu s najväčšou hodnotou svojho vnútorného potenciálu.
- *Stochasticky* — Berúc ohľad na pomer každej z aktivácií výstupných neurónov ku ich sume, inšpirujeme sa *ruletovou selekciou* a obdobne vyberieme neurón, ktorého akcia sa má vykonať.

2. Vstupná vrstva

Pri bode **2** sa vieme odraziť od faktu, že vstupnú vrstvu siete použijeme tak, ako sa v biologickej motivácií problému neurónových sietí spomína. A síce to budú *receptory* hernej postavičky. Táto vrstva bude odrážať stav prostredia, v ktorom sa postavička momentálne nachádza. Teda ju budeme sýtiť vstupmi najmä pomocou *Metód pozorovania* (viď 2.5.1).

Tu sa môžeme rozhodnúť, akú kvalitnú informáciu o prostredí bude neurónová sieť prijímať, teda aký bude počet neurónov vo vstupnej vrstve a aké hodnoty im budeme priraďovať. Prirodzene, v prípade hry *PuppetWars* máme konečnú

informáciu o momentálnom stave prostredia, v ktorom sa postavička môže nachádzať (zahrňujúc napríklad zdravie postavičky, prítomnosť projektilu blížiaceho sa k nej...). To implikuje prítomnosť tak dostatočne kvalitnej informácie, že akákoľvek iná by bola jej podmnožinou. Toto by ale mohlo vyústovať v neprimerane veľký nárast vstupných neurónov, to najmä kvôli tomu, aká môže byť informácia mohutná.

V experimentoch si teda vyskúšame niekoľko formátov vstupnej vrstvy sietí. Otestujeme napríklad neurónové siete prijímajúce neúplnú, ale zato vhodne mohutnú informáciu vyúsťujúcu v malý počet neurónov vo vstupných vrstvách a potenciálne i vo vrstvách skrytých. Tiež skúsime kvalitnejšiu informáciu, ktorá môže priviesť do stratégie postavičky neočakávané rozhodovacie prvky.

3. Skryté vrstvy

Bod **3** je nuž jediný, v ktorom nemáme žiaden bod, či vlastnosť, pomocou ktorej by sme vedeli aspoň trochu špecializovať ideálnu topológiu.

Je ale kľúčové zvoliť takú veľkú skrytú časť siete, aby bola schopná dostatočne reagovať na stavy prichádzajúce zo vstupnej vrstvy, ale aj dosť malú na to, aby učenie siete nezaberalo priveľa času.

Topológia rekurentných neurónových sietí

Doteraz sme si rozprávali problémy, ktoré sa týkali neurónových sietí všeobecne. Tu sa nám nabáda otázka, aké neuróny pridať a aké spojiť, aby nám z viacvrstvovej neurónovej siete vznikla sieť rekurentná.

V práci sme cielili po jednoduchosti a snažili sa zmeniť obyčajnú *viacvrstvovú neurónovú sieť* tak, aby čo najviac vynikla vlastnosť „pamäte“, ktorou rekurentná neurónová sieť disponuje.

Definícia 2. *Majme viacvrstvovú neurónovú sieť $\mathcal{N}$ obsahujúcu skryté vrstvy $h_1, \ldots, h_n$ (predpokladá sa aspoň jedna skrytá vrstva).* Pamäťová vrstva P *v* $\mathcal{N}$ *je vrstva o ľubovoľnom počte neurónov, ktoré medzi sebou nie sú spojené. Výstup P slúži ako vstup vrstiev $h_1, \ldots, h_n$ a naviac výstup h_n je spojený so vstupom P.*

Nami používané rekurentné neurónové siete budú teda len viacvrstvové siete, ku ktorým bude pridaná *pamäťová vrstva* (viď Definícia 2). **Počet neurónov v pamäťovej vrstve** budeme brať ako hyperparameter rekurentnej siete.

Keďže sme si zadefinovali všeobecnú šablónu topológií rekurentných sietí, je správny čas zadefinovať si i značenie konkrétnej instancie takejto siete. Majúc teda napr. 3-5-4 viacvrstvovú sieť, rekurentnú sieť vzniknutú z nej pridaním pamäťovej vrstvy označíme 3-5-4.x sieť, kde x je veľkosť pamäťovej vrstvy.

4.2.2 Hyperparametre evolúcie

CMA-ES má prirodzene svoje hyperparametre zdieľané s evolučnými stratégiami, sú to tieto:

- počet jedincov v populácií μ
- počet generovaných potomkov λ

- typ rekombinácie

Ladenie týchto hyperparametrov[1] na ideálne hodnoty je jeden z typických príkladov metódy pokus - omyl. Zvyčajne sa zvolí základná, preddefinovaná hodnota, od ktorej sa snažíme odraziť hľadajúc lepšiu konfiguráciu hyperparametrov.

4.2.3 Hyperparametre hry

I samotná hra *PuppetWars* má vlastnosti, ktoré dokážu v markantnej miere ovplyvniť smerovanie učenia umelej inteligencie postavičiek. Teraz si predvedieme, ktoré to sú.

Skóre

Skóre je celé číslo, ktoré postavička počas hry nadobúda. Čím väčšie skóre dosiahne, tým si lepšie počínala v bojovom poli. Dôležitá vlastnosť je, že táto neoddeliteľná súčasť postavičky určuje jej *zdatnosť*, ktorá je hojne využívaná v použitom evolučnom algoritme[2].

Teda je len ľahko odvoditeľné, že nesprávne nastavené počítanie skóre postavičiek môže viesť k neblahým výsledkom. Pre evolučné algoritmy je zdatnosť jedinca jediná spätná väzba v tom, ako zvláda riešiť pridelený problém. Tým pádom, ak napríklad nastavíme, nech sa skóre postavičky zväčší každým ubudnutím jej života, evolučný algoritmus bude len preferovať postavičky s takýmito samovražednými sklonmi. Pre naše ciele sa takýmto výsledkom vyhneme.

Bojové pole

Bojové pole je jedno z najdôležitejších prvkov hry. Tiež je to miesto, na ktorom sa naše neurónovými sieťami ovládané postavičky učia byť stále zdatnejšie. Výber správneho poľa vie očividne do veľkej miery riadiť postup evolúcie.

Tento hyperparameter rozdelíme na menšie:

- **Nepriatelia**

  Je dôležité mať v bojovom poli nepriateľov. Oni sú práve to, čo tlačí postavičku k lepšej zdatnosti. Bez ich prítomnosti by v takejto instancii bojového poľa bola slabá rozmanitosť nadobúdania skóre a evolúcia by z nášho pohľadu stagnovala.

  Preto budú učené neurónové siete vystavené postavičkám s preddefinovaným chovaním (alebo tímom postavičiek) pre čo najväčšie stresovanie ich zdatnosti.

- **Štruktúra poľa**

  Štruktúra poľa, alebo konkrétne usporiadanie blokov v mriežke, definuje bojové pole. Tento hyperparameter nám prináša možnosť rôznych scenárov

[1] Čitateľ si mohol v zozname všimnúť absenciu ρ. To sa ako hyperparameter neuvádza, pretože CMA-ES je jedna z variácií evolučných stratégií, kde platí $\rho = \mu$ a vo veľkej miere sa používa už spomenutý *rodičovský centroid*.

[2] Práve kvôli faktu, že skóre určuje zdatnosť jedinca, sme skóre nespomenuli v druhej kapitole, ale zaradili sme ho radšej do tejto tematicky bližšej kapitoly.

na otestovanie žiadaných vlastností postavičky (môžeme napr. vytvoriť scenár, kde dokáže postavička prežiť len ak vezme zbraň z tam prítomného zdroja). Správna štruktúra poľa môže viesť k naučeniu špecializovaných vlastností postavičiek.

Dĺžka hrania

Pod *dĺžkou hrania* máme na mysli to, ako dlho necháme učené postavičky hrať hru na zistenie zdatnosti neurónovej siete ju ovládanej.

Je rozumné hranie ohraničiť počtom kôl, pretože v istých situáciách (najmä na začiatku evolúcie) nebudú postavičky schopné vyhrať — a teda i ukončiť — hru poskytnúc im hoci aj ľubovoľne veľa času.

Tréningová množina

Poslednou z vecí, ktorú je nutné spomenúť je *tréningová množina*, teda množina bojových polí, na ktorých evolúcia zisťuje zdatnosť jedincov.

Jednou z možností je, že tréningová množina obsahuje viac ako jedno bojové pole. Tu vytvárame robustné stratégie, ktoré ovládli umenie hry a dokážu vyhrávať ideálne na ľubovoľnom inom bojovom poli.

Ďalšou možnosťou je ale jednoprvková tréningová množina. Tu trénujeme jedincov len pre jedinú instanciu bojového poľa, čo môže mať za následok naučenia sa vzorov a využitia slabín daného poľa. Toto môže viesť k tomu, že v takomto bojovom poli dokážu byť lepšie ako robustné stratégie, hoci v ľubovoľne inom si typicky nevedú priveľmi dobre.

5. Experimenty

Táto kapitola sa zameriava na vykonané experimenty. Na začiatku si predstavíme tri stratégie, ktorých kvalitu budeme pomocou experimentov porovnávať. Ďalej prejdeme k popísaniu konfigurácií použitých pri pokusoch a ich samotnému špecifikovaniu. Na záver si ukážeme výsledky, ktoré si zhrnieme a porovnáme.

5.1 Porovnávané stratégie

Experimentami budeme skúmať, ako si vedú rôznymi metódami vytvorené stratégie postavičiek v hre *PuppetWars*. Budeme porovnávať tri stratégie:

- **Baseline**
- **Stratégia** vytvorená pomocou evolúcie **Viacvrstvovej neurónovej siete**
- **Stratégia** vytvorená pomocou evolúcie **Rekurentnej neurónovej siete**

Baseline

Stratégia *baseline* určuje minimálne požiadavky, ktoré od vytvorených stratégií očakávame, preto bude ovládať nepriateľov, na ktorých budú ďalšie stratégie učené. Je to stratégia napísaná ručne napodobňujúca chovanie tých umelých inteligencií v hrách, ktoré nejavia žiadne známky hlbšieho rozhodovania sa.

Jej chovanie vystihuje prívlastok „robotické“. Skladá sa z troch akcií, z ktorých si postavička v jednom kroku hry vždy jednu vyberie a vykoná v závislosti od splnenia podmienok akciám prideleným:

1. Ak vidíš nepriateľa, strieľaj.
2. Ak stojíš na zdroji, vezmi objekt.
3. Inak sa pohni.

Konkrétny kód vystihujúci túto postupnosť akcií bol k nahliadnutiu v Kapitole 2, **baseline** sa líši naviac tým, že nestrieľa na každú bábku, ktorú zbadá, ale rešpektuje svojich tímových spoluhráčov.

Stratégie vzniknuté neuroevolúciou

Oba modely siete, teda *viacvrstvovú* i *rekurentnú* neurónovú sieť budeme učiť pomocou CMA-ES. Tu teda prichádzajú na rad samotné experimenty, ktoré sa nám pokúsia priblížiť aké konfigurácie pokusov vzídu v najlepší úspech.

5.2 Pokusy

Je dôležité a tiež samozrejmosťou detailne špecifikovať a vysvetliť konfiguráciu a parametre pokusov. O to sa teraz pokúsime, začnúc definovaním parametrov spoločných pre všetky pokusy, pokračujúc popisom použitých variácií konkrétnych parametrov, všetko zakončiac preskúmaním výsledkov.

5.2.1 Spoločné parametre

Skóre

Pri každom pokuse budeme používať **skóre**. Skóre bábky je vypočítané sčítaním počtu *udalostí*, ktoré sú najprv ale vynásobené im prislúchajúcim *váham*. *Udalosti* a *váhy* sú parametre špecifické pre každé skóre. V pokusoch budeme používať skóre, ktoré je k nahliadnutiu v Tabuľke 5.1.

Udalosť	**Váha**
Pohyb	1
Nepriateľ vo výhľade	2
Poškodenie	50
Zranenie	-25
Lepšia zbraň	50
Horšia zbraň	-50
Vylepšenie	50

Tabuľka 5.1: Udalosti s prislúchajúcimi váhami, používané na výpočet skóre.

Pohyb značí ľubovoľné pohnutie sa bábky. Do skóre sme túto udalosť pridali motivujúc evolučnú stratégiu preferovať bábky, ktoré sa pohybujú a nestoja len na mieste. Hra tiež odmeňuje za to, že bábka má *nepriateľa vo výhľade* a že mu udelí *poškodenie* (hodnota ubudnutého zdravia nepriateľa), teda cielime po agresívnych stratégiách. Tiež si môžeme všimnúť, že váha *zranenia* (hodnoty ubudnutého zdravia bábky) je menšia, než váha *poškodenia*, čo bábke ideálne dovolí neváhať hrnúť sa do súbojov. Udalosti *lepšia/horšia zbraň*, *vylepšenie* zase motivujú bábky vziať objekty zo zdrojov, trestajúc ich ale za bezhlavé správanie v tomto smere.

Výstupná vrstva

Výstupná vrstva sietí použitých vo všetkých pokusoch je rovnaká. Skladá sa z piatich neurónov, každému prislúchajúca jedna z *Metód akcií* (viď 2.5.1), detailne popísaná v Tabuľke 5.2.

Neurón	**Akcia**
N0	`GoForward`
N1	`Shoot`
N2	`TurnLeft`
N3	`TurnRight`
N4	`PickItem`

Tabuľka 5.2: Konfigurácia výstupnej vrstvy testovaných neurónových sietí.

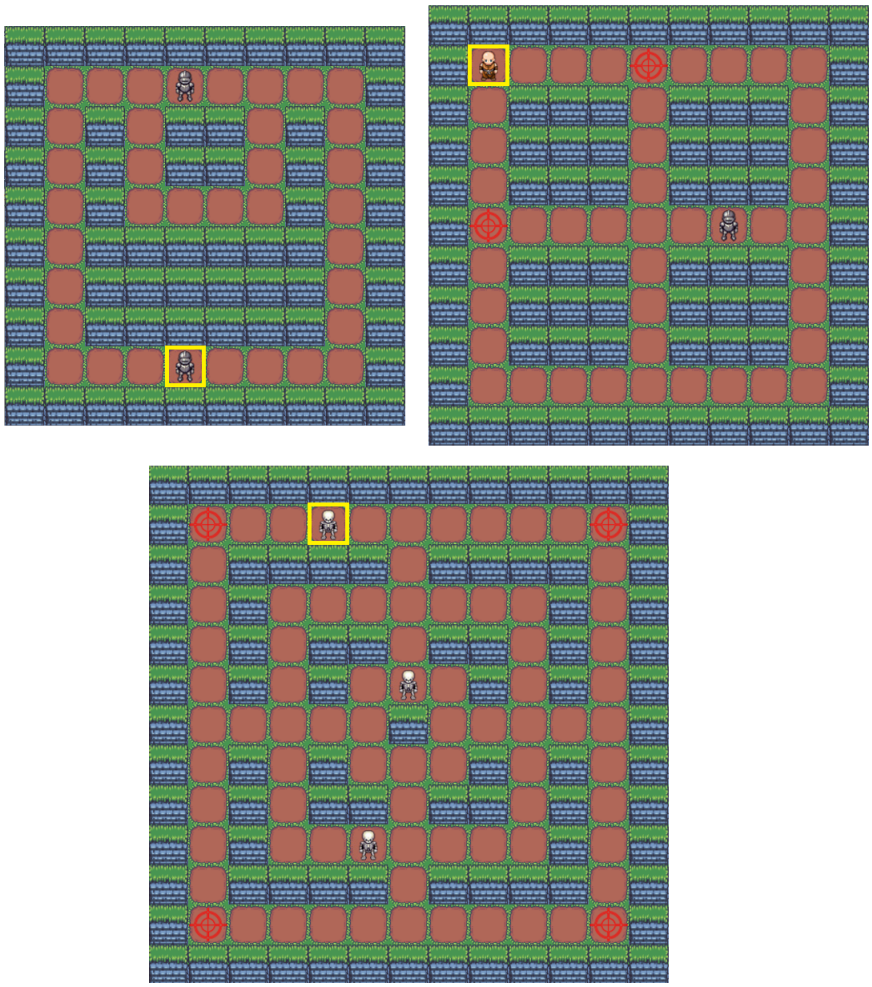
5.2.2 Použité tréningové množiny

V tejto sekcií si popíšeme dve tréningové množiny, na ktorých budeme učiť neurónové siete a vytvárať stratégie. Prvá tréningová množina pozostáva z troch

rôznych instancií bojových polí. Druhá je jednoprvková obsahujúca len jednu z troch ďalej špecifikovaných. Cielime tu k potvrdeniu našej teórie, že stratégie vzniknuté trénovaním na jednoprvkovej množine budú pre dané bojové pole vycvičenejšie, celkovo si ale v plnej tréningovej množine neobstojac dobre.

Množina troch

Táto tréningová množina sa skladá z troch instancií bojového poľa zobrazených na Obrázku 5.1.



Obr. 5.1: Snímky troch instancií bojového poľa. Žltým štvorčekom sú znázornené vyvinutou stratégiou ovládané postavičky.

V prvom bojovom poli (Obrázok 5.1 vľavo hore) je postavička nútená prejsť vzdialenosť aby narazila na nepriateľa a zneškodnila ho. Cielime teda po naučení postavičky zvedavosti, nie len nečinnému státiu na jednom mieste. Postavička i jej nepriateľ sú typu *knight* pre vyrovnanie síl.

Do druhého bojového pola (Obrázok 5.1 vpravo hore) je postavička typu *monk*, teda má oproti nepriateľovi zdravotnú nevýhodu. Je tu ale pridaný prvok, ktorý môže postavička využiť vo svoj prospech, a síce sú po bojisku rozmiestené zdroje. Vyvinutá stratégia, ktorá sa ich naučí používať získa veľkú výhodu odzrkadľujúcu sa na jej skóre.

V treťom poli (Obrázok 5.1 dole) je bábka postavená proti prevahe. Nemôže si teda dovoliť ísť bezhlavo do boja, musí si dávať pozor na svoje zdravie a podľa toho konať. Všetky bábku sú tu typu *skeleton*, využívajúc svoju rýchlosť vo väčšom bojovom poli.

Zdatnosť siete v populácií sa získa súčtom skóre dosiahnutých stratégiou siete vo všetkých hrách tréningovej množiny.

Jednoprvková

Jednoprvková množina bude obsahovať tretie, najkomplexnejšie bojové pole. Tu sa bude zdatnosť trénovanej siete rovnať skóre nahranom jej stratégiou.

5.2.3 Konfigurácie vstupnej vrstvy

Tu si ukážeme dva konkrétne počty vstupných neurónov a im priraďované vstupné hodnoty. Jedna z týchto konfigurácií bude jednoduchá dávajúc sieti malú, no dostatočnú informáciu o okolitom prostredí. Druhá bude zložitejšia podávajúca sieti niektoré komplexné vnemy z okolia. Náš predpoklad je, sýtiac sieť silnejšou informáciou o jej prostredí, bude si počínať lepšie ako taká s jednoduchou konfiguráciou.

Základná konfigurácia

Na začiatok použijeme jednoduchú vstupnú vrstvu popísanú v Tabuľke 5.3.

Neurón	**Vstup**
N0	Vpredu cesta
N1	Naľavo cesta
N2	Napravo cesta
N3	Nepriateľ vo výhľade
N4	Objekt vo výhľade
N5	Zdroj v blízkosti

Tabuľka 5.3: Základná konfigurácia vstupnej vrstvy neurónových sietí v pokusoch.

Neuróny budú podráždené (priradená im hodnota 1), ak *vstup* vyústi v pravdu, inak neurón podráždený nebude (priradí im hodnotu 0).

Prvých päť vstupov podáva sieti za seba hovoriace vstupy, teda to, čo bábka práve vidí. *Zdroj v blízkosti* znamená, že je bábka dostatočne blízko objektu, aby mohla úspešne zavolať `PickItem()` a vziať objekt.

Pokročilá konfigurácia

Ďalej v pokusoch použijeme i zložitejšiu vstupnú vrstvu (viď Tabuľka 5.4). Konfigurácia má spolu 13 vstupných neurónov poskytujúce okrem vstupov zo *základnej konfigurácie* aj niektoré pokročilejšie vstupy, ktoré si teraz vysvetlíme.

Vstup *nízke zdravie* podráždi neurón, ak zdravie postavičky klesne pod polovicu jej maximálneho zdravia. Neurón so vstupom *vo výhľade nepriateľa* dostane vstup 1, ak postavička vidí nepriateľa, ktorý je otočený oproti nej, teda i nepriateľ

Neurón	Vstup
N0	Nízke zdravie
N1	Vpredu cesta
N2	Naľavo cesta
N3	Napravo cesta
N4	Nepriateľ vo výhľade
N5	Vo výhľade nepriateľa
N6	Blížiaca sa strela
N7	Zbraň pripravená
N8	Pištoľ vo výhľade
N9	Puška vo výhľade
N10	Brokovnica vo výhľade
N11	Vylepšenie vo výhľade
N12	Zdroj v blízkosti

Tabuľka 5.4: Pokročilá konfigurácia vstupnej vrstvy neurónových sietí v pokusoch.

vidí ju. *Blížiaca sa strela* znamená prítomnosť vystreleného projektilu vo výhľade postavičky, ktorý smeruje k nej. Vstup *zbraň pripravená* podráždi neurón za podmienky, že zbraň je pripravená na streľbu.

5.2.4 Vykonávané pokusy

Majúc popísané a nazvané konfigurácie, ktoré budeme používať, predstavíme si samotné pokusy (viď Tabuľka 5.5).

Pokus	Tréningová množina	Vstupná vrstva	Typ siete	Topológia	Determ. siete
1.1.1.1a	Množina troch	Základná	Viacvrst.	6-10-5	Determ.
1.1.1.1b				6-10-5	Stoch.
1.1.1.2b				6-10-10-5	Stoch.
1.1.2.1a			Rekur.	6-10-5.5	Determ.
1.1.2.1b				6-10-5.5	Stoch.
1.2.1.1a		Pokročilá	Viacvrst.	13-10-5	Determ.
1.2.2.2b		Pokročilá	Rekur.	13-20-5.10	Stoch.
2.1.1.2b	Jednoprvková	Základná	Viacvrst.	6-10-10-5	Stoch.
2.2.2.2b		Pokročilá	Rekur.	13-20-5.10	Stoch.

Tabuľka 5.5: Rozpis špecifikovaných pokusov.

Pokusy **1.1.1.1a** a **1.1.1.1b** budeme porovnávať na zistenie, v akej miere ovplyvňuje *determinickosť* konfigurácie neurónovej siete jej stratégiu v hre.

K spomenutým dvom pokusom môžeme respektívne priradiť pokusy **1.1.2.1a** a **1.1.2.1b**, ktoré zahrnutým sieťam pridávajú *pamäťovú vrstvu* o veľkosti 5.

Pokus **1.1.1.2b** porovnáme s pokusom **1.1.1.1b** snažiac sa zistiť, ako si s rovnakým problémom poradí zložitejšia topológia siete.

V pokusoch **1.2.1.1a** a **1.2.2.2b** je neurónovým sieťam poskytnutá pokročilá konfigurácia vstupnej vrstvy. Výsledok prvého z práve spomenutých pokusov porovnáme s **1.1.1.1a** a zistíme, v akej miere si bude lepšie činiť zložitejšia neurónová sieť.

Pokusom **2.1.1.2b** a **2.2.2.2b** bude poskytnutá jednoprvková tréningová množina. Ich výsledky porovnáme s im prislúchajúcimi pokusmi **1.1.1.2b** a **1.2.2.2b** a presvedčíme sa, či je tvrdenie o lepšej pripravenosti v bojovom poli jednoprvkovej množiny pravdivé.

5.2.5 Výsledky

Teraz si predstavíme a zhrnieme výsledky uskutočnených pokusov. Každý pokus skončil, keď evolúcia dosiahla 1000-tu generáciu a každá hra na zistenie zdatnosti jedinca bola obmedzená na 1500 kôl. Týmto výberom sme cielili po kompromise medzi dostatočným množstvom času na evolúciu a nadbytočnou evolúciou ideálne zdatných jedincov.

Správu parametrov samotnej evolúcie sme prenechali na použitú knižnicu.

Pokusy sme zopakovali 4-krát so snahou viac sa priblížiť strednej hodnote výsledku evolúcií. Spriemerovaná zdatnosť najlepších jedincov poslednej generácie v evolúcii je zobrazená v Tabuľke 5.6. Všetky získané dáta sú tiež k nahliadnutiu v priloženom CD.

Pokus	**Zdatnosť**	**Smerod. odchýlka**
1.1.1.1a	8254	2483
1.1.1.1b	14883	3474
1.1.1.2b	18537	1648
1.1.2.1a	20319	7131
1.1.2.1b	21287	1731
1.2.1.1a	6369	4539
1.2.2.2b	3955	1476
2.1.1.2b	7625	1912
2.2.2.2b	3516	1944

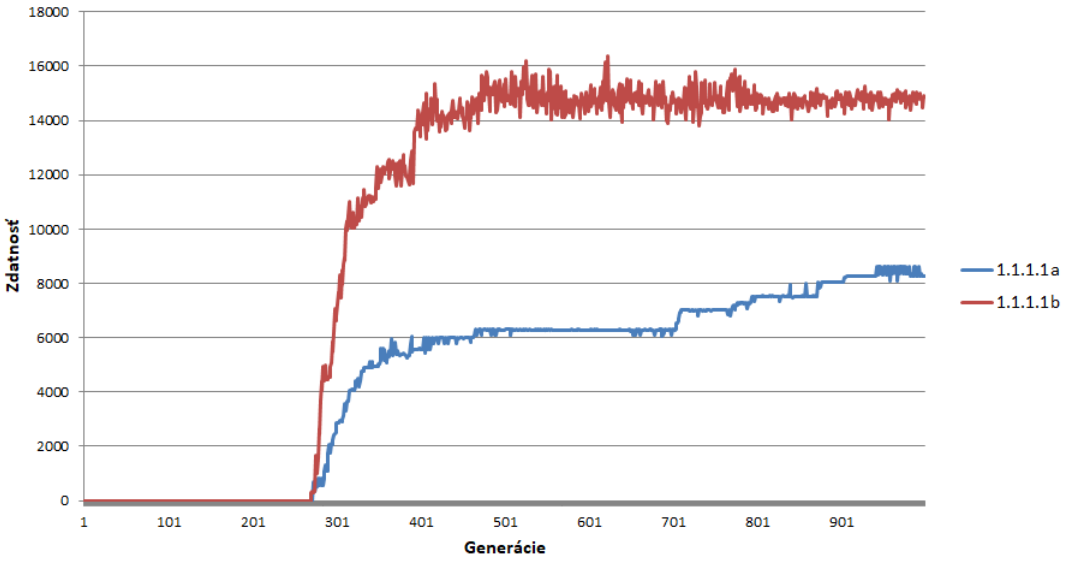
Tabuľka 5.6: Výsledky uskutočnených pokusov.

Determinickosť

Prvá vec, ktorú môžeme pozorovať je, že pri variantách pokusov s rovnakými parametrami až na *determinickosť siete* vedeli dosiahnuť lepšiu zdatnosť siete so stochastickou zložkou. Tiež si môžeme všimnúť väčšiu stabilitu získaných výsledkov, pozorujúc všeobecne nižšiu smerodajnú odchýlku.

Toto si vysvetľujeme faktom, že stochastická sieť umožňuje uvoľnenejší prechod medzi rôznymi akciami pri evolúcií siete, pričom sieť deterministická, vzhľadom na to, ako pracuje, má v tomto obore skôr charakter skokový. Prvok pravdepodobnosti má teda väčšiu šancu dovoliť jedincom vykonať i menej pravdepodobné akcie a teda aj širšie preskúmať prehľadávaný priestor pod záujmom získania lepšej zdatnosti.

Siete so stochastickým parametrom môžu mať ale problém s možnosťou chovať sa na pohľad až priveľmi náhodne. Toto chovanie pozorujeme najmä keď je sieť v mladej generácií evolúcie a jej váhy ešte nie sú dostatočne správne nastavené (viď Obrázok 5.2).



Obr. 5.2: Porovnanie pokusov líšiacich sa v determinickosti siete.

Model siete

Ďalšia vlastnosť, ktorá sa dá pozorovať je veľká obstojnosť pokusov pracujúcich s rekurentnými neurónovými sieťami oproti pokusom s rovnakými parametrami, no používajúc viacvrstvovú neurónovú sieť (viď Obrázok 5.3).

Vieme si to obhájiť myšlienkou, že v type hry, na ktorej boli siete trénované, je veľmi užitočné, keď postavička má k dispozícií a smie používať *pamäť*. Teda možnosť rozhodovať len podľa toho, čo postavička vidí a bez ďalšej informácie o predchádzajúcich udalostiach sa ukazuje ako veľká nevýhoda pri testovaných stratégiách.

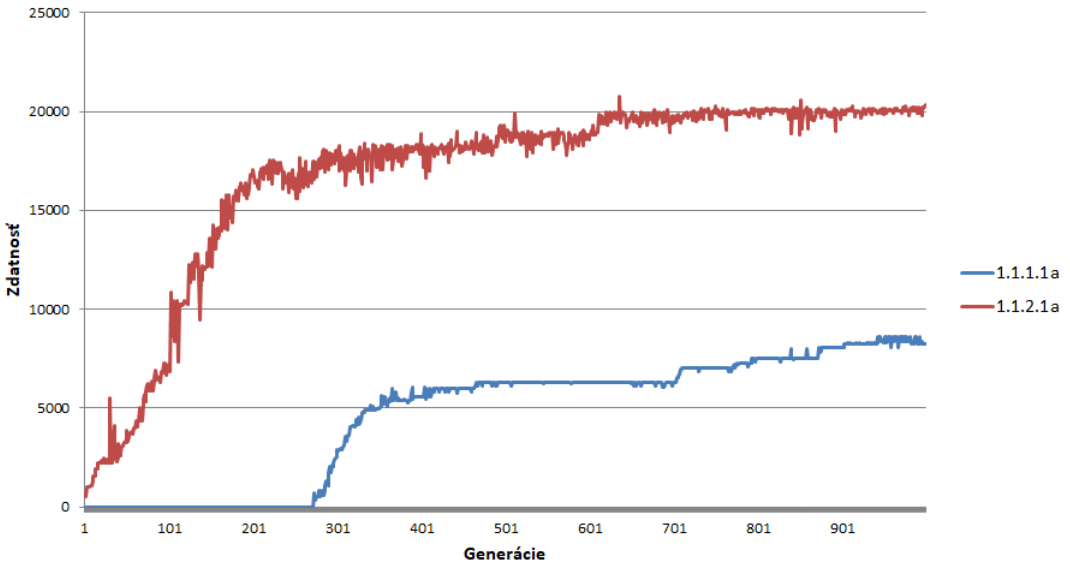
Topológia

Výsledky tiež podporujú našu teóriu, kde predpokladáme lepšiu zdatnosť siete rozdielnu od menej zdatnej iba v topológii skrytých vrstiev siete (viď Obrázok 5.4).

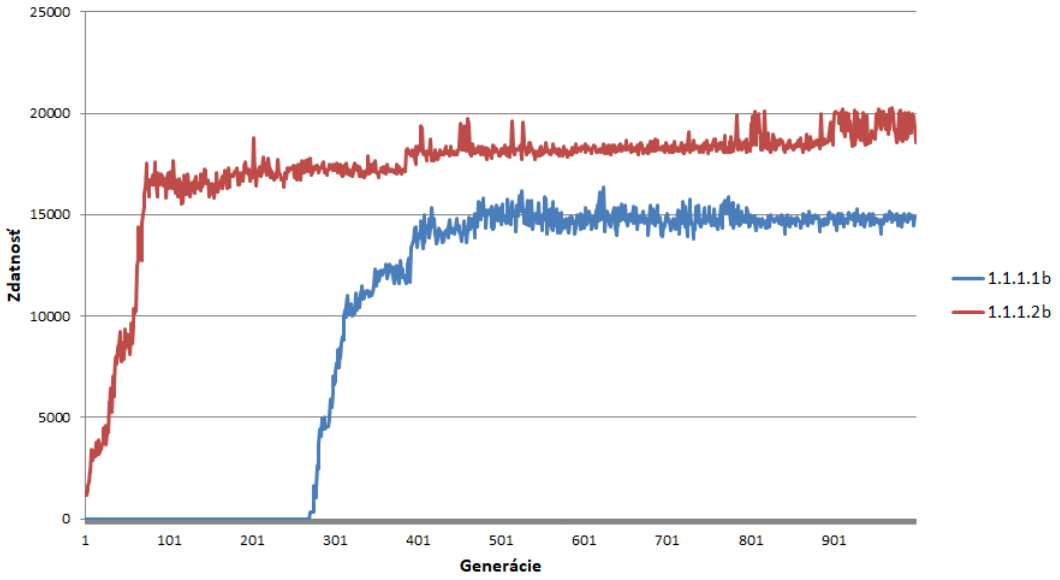
Vysvetľujeme si to tým, že siete so zložitejšou topológiou majú lepšiu schopnosť spracovať informáciu získanú na vstupe a teda i lepšiu schopnosť vrátiť ideálny výstup pre danú situáciu.

Vstupná vrstva

Pokusy s pokročilou konfiguráciou vstupnej vrstvy nedosiahli očakávané výsledky. Pokus **1.2.1.1a** nepredčil jemu ekvivalentný **1.1.1.1a**. Predpokladáme, že pôvodom problému je fakt, že konfigurácia siete bola priveľmi komplikovaná



Obr. 5.3: Porovnanie pokusov líšiacich sa v modely siete.



Obr. 5.4: Porovnanie pokusov líšiacich sa v topológii skrytých vrstiev siete.

na to, aby sa za poskytnutý počet generácií dokázal vytvoriť dostatočne zdatný jedinec.

Pri pokusoch **2.2.2.2b** a **1.2.2.2b** sa tento trend dá pozorovať ešte väčšmi usudzujúc, že evolúcia sietí s pokročilou konfiguráciou vstupnej vrstvy spojenou so zložitou topológiou nedokáže vyvynúť zdatného potomka tak rýchlo ako je to pri jednoduchších sieťach.

Tréningová množina

Ukázala sa pravdivosť nášho tvrdenia, kde sme naznačovali možnosť siete intenzívne učenej na jednej instancii bojového poľa prekonať v danom bojovom poli robustnejšie siete učené aj na ďalších instanciách poľa. Konkrétne, najzdatnejšia sieť z pokusu **2.1.1.2b** dokázala získať skóre 10902, pričom v pokuse **1.1.1.2b**

sa najzdatnejšej sieti podarilo v danom bojovom poli nahrať 6080. Usudzujeme teda, že sieť posilnila svoje vlastnosti užitočné pre dané bojové pole.

Idea, že vznikla stratégia úplne špecializovaná na jednu instanciu poľa je len podporená faktom, že, uskutočňujúc ďalšie pokusy, zdatnosť takejto stratégie na zvyšných instanciách poľa nepresiahla hodnotu 200.

Porovnanie s Baseline

Baseline v *množine troch* nahralo skóre 13921 a v *jednoprvkovej* zas 2792. Väčšina pokusov prekonala tieto hodnoty, dajúc pokusom s pokročilou konfiguráciou vstupnej siete viac času na evolúciu, pravdepodobne by *baseline* tiež predčili. V tomto ohľade teda môžeme zhrnúť výsledky experimentov za uspokojivé.

Záver

Pre potreby tejto práce bola implementovaná programovacia 2D strieľačka *PuppetWars*, ktorá dala vznik prostrediu na učenie umelej inteligencie. Toto prostredie sme využili a otestovali rôzne spôsoby učenia neurónových sietí pomocou evolučných algoritmov (najmä *Evolučnej stratégie*) na vytvorenie hernej stratégie, ktorá predčí jednoduché stratégie a vytvorí odolného protivníka.

Vytvorené stratégie mali ovládať postavičku v hre, ktorá mala správne reagovať na vnemy okolia, teda eliminovať nepriateľov a efektívne využívať objekty prostredia. Pomocou evolučnej stratégie boli skúmané vlastnosti rôzne konfigurovaných neurónových sietí. Použili sme modely sietí ako *Rekurentné neurónové siete* a *Viacvrstvové neurónové siete*.

Pokusmi sme si potvrdili teóriu, že evolúciou rekurentných sietí vytvoríme zdatnejšie stratégie, než to zvládnu obyčajné viacvrstvové siete, a to najmä kvôli prítomnosti pamäti. Ďalej sme pozorovali zaujímavý fakt, a síce že deterministické spracovanie výstupnej vrstvy vedie k spomaleniu evolúcie a vytvoreniu menej zdatných stratégií, na rozdiel od konfigurácií, kde sme do tohto procesu pridali stochastický prvok.

Posledné z pokusov sa zaoberali otázkou, či stratégie získané evolúciou sietí nad jedinou instanciou bojového poľa dokážu v danom poli predčiť robustné stratégie vytvorené neuroevolúciou nad väčšou tréningovou množinou, v ktorej je spomenutá instancia poľa obsiahnutá. Zistili sme, že evolučná stratégia dokáže vytvoriť špecializované stratégie, ktoré sú na jej trénovanom bojovom poli oveľa zdatnejšie ako robustné stratégie, využívajúc slabiny bojového poľa vo svoj prospech. Táto stratégia si ale v iných bojových poliach viedla prirodzene horšie.

Celkovo nám pokusy ukázali, že už len jednoduché konfigurácie neurónových sietí dokážu dosiahnuť očakávané výsledky, stávajúc sa slušnou konkurenciou reálnym hráčom. Zložitejšie konfigurácie by teda mohli vytvoriť stratégie rovnocenné ľudsky vytvoreným, nutne potrebujúc ale viac času na evolúciu.

Možné rozšírenia

Táto práca je otvorená rôznym pokračovaniam. Hra *PupperWars* bola programovaná so zreteľom na jednoduchú rozšíriteľnosť o nové herné prvky, ako zbrane, vylepšenia, či políčka. Toto všetko môže viesť k zložitejšiemu a zaujímavejšiemu prostrediu, v ktorom učiť umelú inteligenciu.

V pokračovaní práce by mohlo byť tiež zahrnuté vytváranie tímových stratégií, teda stratégií ovládajúcich celé tímy postavičiek, snažiacich sa spoločne kooperovať a zvíťaziť nad iným tímom.

Ďalšie z možných rozšírení by bolo uskutočňovanie pokusov s učením pomocou koevolície, teda by sa vyvíjali dve populácie, ktoré by medzi sebou na bojovom poli súperili. Takýto jav konkurencie by mohol dopomôcť evolúcií a viesť k nevšedne zdatným stratégiám.

Zoznam použitej literatúry

[1] D. G. Jones and A. K. Dewdney. Core war guidelines. *Scientific American*, 1984.

[2] *CoreWars*. URL: www.corewars.org, accesed May 4, 2018.

[3] Pamela McCorduck. *Machines Who Think: A Personal Inquiry into the History and Prospects of Artificial Intelligence*. A K Peters/CRC Press, 2004.

[4] Alex Lubberts and Risto Miikkulainen. Co-evolving a go-playing neural network. In *Proceedings of the GECCO-01 Workshop on Coevolution: Turning Adaptive Algorithms upon Themselves*, pages 14–19, 2001.

[5] David E Moriarty and Risto Miikkulainen. Discovering complex othello strategies through evolutionary neural networks. *Connection Science*, 7(3):195–210, 1995.

[6] Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[7] Luigi Cardamone, Daniele Loiacono, and Pier Luca Lanzi. Evolving competitive car controllers for racing games with neuroevolution. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1179–1186. ACM, 2009.

[8] A. Ghosh and S. Tsutsui. *Advances in Evolutionary Computing: Theory and Applications*. Springer, 2003.

[9] Charles Darwin. *On the Origin of Species by Means of Natural Selection, or the Presservation of Favoured Races in the Struggle for Life*. 1859.

[10] Thomas Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[11] Wolfgang Banzhaf, Peter Nordin, Robert E. Keller, and Frank D. Francone. *Genetic Programming: An Introduction*. Morgan Kaufmann Publishers, San Francisco, 1998.

[12] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence (Complex Adaptive Systems)*. A Bradford Book, 1992.

[13] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. PhD thesis, TU Berlin, 1971.

[14] J. R. Koza. *Genetic Programming – On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[15] L.J. Fogel, A.J. Owens, and M.J. Walsh. *Artificial intelligence through simulated evolution*. Wiley, Chichester, WS, UK, 1966.

[16] Nikolaus Hansen, Dirk V. Arnold, and Anne Auger. *Evolution Strategies*, pages 871–898. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[17] Thomas Bäck, Christophe Foussette, and Peter Krause. *Contemporary Evolution Strategies (Natural Computing Series)*. Springer, 2013.

[18] Jiří Šíma and Roman Neruda. *Teoretické otázky neuronových sítí*. Matfyzpress, Praha, 1. edition, 1996.

[19] Raul Rojas. *Neural Networks: A Systematic Introduction*. Springer, 1996.

[20] Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947–951, 2000.

[21] Nikolaus Hansen and Andreas Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 312–317. IEEE, 1996.

A. Obsah priloženého CD

adresár `data`

Obsahuje dáta vygenerované pri vykonávaní pokusov. Taktiež je v ňom súbor popisujúci štruktúru a obsah samotných dát.

adresár `doc`

Adresár obsahujúci dokumentácie k hre *PuppetWars* a aplikácií, ktorá vykonáva experimenty.

adresár `src`

Adresár zahrňuje zdrojové kódy k použitým programom práce.

adresár `bin`

Obsahuje spustiteľnú hru *PupperWars* ako aj preloženú aplikáciu vykonávajúcu pokusy.