SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD

Evolucijsko oblikovanje neuronskih mreža

Miroslav Crnić
Voditelj: *Domagoj Jakobović*

Zagreb, rujan, 2012.

Zahvaljujem mentoru prof. dr. sc. Domagoju Jakoboviću
na podršci i stručnom vodstvu pri izradi ovog rada.

Sadržaj

Popis tablica

Popis slika

1. Uvod

Umjetne neuronske mreže često su korištena metoda za rješavanje problema aproksimacije funkcija, klasifikacije, robotike ili obrade podataka. Radi njihove sposobnosti učenja i generalizacije na temelju primjera posebno su prikladne za probleme gdje količina ili kompleksnost podataka onemogućuje traženje rješenja postupcima koji daju egzaktan rezultat.

Ovisno o vrsti problema i načinu na koji se prezentiraju podaci koriste se razni algoritmi za učenje neuronskih mreža svaki sa svojim ograničenjima primjene i različitim razinama uspješnosti. Pokazano je da evolucijski algoritmi daju mnogo bolje rezultate od ostalih metoda pri učenju neuronskih mreža gdje postupak učenja pripada skupini algoritama za učenje s podrškom [6,7,8]. U okviru ovog rada bit će analizirana uspješnost NEAT metode [5] pri učenju neuronske mreže u odnosu na učenje neuronske mreže genetskim algoritmom. Obje metode pripadaju skupini evolucijskih algoritama.

Neuronske mreže će kroz postupak učenja pokušati aproksimirati funkciju kvalitete stanja na ploči za pojedinog igrača u igri dama.

2. Dama

Igra "dama" je jedna od najstarijih igara na ploči. Prvi tragovi igre potječu iz trećeg tisućljeća prije nove ere, a nađeni su na arheološkom nalazištu Ur u Iraku. Verzija igre nađena u Iraku koristila je malo drugačiju ploču, različit broj figura i nisu poznata točna pravila. Kasnija verzija igre potječe iz Egipta, oko 1400. godine prije nove ere, igrala se na ploči veličine 5x5 polja i bila je vrlo popularna. Prijelaz na šahovsku ploču dogodio se oko 1100. godine u Francuskoj čime je i broj figura jednog igrača porastao na dvanaest. Ubrzo nakon toga uslijedilo je i pravilo obaveznih skokova što je igru učinilo dinamičnijom. 1847. godine održano je prvo svjetsko prvenstvo u dami. Danas se dama u raznim varijantama igra širom svijeta.

Dama se igra na tamnim poljima šahovske ploče koja je u dami tradicionalno crveno-bijela ili zeleno-bijela. Igraju je dva igrača od kojih svaki na početku ima dvanaest identičnih figura (jedan igrač ima bijele, a drugi crveni ili crne) postavljenih na tamna polja u igraču tri najbliža reda. Prvi potez vuče bijeli, a dalje se igra naizmjenično. Postoji dvije vrste dozvoljenih poteza za figuru. Pomak je takav potez gdje se figura pomiče dijagonalno za jedno polje, ako je to polje slobodno. Skok je takav potez u kojem se figura pomiče za dva polja dijagonalno, ako je to polje slobodno i ako se na polju između početne i konačne pozicije figure nalazi protivnička figura. Protivnička figura se u tom trenutku uklanja sa ploče i iz igre. Ako je iz konačne pozicije skoka moguće napraviti novi skok, mora ga se napraviti i to je dio istog poteza. Isto tako, ako je moguće napraviti više poteza od kojih su neki skokovi nužno je odabrati potez koji je skok. U trenutku kada figura jednog igrača dođe u red najbliži protivničkom igraču, postaje kralj. Kralj se, za razliku od običnih figura koje se mogu micati i izvoditi skokove samo prema naprijed, može micati i izvoditi skokove prema naprijed i prema nazad. Igra završava u trenutku kada igrač koji je trenutno na potezu ne može povući potez. Igrač ne može povući potez ako nema više figura ili ako su figure blokirane to jest nemaju na raspolaganju niti jedan dozvoljen potez. U tom trenutku je taj igrač izgubio. Ako se tokom igre tri puta ponovi pozicija sa istim raspored figura na ploči i istim igračem na potezu, igra završava neriješeno.

Razvoj računalnih programa koji igraju damu počeo je sredinom prošlog stoljeća. Prvi program koji je igrao damu napisao je Christopher Strachey 1951. godine.

Vjerojatno najpoznatiji program za igranje dame, Chinook, napisan je 1989. godine te je danas najjači program za igranje dame. Prvu titulu svjetskog prvaka osvojio je 1994 godine protiv višegodišnjeg svjetskog prvaka Mariona Tinsleya. 1996 osvojio je prvenstvo Sjedinjenih Američkih država sa najvećom bodovnom razlikom ikad. Time su prestala natjecanja ljudskih igrača dame protiv programa. 2007 godine razvojni tim Chinooka objavio je da je pronađeno rješenje dame odnosno da je pronađena optimalna strategija pri kojoj, ako oba igrača igraju optimalno, igra završava neriješeno [1].

Procijenjeni broj dozvoljenih pozicija u dami je reda veličine 10^{20} dok je procijenjeni broj pozicija u igračem stablu reda veličine 10^{40} . Za ispitivanje svih mogućnosti odigravanja od početne pozicije bilo bi potrebno previše vremena.

Prilikom traženja rješenja dame korištena je drugačija metoda [1]. Prvo su generirane sve legalne pozicije u kojima igra završava pobjedom jednog od igrača ili neriješeno. Iz tih pozicija unatrag su generirana sva stanja koja su mogla voditi prema tim završnim pozicijama sve do pozicija na kojima se nalazilo deset figura na ploči. Kako bi dokaz bio potpun krenulo se iz početne pozicije te su se ispitivali nizovi poteza koji vode do dobivenih pozicija sa deset figura. Pri tome nisu ispitivane sve mogućnosti već je korišten program Chinook kako bi se odabrali nizovi koji završavaju u manjem broju koraka te su se odbacivali dijelovi stabla koji uz optimalnu igru jednog igrača brzo završavaju porazom. Na taj način ispitano je svega 10^{14} mogućih stanja i dokazano je da uz optimalnu igru oba igrača igra završava neriješeno.

3. Algoritmi za pretraživanje stabla igre

Stablo igre se definira kao usmjereni graf u kojem su čvorovi pozicije u igri, a veze između čvorova potezi. Ako kao korijen stabla uzmemo početnu poziciju igra te generiramo kompletan graf sa svim mogućim potezima i pozicijama dobivamo potpuno stablo igre.

Pri igranju neke igre odabiranje najboljeg poteza svodi se na pretraživanje stabla igre za potezom koji će nas uz optimalno igranje oba igrača što prije dovesti do pobjede ili što kasnije do poraza. Najpoznatiji i najjednostavniji algoritam za pretraživanje stanja igre je minimax algoritam.

3.1 Minimax algoritam

Minimax je algoritam iz teorije igara koji pretražuje stablo igre za najboljim mogućim potezom [2]. Svakom završnom stanju igre dodjeljuje se neka vrijednost. Pobjedi prvog igrača dodjeljuje se vrijednost beskonačno, pobjedi drugog igrača minus beskonačno, a neriješenom rezultatu, ako je moguć, dodjeljuje se neutralna vrijednost nula. U ovakvom slučaju prvi igrač pokušava završiti igru sa vrijednošću beskonačno, odnosno pokušava maksimizirati rezultat, dok ga drugi igrač nastoji minimizirati. Iz postupka traženja najboljeg poteza algoritam je dobio ime minimax.

Prilikom pretraživanja stabla svaki igrač pretpostavlja da će drugi igrač igrati optimalno, odnosno igrač se neće kretati prema pobjedi koja je uzrokovana pogreškom drugog igrača [2]. U slučaju kada je stablo maleno moguće je izvršiti iscrpnu pretragu cijelog stabla i stvarno izabrati optimalan potez, međutim u igrama sa velikim stablom iscrpna pretraga nije moguća te se određuje dubina do koje se stablo pretražuje. Budući da stanja na određenoj dubini nisu nužno sva završna nije moguće egzaktno odrediti vrijednost koju im treba dodijeliti. Kako bi odredili kvalitetu takvih nezavršnih stanja koristi se neka funkcija dobrote koja određenom stanju dodijeli vrijednost to veću što je stanje povoljnije za igrača koji pokušava maksimizirati odnosno to manju što je stanje povoljnije za igrača koji pokušava minimizirati. Prilikom potrage za idućim potezom minimax pretražuje sva stanja do određene dubine kako bi našao najbolji potez.

Postoje mnoge varijacije minimax algoritma koje uglavnom nastoje smanjiti broj

stanja koje je potrebno pregledati kako bi se odabrao najbolji potez. Na taj način skraćuje se vrijeme potrebno za odabir poteza odnosno moguće je povećati dubinu pretrage i pritom vratiti najbolji potez u razumnom vremenu.

3.2 Algoritam alfa-beta rezanja

Algoritam alfa-beta rezanja pokušava smanjiti broj stanja koje se pregledavaju u minimax algoritmu pri određivanju najboljeg poteza. Ako se prilikom pregledavanja određenog stanja ustanovi da uz optimalnu igru oba igrača možemo završiti u stanju sa manjom vrijednošću od do sada najboljeg stanja, prestaje se sa pretraživanjem tog dijela stabla. Budući da je već nađen bolji potez i znamo da u dijelu stabla koje se trenutno ispituje postoji gori ne zanima nas da li postoji neki još gori potez u tom dijelu stabla. Dolazi do rezanja dijela stabla te se smanjuje broj ispitanih stanja. U slučaju da ne dođe do rezanja, algoritam alfa-beta rezanja će ispitati jednak broj stanja kao i minimax algoritam.

Ako je prosječno grananje stabla b i dubina pretrage d te ne dođe do rezanja potrebno je ispitati $O(b^d)$ stanja. Pretpostavimo da su djeca svakog čvora slučajno raspoređena s obzirom na svoju vrijednost odnosno da je vjerojatnost s kojom dolazi do rezanja jednaka za svako dijete. U tom slučaju prosječan broj ispitanih stanja iznosi $O(b^{3d/4})$ [2].

Često se bolji rezultati postižu postupkom iterativnog produbljivanja pri čemu se prvo traži najbolji potez gledajući do dubine n pa zatim ako ima vremena do dubine $n+1$ dok ne ponestane vremena. Prilikom iterativnog produbljivanja dodatna optimizacija može se izvesti pamćenjem dobivenih vrijednosti pozicija kako bi se u idućem koraku prvo krenuli ispitivati dijelovi stabla koji vjerojatno nude bolje rezultate. Na taj način se vjerojatnost rezanja dodatno povećava.

4. Umjetne neuronske mreže

Umjetna neuronska mreža je matematički model bioloških neuronskih mreža. Sastoji se od više umjetnih neurona koji su međusobno povezani tako što su izlazi nekih neurona spojeni na ulaze drugih neurona ili na svoje ulaze. Neuroni na čije se ulaze postavljaju ulazni podaci nazivaju se izlazni neuroni, dok se neuroni čiji izlazi daju izlazne podatke mreže zovu izlazni neuroni. Neurone koji nisu niti ulazni niti izlazni nazivamo skriveni neuroni.

Prema strukturi neuronske mreže možemo podijeliti na neuronske mreže bez povratne veze i na neuronske mreže sa povratnom vezom. U neuronskim mrežama bez povratne veze signali putuju samo u jednom smjeru, od ulaza prema izlazu. Nigdje u strukturi nema ciklusa. U trenutku kada signali dođu na izlazne neurone stanje na njima je stabilno i više se ne mijenja dok se ne promjene ulazni podaci. Kod neuronskih mreža sa povratnom vezom mogući su ciklusi u strukturi neuronske mreže te povratak signala sa izlaznih neurona na ulazne ili na neki drugi dio neuronske mreže. Nakon promjene podataka na ulazu izlaz neuronske mreže sa povratnom vezom se konstantno mijenja dok mreža ne dođe u stanje ravnoteže.

Umjetni neuron se sastoji od "jezgre", nula ili više ulaza i jednog izlaza. Jezgra neurona je neka matematička funkcija koja preslikava ulaze na izlaz. Konstante u odabranoj funkciji su parametri neurona, a parametri svih neurona u mreži određuju ponašanje neuronske mreže to jest definiraju funkciju preslikavanja ulaza na izlaze neuronske mreže.

Sposobnost da nauče promatrajući dani skup ulaznih podataka, otpornost na pogreške i mogućnost paralelizacije izvođenja neke su prednosti koje čine neuronske mreže vrlo zastupljenim u rješavanje problema aproksimacije funkcija i raspoznavanje uzoraka te kompleksnih pravila u skupu podataka.

Jedna od mana neuronskih mreža, osobito u robotici, je činjenica da zahtijevaju vrlo dug i kompleksan proces učenja kako bi bile primjenjive u stvarnom svijetu. Druga mana se odnosi na problem tumačenja rezultata. Naučena neuronska mreža koja rješava određeni problem potencijalno sadrži toliku količinu parametara i dimenzija njezine ulazne funkcije onemogućava bilo kakvu analizu načina na koji mreža obrađuje ulazne rezultate.

Postupak određivanja parametara neuronske mreže, kako bi njezino ponašanje

rješavalo zadani problem, nazivamo učenje neuronske mreže. Prilikom učenja neuronske mreže nastojimo iz manjeg skupa reprezentativnih ulaznih podataka, određivanjem parametara, izvući pravila koja vrijede za cijeli skup ulaznih podataka, odnosno želimo da neuronska mreža nauči generalizirati.

U slučaju da je struktura neuronske mreže previše kompleksna za dani problem moguće je da će doći do prenaučenosti to jest da će neuronska mreža davati ispravne izlaze za sve ulazne podatke kojima je bila izložena tokom učenja, ali krive za ostale podatke iz skupa ulaznih podataka. U takvom slučaju mreža nije izvela nikakva generalna pravila nego je zapamtila ispravan izlaz za svaki ulaz kojem je bila izložena. Kada je struktura mreže nedovoljno kompleksna za neki problem postupak učenja neće uspjeti niti za skup ulaznih podataka prezentiranih tokom učenja.

Ovisno o vrsti podataka korištenih prilikom procesa učenja neuronske mreže, algoritmi za učenje neuronskih mreža mogu se podijeliti u tri kategorije:

- algoritmi za učenje pod nadzorom
- algoritmi za učenje bez nadzora
- algoritmi za učenje s podrškom

4.1 Učenje pod nadzorom

Kako bi se ostvarilo učenje neuronske mreže pod nadzorom potreban je dovoljno veliki skup ulaznih podataka sa poznatim željenim izlazima. Postupak učenja svodi se na podešavanja parametara neuronske mreže dok se za ulazne podatke ne dobiju željeni izlazni podaci. Nakon što je mreža naučila davati željene izlazne podatke za neki dovoljno veliki skup ulaznih podataka očekujemo da će davati dovoljno dobre rezultate za bilo koji element iz cijelog skupa mogućih ulaznih podataka. Očekujemo da je postupak učenja uspio iz skupa za učenje izvesti neka generalna pravila koja vrijede za sve moguće ulazne podatke. Kako bi se izbjegla prenaučenost skup ulaznih podataka sa poznatim izlazima se prije postupka učenja dijeli na dva skupa:

- skup za učenje
- skup za ispitivanje

Elementi skupa za učenje koriste se za podešavanje parametara neuronske mreže dok se skup za ispitivanje koristi za ispitivanje koliko dobro je neuronska mreža naučila.

4.2 Učenje bez nadzora

Učenje bez nadzora koristi se kad nije poznat željeni izlaz za skup ulaznih podataka. Nastoje se ustanoviti pravila u strukturi ulaznih podataka i za ulaze sa sličnom strukturom vratiti slične izlaze.

4.3 Učenje s podrškom

Kod učenja s podrškom nije moguće točno definirati koliko je izlaz neuronske mreže dobar odnosno loš za neki određeni ulaz. Definira se pojam kumulativne nagrade kao ukupna nagrada koja je dobivena određenim nizom izlaza neuronske mreže koji su dobiveni kao odgovori na ulaze koje postavlja okolina. Neuronsku mrežu možemo promatrati kao agenta u interakciji sa okolinom. Agent prima trenutno stanje okoline i može poduzeti neku akciju (izlaz neuronske mreže) koja okolinu postavlja u neko novo stanje. Svako novo stanje donosi neku trenutnu nagradu. U trenutku prekida interakcije kumulativna nagrada definira se kao suma trenutnih nagrada svih stanja u kojima se okolina nalazila. Cilj agenta je odabrati takve akcije da se maksimizira kumulativna nagrada.

Dobrota agentovih akcija poznata je tek nakon prekida interakcije i to ne za svaku akciju posebno već samo kao ukupna dobrota niza akcija.

Evolucijske metode u mnogim ispitivanjima pokazuju znatno bolje rezultate pri rješavanju problema učenja s podrškom u odnosu na druge metode [6,7,8].

5. Genetski algoritmi

Sredinom dvadesetog stoljeća javlja se ideja o korištenju Darwinovih načela evolucije pri rješavanju optimizacijskih problema. Kako je biološka evolucija uspješno stvorila živa bića koja svakog dana rješavaju mnoge kompleksne probleme vjeruje se da će simuliranje evolucije dovesti do pronalaska novih, efikasnijih rješenja optimizacijskih problema. Kako je kroz godine rasla procesna moć računala tako je rasla i popularnost i primjenjivost ovakvih optimizacijskih tehnika.

Šezdesetih godina dolazi do prve jasnije podjele na tri grane: evolucijsko programiranje, genetske algoritme i evolucijske strategije. Kasnije se

Genetski algoritam zajedno sa genetskim programiranjem i evolucijskim strategijama pripadaju skupini evolucijskih algoritama. Kasnije se pojavljuje još i genetsko programiranje te klasifikatorski sustavi. Sve navedene grane pripadaju zajedničkoj skupini algoritama sa nazivom evolucijski algoritmi.

Genetski algoritam najčešće se koristi za rješavanje kombinatoričkih problema za koje nije poznat algoritam koji pronalazi egzaktno rješenje. Određivanjem ograničenja parametara problema koji se pokušava riješiti definira se prostor mogućih rješenja koji se pretražuje. U većini slučajeva se problem može definirati kao traženje globalnog minimuma ili maksimuma neke funkcije. Postupak rješavanja svodi se na traženje parametara funkcije za koje ona postiže svoj globalni minimum ili maksimum. Funkcija može imati velik broj lokalnih ekstrema koji bi predstavljali zapreku drugim metodama traženja globalnih ekstrema funkcije na primjer tehnici gradijentnog spusta.

Pretraga prostora rješenja započinje iz više točaka od kojih svaka predstavlja jedinku u početnoj populaciji evolucijskog algoritma. Početne točke se izabiru nasumično kako bi se smanjila mogućnost da pretraga završi u lokalnom ekstremu. Svaka jedinka ima svoj genetski materijal predstavljen nekom podatkovnom strukturom koju nazivamo kromosom. Nove jedinke u generaciji nastaju kombinacijom genetskog materijala postojećih jedinki, kroz operator križanja, i unošenjem novog genetskog materijala kroz operator mutacije. Križanje je rekombinacija genetskog materijala dvije ili više jedinki čime nastaje obično jednak broj novih jedinki. Pri tome jedinke koje daju bolje rezultate imaju veće šanse prenijeti svoj genetski materijal od lošijih jedinki. Operator mutacije nasumično mijenja

genetski materijal neke jedinke te na taj način unosi novi genetski materijal u populaciju. Kvalitetu rezultata jedinke određujemo funkcijom dobrote. Korištenjem opisanih operacija nad jedinkama stvaraju se nove jedinke i mijenja se sadržaj populacije odnosno nastaje nova generacija populacije. Generacije se izmjenjuje dok nije ispunjen uvjet zaustavljanja algoritma. Kada je ispunjen uvjet zaustavljanja jedinke s najvećim dobrotama prijavljuju se kao moguća rješenja optimizacijskog problema.

5.1 Predstavljanje rješenja kromosomom

Kako bi mogli koristiti genetski algoritam za traženje rješenja nekog problema potrebno je naći način za predstavljanje mogućih rješenja pomoću skupine gena odnosno kromosoma. Budući da o korištenjom načinu ovisi kako će biti izvedeni operatori križanja i selekcije, odabir načina prikaza rješenja dosta utječe na rezultate genetskog algoritma. Često korištena metoda je prikazivanje mogućih rješenja pomoću niza binarnih znamenki. Svaki gen sadrži niz binarnih znamenki koji opisuje neko svojstvo rješenja. Niz svih gena čini kromosom. Među ostalim postojećim metodama nalaze se i prikaz pomoću cijelih brojeva, prikaz pomoću brojeva sa pomičnim zarezom, prikaz pomoću slova i mnogi drugi.

5.2 Operator mutacije

Operator mutacije unosi novi genetski materijal u populaciju. U slučaju da se operator mutacije ne bi koristio traženje rješenja bilo bi ograničeno genetskim materijalom koji se nalazi u početnoj populaciji. Uz vrlo malu vjerojatnost mutacije moguće je da se novi genetski materijal brže gubi nego stvara jer ga se istiskuje kroz selekciju materijalom postojećih jedinki bolje kvalitete. To može dovesti do toga da pretraga prostora rješenja zaglavi u lokalnom ekstremu. Prevelik operator mutacije brže mijenja genetski materijal jedinki nego što se kvalitetan materijal može spojiti u jednoj jedinci kroz operator križanja. Pretraživanje sve više postaje nasumično traženje rješenja.

Postoji mnogo raznih vrsta mutacijskih operatora, a neki od češće korištenih su:

- promjena bit-a; koristi se na kromosomima sa binarnim prikazom rješenja
- granična mutacija: mijenja gen sa jednom od graničnih vrijednosti

uglavnom se koristi kod kodiranja pomoću cijelih ili brojeva sa pomičnim zarezom

- uniformna: mijenja gen sa nasumičnom vrijednošću po uniformnoj distribuciji
- Gaussova mutacija: mijenja gen sa nasumičnom vrijednošću po Gaussovoj distribuciji

5.3 Operator križanja

Pri križanju se uzimaju dijelovi genetskog materijala jedinki roditelja i kombiniraju se u genetski materijal novih jedinki djece. Pokazalo se da je operacija križanja manje destruktivna ako se za mjesta križanja odabiru geni koji predstavljaju početak ili kraj neke osobine [3, 4]. Bit križanja je kombiniranje dobrih osobina jedinki roditelja, kako bi se mogla stvoriti jedinka dijete s boljim osobinama od svojih roditelja.

Često korišteni operatori križanja:

- Križanje u jednoj točki: pri ovom načinu križanja odabire se jedan gen nakon kojeg dolazi do zamjene svih gena jedne jedinke sa drugom i obratno.
- Križanje u više točaka: može se promatrati kao više uzastopnih križanja u jednoj točki
- Uniformno križanje: za svaki gen u kromosomu se nasumično odabire od kojeg roditelja dolazi

5.4 Operator selekcije

Selekcija je mehanizam koji uzrokuje napredak populacije, tj. povećanje prosječne dobrote jedinke. Njome se određuje jedinke populacije čiji genetski materijal prelaze u sljedeću generaciju. Metoda selekcije jedinki koje preživljavaju mora biti pažljivo odabrana. Ako je selekcija prejaka (npr. loše jedinke uopće ne preživljavaju), postoji šansa da će u populaciji preostati samo jedinke vrlo sličnih karakteristika, čime se efektivno populacija kandidata rješenja zamjenjuje varijacijama jednog kandidata rješenja (koji bi mogao predstavljati lokalni ekstrem funkcije). Tada se kaže da je genetski materijal populacije osiromašen. Ako je pak selekcija preslaba (npr. loše jedinke u gotovo jednakoj mjeri preživljavaju i razmnožavaju se kao i dobre jedinke), tada postupak pretrage vrlo dugo traje i teško dolazi do dobrih jedinki.

Odabir jedinki koje sudjeluju u razmnožavanju (rekombinaciji genetskog materijala operatorom križanja) također je bitan aspekt selekcije. Neki od mehanizama su:

- Proporcionalno razmnožavanje tj. roulette selekcija: broj razmnožavanja neke jedinke proporcionalan je njenoj dobroti
- Odabir po rangu: jedinke populacije sortiraju se silazno po dobroti, kako bi se dobila rang-lista. Broj razmnožavanja neke jedinke obrnuto je proporcionalan njenom rangu u populaciji (što je rang niži, to je veći broj razmnožavanja neke jedinke)
- Turnirski odabir: bira se slučajan broj jedinki iz populacije, a najbolje od njih postaju roditelji nove populacije. Ovaj postupak se ponavlja dok se ne stvori nova populacija.

U slučajevima kada je potrebno najbolje jedinke sačuvati od promjena možemo određeni najbolji postotak trenutne generacije prebaciti u iduću bez primjene operacija mutacije i križanja. Za selekciju koja sadržava takvo pravila kažemo da je selekcija sa elitizmom jer se pogoduje najboljim tj. elitnim jedinkama.

5.5 Uvjet zaustavljanja

Uvjet zaustavljanja određuje u kojem trenutku dolazi do prestanka rada genetskog algoritma odnosno do kraja evolucije. Postoji više načina određivanja kraja postupka evolucije. Standardni uvjeti prekidanja evolucije uključuju:

- Nađeno je rješenje koje zadovoljava kriterije
- Postignut je neki unaprijed zadan broj generacija
- Potrošeno je neko unaprijed zadano procesorsko vrijeme
- Kroz nekoliko generacija nije primijećen napredak u pronađenim rješenjima svake pojedine generacije
- Ručnim pregledom pronađenih rješenja te zaključkom da se daljnjim ostvarivanjem postupka neće postići bolji rezultati
- Kombinacijom već navedenih uvjeta

6. NEAT

Neuroevolucija rastuće strukture, NEAT (engl. Neuroevolution of augmenting topologies), pripada metodama učenja umjetnih neuronskih mreža promjenjive strukture i parametara (engl. Topology and Wright Evolving Artificial Neural Networks – TWEANNs). Za razliku od neuronskih mreža fiksne strukture, ovakvim neuronskim mrežama se osim parametara neuronske mreže, tokom postupka učenja evoluira i struktura neuronske mreže. NEAT pokazuje bolje rezultate nego do tada korištene metode evolucije za učenje neuronskih mreža fiksne strukture na problemu balansiranja štapova. Problem balansiranja štapova često se koristi za ispitivanje uspješnosti algoritma u rješavanju problema učenja s podrškom [5].

Do pojave NEAT-a druga ispitivanja pokazala su da učenje neuronske mreže fiksne strukture evolucijskim algoritmima pokazuje bolje rezultate nego učenje neuronskih mreža promjenjive strukture [6]. Pokazano je da su specifičan operator križanja koji uzima u obzir povijest evolucije strukture, evoluiranje od minimalne struktura neuronske mreže (u početnoj populaciji sve jedinke imaju istu minimalnu strukturu) i dijeljenje populacije na vrste ovisno o strukturi mreže zaslužni za dobre rezultate NEAT-a [5]. Uklanjanje bilo kojeg od ta tri dijela iz NEAT algoritma znatno umanjuje efikasnost učenja.

6.1 Genetsko kodiranje NEAT-a

Prvi korak u genetskom kodiranju algoritma NEAT izvodi se prije stvaranja početne populacije. U minimalnoj strukturi neuronske mreže koja se uči svakom neuronu dodjeljuje se jedinstveni identifikator te se na taj način dobiva i identifikator svake veze među neuronima kao par identifikatora početnog i konačnog neurona te veze. Svaki gen u genomu korištenom u NEAT algoritmu predstavlja jednu vezu u strukturi neuronske mreže odnosno predstavlja jedan parametar neuronske mreže. Veza je jednoznačno označena parom početnog i konačnog neurona. Svaki gen također ima i povijesni identifikator pomoću kojega je moguće pratiti kroz generacije kretanje tog gena. Spomenuti identifikator koristi se prilikom operacija križanja kako bi se osigurala zamjena gena koji su evoluirali iz iste strukture te možemo pretpostavljati da prenosi slične osobine. Osim povijesnog identifikatora, početnog

neurona i konačnog neurona svaki gen još sadrži informaciju da li je aktivan te vrijednost parametra veze u neuronskoj mreži. Način korištenja pojedinih dijelova gena objašnjen je u nastavku.

Kako u početnoj populaciji NEAT algoritma sve jedinke imaju identičnu minimalnu strukturu neuronske mreže genomi svih početnih jedinki su isti do na parametre vrijednosti veza među neuronima. Kada se struktura neuronske mreže ne bi mijenjala kroz generacije, odnosno uz drugačiji operator mutacije u NEAT algoritmu, učenje neuronskih mreža pomoću NEAT-a davalo bi iste rezultate kao i obično učenje neuronske mreže fiksne strukture.

6.2 Operator mutacije u NEAT algoritmu

Operator mutacije u NEAT algoritmu osim promjene vrijednosti parametra veze pojedinog gena može i dodavati nove gene te na taj način mijenjati strukturu neuronske mreže.

Postoje dvije vrste mutacije strukture u NEAT algoritma; dodavanje neurona i dodavanje veze.

Prilikom dodavanja neurona odabire se neki aktivni gen, odnosno veza koja se koristi u trenutnoj strukturi, te se ta veza zamjenjuje sa novim neuronom i dvije nove veze kako bi se ostvarila povezanost koja je prekinuta. U genomu jedinke gen čija veza je zamijenjena označuje se kao neaktivan i dodaju se dva nova gena koja opisuju dvije dodane veze. Parametri dodanih veza postavljaju se na neku nasumičnu vrijednost. Povijesni identifikatori dodanog neurona i gena se postavljaju na nove vrijednosti ili se uzimaju već postojeći povijesni identifikatori ovisno da li je u bilo kojoj drugoj jedinci kroz sve generacije već dodan neuron na tu vezu ili nije.

Kod dodavanja nove veze u strukturu mreže odabiru se dva nepovezana neurona pri čemu, ovisno o vrsti strukture na koju se želimo ograničiti, treba paziti da ne dođe do ciklusa. U genotip mutirane jedinke dodaje se novi gen koji opisuje dodanu vezu. Parametar dodane veze postavlja se na nasumičnu vrijednost. Dodani gen dobiva novi ili već postojeći povijesni identifikator ovisno da li je takva veza već dodana mutacijom u bilo kojoj drugoj jedinki kroz sve generacije do tada.

6.3 Operator križanja u NEAT algoritmu

Operator križanja u NEAT algoritmu nastoji pronaći dijelove genoma koja predstavljaju iste osobine strukture roditelja te izvesti križanje zamjenjujući samo srodne strukture. Na taj način operator križanja postoji znatno manja vjerojatnost da će operacija križanja dovesti do uništavanja neke dobre osobine roditelja.

Nalaženje genoma koji predstavljaju iste osobine strukture roditelje obavlja se pomoću povijesnih identifikatora gena. Operator mutacije koji mijenja strukturu neuronske mreže provjerava da li se u bilo kojoj mutacije kroz sve generacije već pojavio gen koji opisuje istu mutaciju te postavlja povijesni identifikator gena.

Prije obavljanja operacije križanja genomi roditelja postavljaju se tako da se povijesni identifikatori pojedinih gena poklapaju. Takvi geni se nasljeđuju od roditelja sa većom vrijednosti funkcije dobrote ili ako su oba roditelja jednako dobri nasumično se odabire jedan od roditelja od kojeg se nasljeđuju geni. Za križanje ostalih gena koji se razlikuju samo u parametru vrijednosti neuronske veze nije određena neka specifična metoda kojom je potrebno izvesti križanje. Moguće je koristiti standardne metode križanja genetskih algoritama (križanje u jednoj ili više točaka, uniformno križanje ili neki drugi).

6.4 Očuvanje inovacija kroz podjelu na vrste

Dodavanje novih veza odnosno neurona u struktura neuronske mreže obično u početku smanjuje dobrotu jedinke. Kako bi se nova struktura očuvala kroz generacije i dobila priliku da popravi dobrotu jedinke NEAT koristi podjelu jedinki na vrste. Podjelom jedinki na vrste osigurava se očuvanje novih struktura kroz nekoliko generacija što im daje vrijeme potrebno za optimizaciju svojih parametara.

Podjela na vrste često se koristi u optimizaciji multimodalnih funkcija gdje joj je glavna uloga očuvanje raznovrsnosti populacije [9, 10].

Povijesni identifikatori omogućuju da se identificiraju geni koji su nastali iz srodnih struktura odnosno omogućuje jednostavnu usporedbu sličnosti struktura.

U NEAT algoritmu definira se mjera različitosti jedinki δ kao jednostavna linearna kombinacija broja gena koji se razlikuju te prosječne razlike težina srodnih gena. $\delta = \frac{c_1 * E}{N} + \frac{c_2 * D}{N} + c_3 * \bar{W}$ Gdje je E broj gena koji se razlikuju na kraju dužeg genoma, D broj gena koji se razlikuju prije kraja kraćeg genoma, a $\bar{W}$ srednja vrijednost razlike parametara gena koji se poklapaju u oba genoma. N služi za normalizaciju i jednake je broju gena u dužem genomu. Parametri c_1, c_2, c_3 definiraju se prije početka postupka učenja i služe za određivanje koliko pojedine razlike u strukturi odnosno težini veza imaju utjecaja na sličnost to jest razliku jedinki.

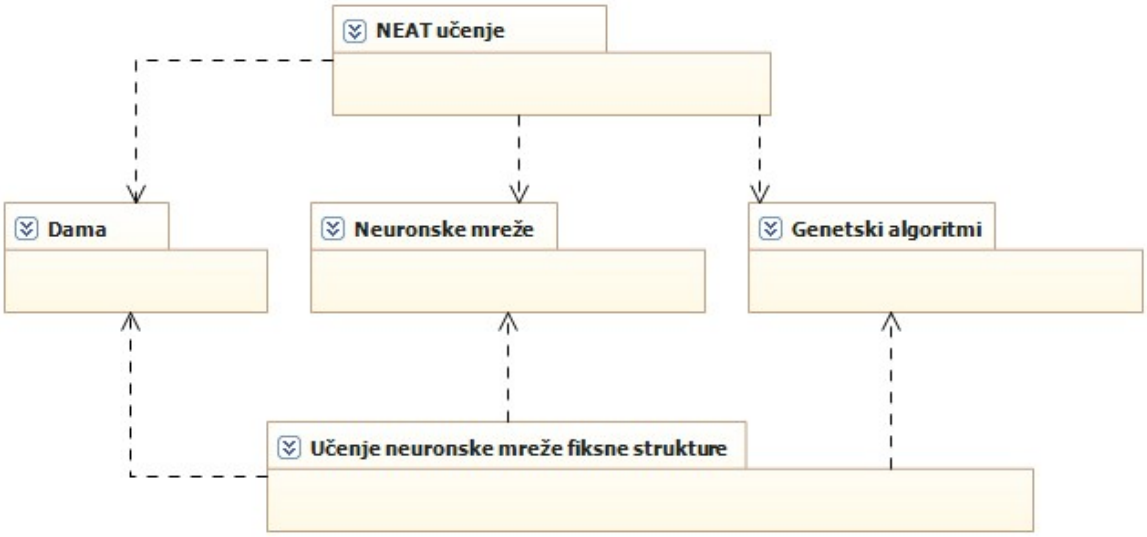
Prije početka postupka učenja također se definira i parametar δ_t koji određuje širinu pojedine vrste odnosno koliko jedinke trebaju biti slične kako bi ih promatrali kao istu vrstu.

Jedinke se dijele u vrste u svakoj generaciji. Jedna po jedna jedinka se uspoređuju sa jedinkama već dodijeljenim u vrste. Ako je različitost jedinki δ manja od δ_t jedinka koja se uspoređuje sa ostalima pridjeljuje se istoj vrsti iz koje je jedinka s kojom je upravo uspoređena. Jedinka koja se ovim postupkom ne dodjeli niti jednoj vrsti postaje jedina jedinka nove vrste.

7. Programska izvedba

Prilikom pisanja programske izvedbe nastojalo se, radi lakšeg razumijevanja i ispitivanja, odvojiti pojedine cjeline u zasebne projekte. Na taj način omogućeno je, kroz korištenje "generic" konstrukata u C#, napraviti program koji je lako nadograditi i ponovno koristiti u drugim projektima. Osim ponovnoj iskoristivosti, velika pažnja dana je i efikasnom korištenju memorije te slobodnih jezgri procesora. Kako bi se osiguralo jednostavnije ispitivanje i ponovljivost, svi parametri bitni za izvođenje programa, uključujući i početne vrijednosti generatora slučajnih brojeva, moguće je postaviti bez ponovnog prevođenja programskog koda.

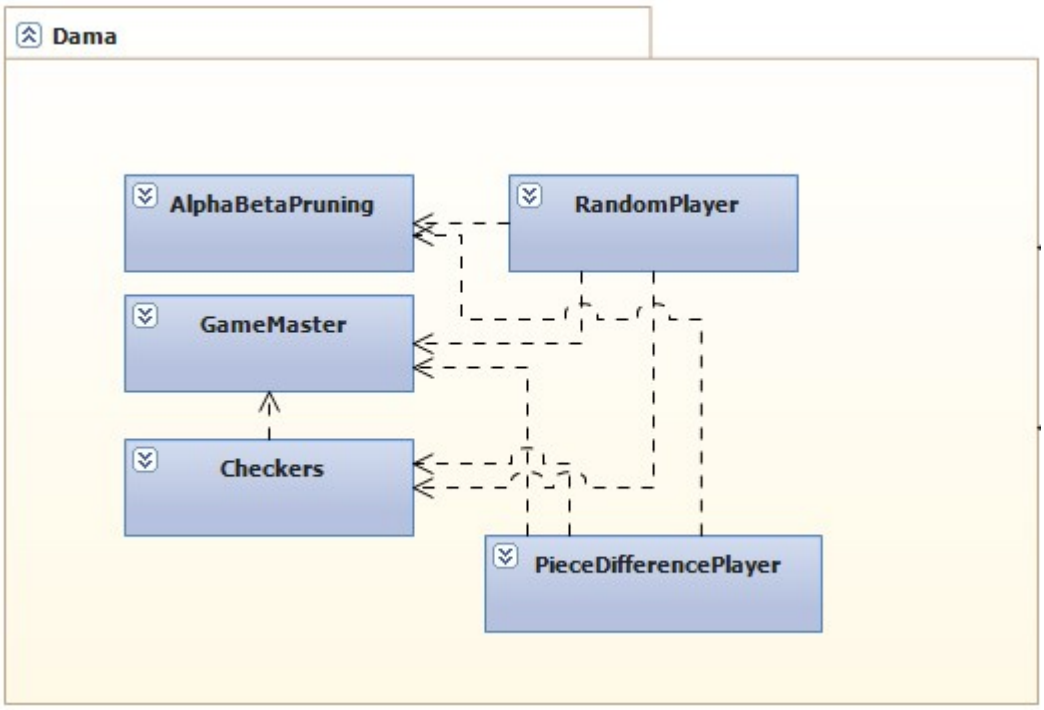
Struktura projekta bit će izložena kroz pet logički zaokruženih cjelina poredanih prema međuovisnost i vremenu nastajanja. [Slika 1: Struktura projekta]



Slika 1: Struktura projekta

7.1 Dama

Cjelina sadrži implementaciju općenitog nadglednika igre, korištene algoritme za pretraživanje stanja, programsku implementaciju pravila dame i jednostavne implementacije igrača koje su korištene za ispitivanje ispravnosti izvođenja [Slika 2: Struktura cjeline dama].



Slika 2: Struktura cjeline dama

7.1.1 Nadglednik igre

Može se koristiti za izvođenje bilo koje igre u kojoj igrači jedan po jedan slijedno izvode akcije te igra time prelazi od početnog do završnog stanja. Osim uvjeta na pravila izvođenja stanje igre mora implementirati sučelje koje omogućava određivanje da li je neko stanje završno. Igrači moraju implementirati sučelje kroz koje mu je moguće predati trenutno stanje igre nad kojima on vrši akcije te vraća novo stanje igre. Nadglednik ne provjerava da li je novo stanje ispravno odnosno ne provjerava da li je igrač varao. Omogućeno je spremanje i vraćanje svih stanja kroz koje je igra prošla od početnog do konačnog stanja.

7.1.2 Algoritam alfa-beta rezanja

Za pretraživanje stabla igre korišten je algoritam alfa-beta rezanja opisan u poglavlju 3.2 uz dodatnu optimizaciju u obliku negamax implementacije. Posebnost negamax implementacije je u korištenju iste evaluacijske funkcije neovisno o tome koji igrač je na potezu, onaj koji minimizira ili onaj koji maksimizira. Evaluacijska funkcija mora vraćati to veću vrijednost što je igrač na potezu u boljoj poziciji odnosno to manju što je igrača na potezu u lošijoj poziciji. Kako bi bilo moguće koristiti ovu implementaciju pretraživanja stanja igre potrebno je samo u stanje igre implementirati metodu kojom je moguće dohvati sva stanja u koja se može preći iz tog stanja jednim potezom.

Dubina pretraživanja stabla predaje se kao parametar prilikom svakog poziva funkcije. Kako je dubinu pretraživanja moguće predati kao parametar, proširivanje implementacije da omogućuje iterativno produbljivanje pretraživanja je vrlo jednostavno. Potrebno je samo dodati metodu koja će pozivati već implementiranu metodu sa inkrementalno rastućim parametrom dubine pretraživanje dok ne istekne neko maksimalno vrijeme određeno za izvođenje poteza. S obzirom da implementacija ne pamti nikakva stanja između dva poziva dodatne optimizacije koje bi omogućile brže rezanje tako što bi prvo išli ispitivati stanja koja su imala velike vrijednosti i povećanom vjerojatnošću rezanja osigurale posjećivanje manjeg broja stanja nisu jednostavno izvedive.

7.1.3 Izvedba pravila igre dama

Velik utjecaj na izvedbu pravila imao je način korištenja u sklopu cijelog projekta. Postupak učenja zahtijevao je odigravanje velike količine partija pa je u konačnici velika pažnja posvećena mogućnosti paralelizacije te što većem iskorištavanju već izračunatih pozicija.

Prva trivijalna implementacija sadržavala je ploču u obliku matrice sa osam polja okomito i vodoravno. Sve operacije računanja mogućih poteza prilikom svakog posjeta stanju izvodile su se iznova. Nakon prvih ispitivanja učenja pokazalo se da program oko 98% vremena provodi u izračunavanju mogućih poteza za pojedinu poziciju.

Kako bi se riješio taj problem te ubrzao postupak učenja uveden je spremnik za već izračunate pozicije unutar jedne partije. Navedenim postupkom došlo je do

znatnog ubrzanja.

Budući da se u izvedenom programu prilikom određivanja dobrote populacije odigrava velik broj partija koje su međusobno nezavisne činilo se kao da je taj dio procesa učenja vrlo jednostavno paralelizirati. Nakon uvođenja više dretvi u proces nije dobiveno nikakvo ubrzanje. Ovisno o broju uvedenih dretvi čak je došlo i do usporenja programa.

Detaljnom analizom utvrđeno je da su dretve gotovo stalno u stanju čekanja te da program većinu vremena provodi stvarajući nove objekte. Broj ispitanih pozicija po generaciji je bio oko dvadeset milijuna i za svaku poziciju bilo je potrebno stvoriti jedan objekt. Kako bi se izbjegla velika količina objekata potrebno je bilo implementirati poziciju kao strukturu. Budući da se u dami koriste samo tamna polja na ploči bilo je nepotrebno spremati sva polja. U konačnici ploča je izvedena kao struktura od četiri broja širine trideset i dva bita, gdje svaki broj predstavlja položaje jedne vrste figura na ploči.

Novi način spremanja pozicije zauzimao je znatno manje memorije te je omogućio stvaranje zajedničkog spremnika za sve izračunate pozicije unutar nekolikog generacija. Zajednički spremnik kojem se pristupalo iz više dretvi zahtijevao je uvođenje kritičnih odsječaka. Iako je dobiveno znatno ubrzanje u odnosu na stanje nakon uvođenja dretvi i dalje nije bilo moguće iskoristiti više od 50% procesora koristeći šest dretvi na procesoru sa šest jezgri.

Vrsta implementacije	Vrijeme odigravanja 100 partija u ms
Prva implementacija sa spremanjem u matricu	63634
Dodano spremanje izračunatih stanja igre	5570
Promjena načina spremanja stanja igre. Prelazak na strukture	1495
Uvođenje više dretvi (šest)	690
Uvođenje novih metoda usklađivanja dretvi	357

Tablica 1: Performanse dame

Naknadno su pažljivo uklonjene sve kritične sekcije. Za sinkronizaciju i ograničenje pristupa među dretvama korištene su nove metode dostupne od .net-a 4.0. Korištenjem memorijskih barijera, aktivnih čekanja i aktivnih zaključavanja potpuno je izbjegnuta zamjena konteksta dretvi te odlazak dretvi u stanje čekanja.

Konačna analiza programa pri korištenju šest dretvi na procesoru sa šest jezgri pokazala je iskorištenje procesora od preko 90%. Program je pritom u stanju sinkronizacije proveo oko 1% vremena. U odnosu na prvu trivijalnu implementaciju postignuto je ubrzanje od nekoliko stotina puta [Tablica 1: Performanse dame].

7.1.4 Jednostavni igrači dame

Za ispitivanje ispravnosti pravila odigravanje implementirane su dvije vrste igrača, igrač nasumičnih poteza i igrač razlike broja figura. Obje vrste igrača koriste algoritam alfa-beta rezanja za pretraživanje stanja igre. U evaluacijskoj funkciji oba igrača prvo se provjerava je li pozicija pobjednička ili gubitnička te se takvoj poziciji vraća maksimalna pozitivna to jest negativna vrijednost. Ako pozicija nije završna, igrač nasumičnih poteza vrača nasumičnu vrijednost između maksimalne pozitivne i maksimalne negativne vrijednosti, dok igrač razlike broja figura vraća razliku broja figura između igrača na potezu i protivničkog igrača.

7.2 Neuronske mreže

Kako je od početka bilo jasno da će se kroz projekt koristiti različite strukture neuronskih mreža za istu funkciju (kao evaluacijska funkcija u algoritmu alfa-beta rezanja), u projektu neuronske mreže definirano je sučelje koje omogućuje da se veliki dijelovi programa korišteni za izvedbu igrača dame ponovno iskoriste.

Sučelje od implementacije neuronske mreže zahtjeva implementaciju jedne metoda koja prima listu vrijednosti postavljene na ulaze neuronske mreže i vraća listu vrijednosti dobivene na izlazima neuronske mreže.

Osim sučelja za rad sa neuronskim mrežama u ovom djelu sadržana je implementacija višeslojnog perceptrona kojoj je moguće definirati strukturu prilikom konstrukcije. Za aktivacijsku funkciju u neuronima ove implementacije korištena je funkcija tangens hiperbolni.

U ranim verzijama ova implementacija je bila dovoljna te su se različite strukture mreža dobivene NEAT postupkom učenja pretvarale u perceptron sa velikom količinom veza težine nula. Kako je struktura dobivena NEAT postupkom učenja rijetko povezana dobar dio težina u perceptronu bila je nula te se prilikom računanja izlaza izvodio velik broj nepotrebnih operacija. Osim problema se performansama

radi velikog količina nepotrebnog računanja program za pretvaranje NEAT struktura u perceptrone je bio dosta kompleksan jer je morao uzeti u obzir i mogućnost da NEAT stvori strukturu u kojoj su na primjer povezani neuroni u prvom skrivenom sloju sa neuronima u trećem skrivenom sloju što kod perceptrona nije moguće.

Radi navedenih problema odustalo se od pokušaja pretvaranja NEAT struktura neuronskih mreža u postojeću implementaciju perceptrona te je napravljena nova implementacija za neuronske mreže proizvoljne strukture s ograničenjem da u strukturi ne smije biti ciklusa.

Kako su u implementaciji neuronske mreže proizvoljne strukture veze među neuronima spremljene u listama i ne računaju se slijedno izlazi jednog po jednog skrivenog sloja, zadržana je i postojeća implementacija perceptrona jer je za gusto povezane strukture neuronskih mreža davala bolje performanse.

7.3 Genetski algoritmi

Nastojalo se napraviti što općenitiju implementaciju osnovnog rada genetskog algoritma tako da se lako mogu mijenjati operatori mutacije, križanja, selekcije, uvjet zaustavljanja i funkcija dobrote.

Kako bi se smanjio broj promjenjivih parametara, a ujedno zadržala mogućnost izmjene svih bitnih dijelova genetski algoritam podijeljen je na četiri dijela: Evaluator, Selector, Producer i osnovni dio genetskog algoritma koji je zadužen za pokretanje izmjenu generacija i zaustavljanje.

Evaluator je zadužen za određivanje dobrote jedinki generacije i ujedno za provjeravanje uvjeta zaustavljanja bez obzira da li je uvjet zaustavljanja definiran kao dostignuće prosječne dobrote generacije, vremena izvođenja ili bilo koji drugi. Selector na temelju dobrote jedinki i same populacije vraća listu parova jedinki koji će postati roditelji iduće generacije. Producer na temelju dobrote jedinki i liste parova roditelja dobivenih od selectora stvara iduću generaciju. Osnovni dio brine se samo za put podataka između navedenih dijelova i vraćanja rezultata pozivatelju genetskog algoritma.

Osim samog genetskog algoritma implementirane su neke općenite verzije često korištenih operatora na primjer roulette selekcija.

7.4 Učenje neuronske mreže fiksne strukture

Za implementaciju učenja neuronske mreže fiksne strukture korištene su sve do sada navedene cjeline projekta. Za implementaciju igrača dame korišten je algoritam alfa-beta rezanja sa neuronskom mrežom tipa perceptron kao evaluacijskom funkcijom.

Svaki perceptron činio je jednu jedinku u genetskom algoritmu. Parametri neuronske mreže prikazani su u genomu kao brojevi s pomičnim zarezom. Nije korišteno tradicionalno binarno kodiranje već se pojedini parametar (broj s pomičnim zarezom) promatrao kao jedan gen. Križanje je izvedeno pomoću operatora križanja u jednoj točki. Vjerojatnost križanja roditelja moguće je zadati kao parametar, a optimalna vrijednost mu je određivana ispitivanjima prije samog postupka učenja. Za mutaciju je definirana vjerojatnost mutacije jedinke kao vjerojatnost da će nakon operacije mutacije jedinka imati barem jedan mutirani gen. Mutacija pojedinog gena izvedena je kao zamjena postojeće vrijednosti broja sa pomičnim zarezom novom nasumičnom vrijednošću. Vjerojatnost mutacije jedinke moguće je zadati kao parametar, a optimalna vrijednost je određivana ispitivanjima prije samog postupka učenja. Za selekcijsku funkciju korištena je ranije implementirana roulette selekcija.

Izvede su dvije različite funkcije dobrote. Jedna funkcija dobrote izvedena je korištenjem igrača koji vuče nasumične poteze. Jedinka bi odigravala manji broj partija protiv tog igrača te bi za svaku pobjedu dobila dva, a za izjednačen rezultat jedan bod. Ukupan broj bodova određivao je dobrotu jedinke. Nakon prvih par ispitivanja postupka učenja ta je funkcija dobrote odbačena jer su jedinke vrlo brzo počele pobjeđivati te bi funkcija dobrote za većinu jedinki vračala maksimalnu vrijednosti i učenje je zaustavljeno. Funkcija dobrote korištena u kasnijim ispitivanjima izvedena je pomoću švicarskog turnirskog sistema opisanog kasnije. Sve jedinke generacije sudjelovale bi u turniru. Rezultat jedinke na kraju turnira bio je ujedno i dobrota jedinke.

Švicarski turnirski sistem često je korištena metoda za određivanje koje partije će se odigrati na turniru i određivanje konačnog rezultata. Turnir se izvodi u više krugova. Broj krugova potreban Kako bi se dovoljno dobro rangirali igrači jednak je $\log(N)$ gdje je N broj igrača koji sudjeluju u turniru. U prvom krugu turnira svi igrači imaju jednak broj bodova i nalaze se u istoj skupini. Igrači se nasumično podjele u

parove i odigraju partiju prvog kruga te dobiju određen broj bodova. U drugom i svakom sljedećem krugu igrači se podjele u skupine po broju bodova te im se protivnik za taj krug bira iz iste skupine. Nakon što se završe svi krugovi rang igrača na turniru jednak je njegovim osvojenim bodovima.

7.5 NEAT učenje

Pri implementaciji NEAT učenja trebalo je učinkovito riješiti problem praćenja i dodjeljivanja povijesnih identifikatora te naći način kako uz malu računsku složenost izbjeći pojavu ciklusa u strukturi mreže do kojih može doći operatorom mutacije. Oba problema riješena su uvođenjem posebne jedinke koja je u genotipu sadržavala sve gene koji su se pojavili u bilo kojem trenutku kroz postupak učenja. Ta jedinka je također imala uz svaki neuron zabilježenu njegovu maksimalnu udaljenost od. Prilikom operacije mutacije provjeravano je da li takav gen već postoji u posebnoj jedinci te ako bi se našao sve daljnje promjene bile su nepotrebne odnosno mutacija je bila dozvoljena. U slučaju da se radilo o mutaciji koja dodaje neuron u strukturu i to takav koji prije nije postojao dodan je novi gen te su preračunate sve dubine neurona u strukturi posebne jedinke. Kod mutacije koja dodaje vezu u strukturu, a da gen koji opisuje takvu mutaciju ne postoji u posebnoj jedinci, jedino potrebno za određivanje da li je takva mutacija dozvoljena, to jest da li će i nakon dodavanje te veze struktura ostati bez ciklusa, bilo je provjeriti da li je maksimalna udaljenost od početnog sloja krajnjeg neurona te veze veća od maksimalne udaljenosti od početnog sloja početnog neurona te veze.

Za operator križanja zajedničkih gena korišten je operator križanja u jednoj točki, isto kao i slučaju učenja neuronske mreže fiksne strukture, dok je dio operatora križanja koji mijenja strukturu implementiran kako je opisano u poglavlju 6. o NEAT algoritmu.

8. Provođenje ispitivanja

Ispitivanje se provodilo u dva dijela. U prvom dijelu traženi su parametri vjerojatnosti mutacije i križanja genetskog algoritma koji su kroz određeni broj generacija dali najbolje rezultate. Nakon određivanja najboljih vrijednosti parametara mutacije i križanja, provedeno je ispitivanje brzine i kvalitete učenja neuronskih mreža sa fiksnom strukturom te ispitivanje brzine i kvalitete učenja neuronskih mreža korištenjem NEAT algoritma.

Kako je navedeno ranije, za funkciju dobrote korišten je švicarski turnirski sistem. Budući da se dobrota jedinke kroz švicarski turnirski sistem određivala samo u odnosu na ostale jedinke u generaciji bilo je potrebno odabrati neku vanjsku mjeru kojom bi odredili napredovanje generacija. Za tu namjenu korištena je jedna od pomoćnih izvedbi igrača dame sa jednostavnom heurističkom funkcijom. Konkretno, korištena je izvedba sa algoritmom alfa-beta rezanja kome je dubina pretraživanja bila postavljena na tri. Za evaluacijsku funkciju nezavršnih stanja korištena je razlika broja figura na ploči. Tu implementaciju ćemo u daljnjem tekstu zvati vanjski igrač.

Kako bi se odredila uspješnost učenja, to jest kako bi se utvrdilo napredovanje kroz generacije, svakih deset generacija izvođeno je vrednovanje generacije. Prilikom postupka vrednovanja svaka jedinka odigrala je jednu partiju protiv vanjskog igrača. Svaka pobjeda jedinke vrednovana je sa dva boda, a neriješena partija sa jednim bodom. Kroz sva ispitivanja korištena je veličina populacije od sto jedinki.

Maksimalna moguća uspješnost generacije iznosila je dvjesto bodova, a minimalna nula bodova. Prilikom izvođenja ispitivanja za određivanje parametara genetskog algoritma za kriterij zaustavljanja korišten je broj generacija. Radi različite računske kompleksnosti učenja neuronske mreže fiksne strukture i neuronske mreže promjenjive strukture za kriterij zaustavljanja prilikom ispitivanja učenja korišten je vremenski kriterij. Na taj način omogućena je usporedba dva pristupa ne samo po brzini učenja kroz broj generacija nego i s obzirom na ukupnu potrošnju vremena potrebnog za učenje.

Sva ispitivanja izvođena su na računalu sa šest jezgri koje za vrijeme izvođenja ispitivanja nije bilo opterećeno drugim procesima.

8.1 Određivanje parametra vjerojatnosti mutacije jedinke

Za određivanje najbolje vjerojatnosti mutacije jedinke korišteno je učenje neuronske mreže fiksne strukture. Učena neuronska mreža imala je trideset i dva ulaza, jedan izlaz te pet neurona u skrivenom sloju. Kako u tom trenu nije bila poznata niti najbolja vrijednost vjerojatnosti križanja, parametar vjerojatnosti križanja postavljen je na 0,6.

Interval u koje se tražila najbolja vrijednost za parametar vjerojatnosti mutacije postavljen je na interval od 0,001 do 0,7. U prvom dijelu intervala, za vrijednosti manje od 0,1 koristio se logaritamski korak pomicanja parametra dok su u dijelu intervala većem od 0,1 ispitna vrijednost parametra kretala linearno sa korakom od 0,1 [Tablica 2: Ispitni parametri vjerojatnosti mutacije].

Vjerojatnost mutacije	0,001	0,01	0,1	0,2	0,3	0,4	0,5	0,6	0,7
Vjerojatnost križanja	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6

Tablica 2: Ispitni parametri vjerojatnosti mutacije

Za kriterij zaustavljanja učenja korišten je broj generacija. Kriterij zaustavljanja postavljen je na tisuću. Uspješnost učenja ovisi i o kvaliteti početne populacije koja se nasumično generira. Kako bi se umanjila vjerojatnost da za neku vrijednost parametara slučajno dobijemo bolje rezultate sva ispitivanja ponovljena su deset puta sa različitim početnim populacijama te je uzeta prosječna vrijednost dobivenih rezultata.

8.2 Određivanje parametra vjerojatnosti križanja

Kao i kod određivanja parametra vjerojatnosti mutacije korišteno je učenje neuronske mreže fiksne strukture. Struktura mreže na kojoj je ispitivano učenje imala je kao i kod određivanja parametra mutacije strukturu koja se sastojala od trideset i dva ulaza, jednog izlaza i pet neurona u skrivenom sloju. U ovom trenutku već je bila poznata najbolja vrijednost parametra vjerojatnosti mutacije te je ta vrijednost korištenja prilikom ovih ispitivanja.

Područje pretraživanja za najboljom vrijednosti parametra vjerojatnosti križanja ograničeno je od 0,1 do 0,9. Kroz cijeli interval pretraživanja vrijednost ispitivanog parametra pomicala se linearno sa korakom 0,1 [Tablica 3: Ispitni parametri vjerojatnosti križanja].

Vjerojatnost mutacije	0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2
Vjerojatnost križanja	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9

Tablica 3: Ispitni parametri vjerojatnosti križanja

Kao i prilikom prošlog ispitivanja za kriterij zaustavljanja bio je odabran broj generacija te je bio postavljen na tisuću.

Iz istih razloga kao i kod određivanja parametra vjerojatnosti mutacije sva ispitivanja ponovljena su deset puta sa različitim, nasumično generiranim, početnim populacijama te su korištene prosječne vrijednosti dobivenih rezultata.

8.3 Učenje igrača dame sa fiksnom strukturom neuronske mreže

Prilikom učenja igrača dame sa fiksnom strukturom neuronske mreže korištena je ista struktura mreže kao i u ispitivanjima najbolje vrijednosti parametara vjerojatnosti križanja i mutacije. Neuronska mreža imala je trideset i dva ulaza, jedan izlaz i pet neurona u skrivenom sloju. Parametri vjerojatnosti mutacije i križanja postavljeni su na najbolje vrijednosti ustanovljene u prethodnim ispitivanjima. Igrač je koristio algoritam alfa-beta rezanja do dubine tri. Vanjski igrač, korišten za mjerenje napretka kroz generacije također je koristio algoritam alfa-beta rezanja do dubine tri.

Korišten je vremenski kriterij zaustavljanja genetskog algoritma. Kriterij zaustavljanja bio je postavljen na jedan dan.

8.4 Učenje igrača dame sa promjenjivom strukturom neuronske mreže

Za učenje igrača dame sa promjenjivom strukturom neuronske mreže korišten je NEAT algoritam opisan u poglavljima 6 i 7.5. Igrač je koristio algoritam alfa-beta rezanja do dubine tri koji za evaluacijsku funkciju nezavršnih stanja koristio neuronsku mrežu promjenjive strukture. Vanjski igrač, korišten za mjerenje napretka

kroz generacije također je koristio algoritam alfa-beta rezanja do dubine tri.

Parametri vjerojatnosti mutacije i križanja postavljene su na isti vrijednosti koje su korištene kod učenja igrača dame sa fiksnom strukturom neuronske mreže. Ostali parametri potrebni za rad NEAT algoritma postavljeni su na vrijednosti korištene u radu koji predstavlja NEAT metodu [5] [Tablica 4: NEAT Parametri].

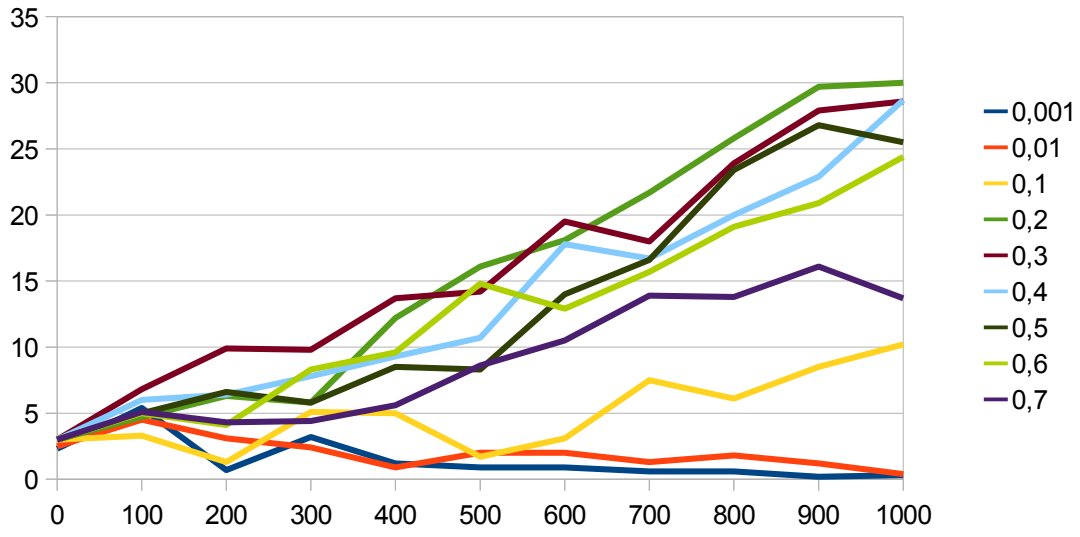
C1	1
C2	1
C3	1
δt	4
Vjerojatnost dodavanja veze	0,3
Vjerojatnost dodavanja neurona	0,03

Tablica 4: NEAT Parametri

9. Rezultati

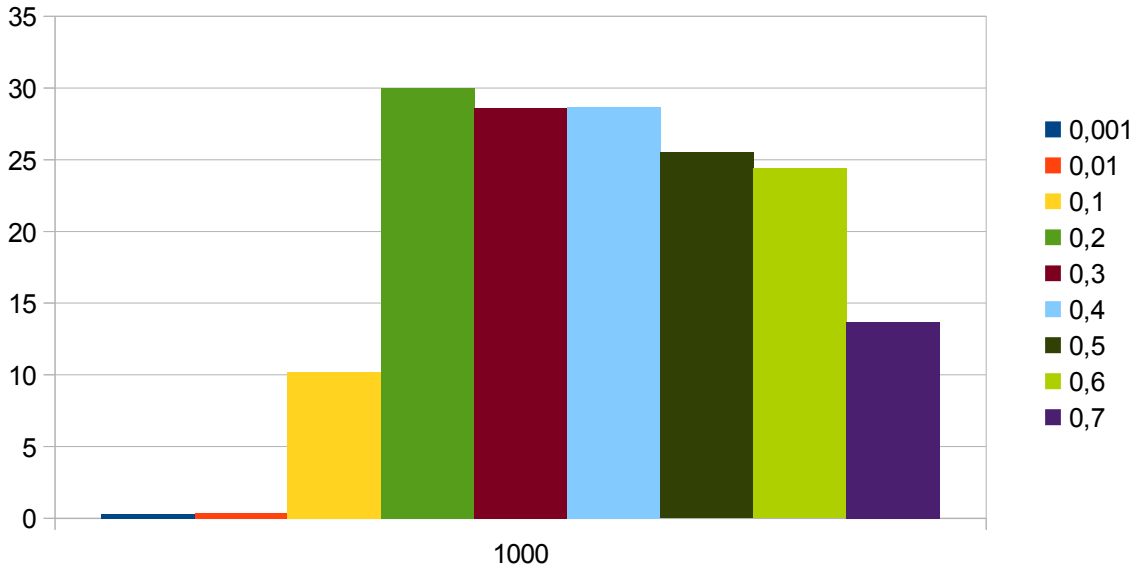
9.1 Rezultati određivanja parametra vjerojatnosti mutacije jedinke

Iz rezultata [Slika 3: Rezultati određivanja vjerojatnosti mutacije] vidljivo je da za vrlo male vrijednosti mutacije napredak ne postoji. Takvi rezultati su očekivani budući da bez mutacije ne dolazi do stvaranja novog genetskog materijala dok se stari gubi selekcijom. Kroz par generacija najbolje jedinke iz početne populacije preuzmu cijelu populaciju i napredak staje. Kako je vanjski igrač koji je korišten za određivanje prosječne dobrote generacije dosta jak najbolje jedinke početne generacije nemaju gotovo nikakve izglede za osvajanje bodova. To potvrđuje i graf genetskog algoritma koji je imao parametar vjerojatnosti mutacije postavljen na 0,001.



Slika 3: Rezultati određivanja vjerojatnosti mutacije

Sa rastom vjerojatnosti mutacije od minimalnih raste i brzina napredovanja jedinki te, kao što se vidi iz priloženog grafa [Slika 4: Dobrota tisućite generacije ovisno o parametru mutacije], najbolje i prilično slične rezultate ostvaruju genetski algoritmi sa vjerojatnošću mutacije postavljenom na 0,2 odnosno 0,3. Budući da razlika u postignutim rezultatima između te dvije vrijednosti parametra nije velika, a manja vjerojatnost mutacije osigurava bržu konvergenciju odabran je parametar vjerojatnosti mutacije jedinke od 0,2.

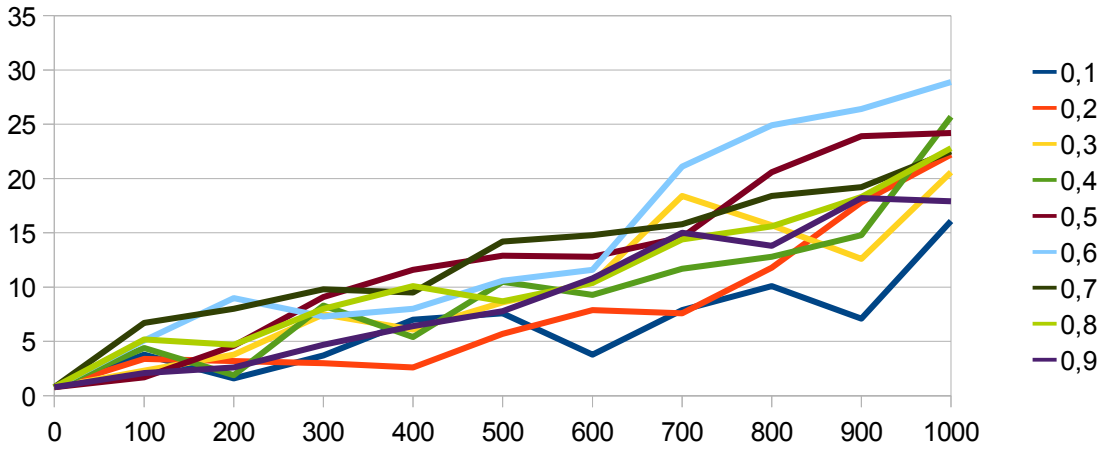


Slika 4: Dobrota tisućite generacije ovisno o parametru mutacije

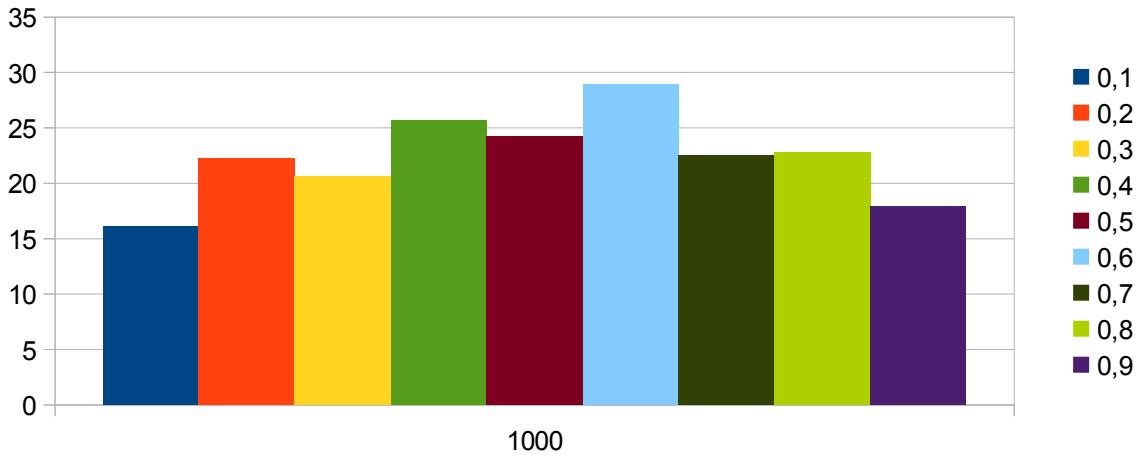
Sa daljnjim rastom vrijednosti mutacije postignuti rezultati se smanjuju što je u skladu sa očekivanim rezultatima. Velik iznos parametra vjerojatnosti mutacije unosi veću destrukciju u genome svih jedinki koje nemaju vremena selekcijom i operacijom križanja popraviti napravljenu štetu.

9.2 Rezultati određivanja parametra vjerojatnosti križanja

Rezultati dobiveni za različite vjerojatnosti križanja ne daju toliko jasno razlučiva dobra odnosno loša ispitivanja kao što je to bio slučaj za vjerojatnost mutacije jedinke [Slika 5: Rezultati određivanja vjerojatnosti križanja]. Na grafu [Slika 6: Dobrota tisućite generacije ovisno o parametru križanja] je vidljivo da vjerojatnosti križanja od 0,5 do 0,7 daju vrlo slične rezultate. Kako se čini da su ipak malo bolji rezultati za vjerojatnost križanja od 0,6 ta vrijednost je korištena u daljnjim ispitivanjima.



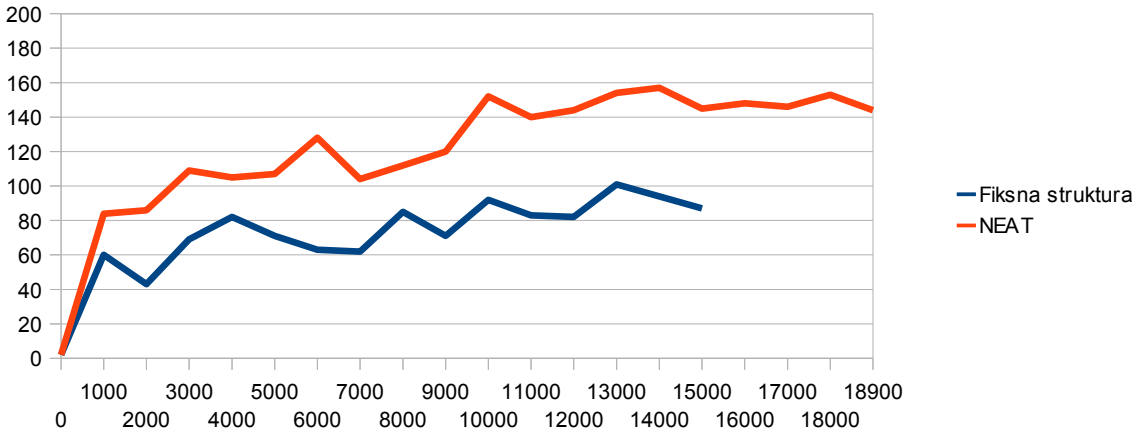
Slika 5: Rezultati određivanja vjerojatnosti križanja



Slika 6: Dobrota tisućite generacije ovisno o parametru križanja

9.3 Rezultati učenja neuronski mreža

Dobiveni rezultati [Slika 7: Rezultati učenja neuronske mreže fiksne strukture i NEAT metode] jasno pokazuju prednost NEAT algoritma pri učenju neuronskih mreža u odnosu na genetski algoritam kojim učimo neuronske mreže fiksne strukture. Time je potvrđena tvrdnja o prednosti NEAT metode iz rada u kojem je NEAT predstavljen [5].



Slika 7: Rezultati učenja neuronske mreže fiksne strukture i NEAT metode

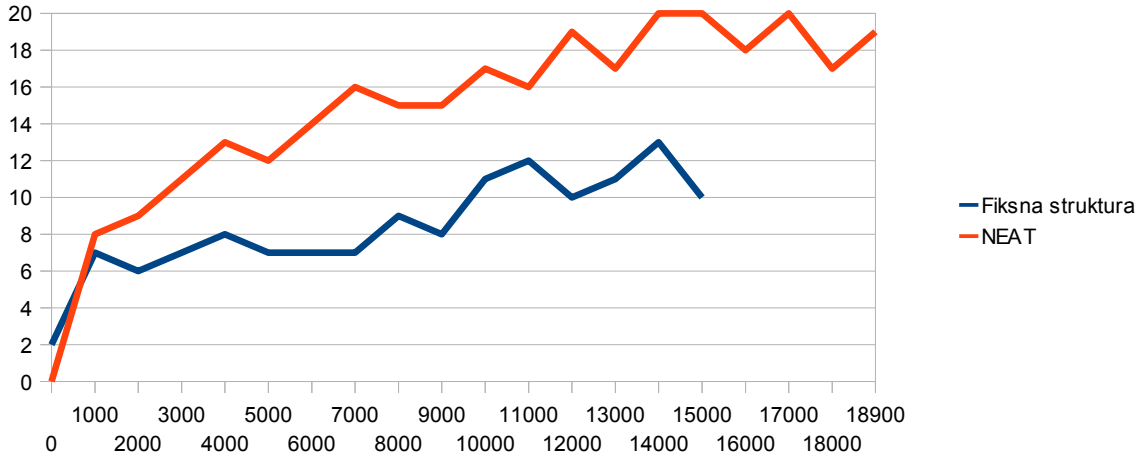
Vidljivo je da učenje postoji kod oba algoritma. U oba slučaja kao kriterij zaustavljanja korišten je vremenski period od jednog dana. NEAT metoda je u tom vremenu uspjela generirati skoro dvadeset posto više generacija. Razlog tome je uglavnom radi jednostavnije strukture neuronskih mreža na početku NEAT algoritma. U neuronskoj mreži jednostavnije strukture sa manjim brojem parametara potreban je manji broj operacija brojeva s pomičnim zarezom kako bi za neke ulazne podatke dala rezultat. Kako se većina vremena potrebna za stvaranje nove generacije provodi u odigravanju partija dame kako bi se odredila dobrota jedinki, tako je broj poziva izračunavanja izlaza neuronske mreže vrlo velik. Iako NEAT algoritam ima računski zahtjevnije operatore mutacije i križanja, broj poziva tih operacija je zanemariv u odnosu na broj poziva izračunavanja neuronske mreže. Radi prirode problema ne možemo biti sigurni da li dobiveni rezultat, prema kojem je NEAT algoritam brži pri izračunavanju generacije, vrijedi i za druge probleme.

Kako su strukture neuronski mreža dobivene NEAT algoritmom vrlo nepravilne u

odnosu na pravilnu strukturu perceptrona korištenu pri učenju mreža fiksne strukture, usporedba računske moći dobivenih struktura nije jednostavno izvediva. Ako grubo procijenimo računsku moć strukture neuronske mreže kao proporcionalnu sa brojem veza možemo zaključiti da najbolje dobivene jedinke NEAT algoritmom imaju znatno manju računsku moć u odnosu na odabrani perceptron. NEAT jedinke imaju nešto veći broj neurona, u prosjeku osam skrivenih neurona, nego što je to slučaj kod odabranog perceptrona koji ima pet skrivenih neurona i oko stotinu veza, dok odabrani perceptron ima sto šezdeset i pet veza.

Primjećeno je da se kroz generacije odigrava sve veći broj neriješenih partija. Uz veliki broj neriješenih partija švicarski turnirski sistem daje vrlo malu razliku među pojedinim igračima iz čega slijedi i mala razlika u dobroti pojedinih jedinki populacije. Kako je korištena selekcija pri kojoj je vjerojatnost da jedinka prenese svoj genetski materijal u iduću generaciju proporcionalna njezinoj dobroti, mala razlika u dobroti jedinki uzrokuju slabiju kvalitetu selekcije odnosno usporava učenje.

Prosječna dobrota generacije niti u jednom korištenom algoritmu ne postiže maksimalni rezultat. Ako promatramo samo deset posto najboljih jedinki svake generacije, NEAT algoritam dostiže najveći mogući rezultat koji je u ovom slučaju dvadeset bodova [Slika 8: Rezultati najboljih deset posto populacije].



Slika 8: Rezultati najboljih deset posto populacije

10. Zaključak

Cilj ovog diplomskog rada bio je istražiti područje učenja umjetnih neuronskih mreža s naglaskom na metode učenja koje pripadaju skupini evolucijskih algoritama, te ih usporediti s obzirom na brzinu izvođenja i konvergencije. U tu svrhu izgrađen je sustav koji standardnim metodama genetskih algoritama uči neuronsku mrežu da ocjenjuje kvalitetu pozicije u igri dama. Osim toga, sustavu je dodana i mogućnost učenja NEAT metodom. Najveća razlika među izgrađenim metodama odnosi se na strukturu mreže koja je u prvoj metodi zadana dok se u NEAT metodi mijenja. Navedene dvije metode uspoređene su po brzini i kvaliteti učenja. Rezultati pokazuju prednost NEAT metode kako po brzini učenja tako i po dobivenim rezultatima najboljih jedinki. Time su potvrđene tvrdnje iznesene u radu koji predstavlja NEAT metodu [5].

Iz dobivenih rezultata samo je djelomično moguće objasniti koji dijelovi NEAT algoritma uzrokuju prednost u učenju. Jasno je da NEAT početnu prednost dobiva radi manje početne dimenzije prostora u kojem se traže rješenja, ali za otkrivanje zašto u konačnici dobiva bolje rezultate potrebna je detaljnija analiza i veći broj ispitivanja.

11. Literatura

[1] Schaeffer, Jonathan: "Checkers Is Solved". Science 317, September 2007

[2] Russell, Stuart J.; Norvig, Peter: "Artificial Intelligence: A Modern Approach" (2nd ed.), Upper Saddle River, New Jersey: Prentice Hall, 2003

[3] Koza, J. R.: "Genetic Programming: On the Programming of Computers by Means of Natural Selection", MIT Press, 1992.

[4] Michalewicz, Z.: "Genetic Algorithms + Data Structures = Evolution Programs", 2nd edition, Springer-Verlag Berlin / Heidelberg, 1994

[5] Kenneth O. Stanley and Risto Miikkulainen: "Evolving Neural Networks Through Augmenting Topologies". Evolutionary Computation 10, 2002

[6] F. Gomez and R. Miikkulainen: "Solving non-Markovian control tasks with neuroevolution". In Proceedings of the 16th International Joint Conference on Artificial Intelligence, Denver, CO, 1999. Morgan Kaufmann.

[7] D. E. Moriarty: "Symbiotic Evolution of Neural Networks in Sequential Decision Tasks". PhD thesis, Department of Computer Sciences, The University of Texas at Austin, 1997.

[8] D. E. Moriarty, Risto Miikkulainen: "Eficient reinforcement learning through symbiotic evolution". Machine Learning, 1996.

[9] S. W. Mahfoud.: "Niching Methods for Genetic Algorithms". PhD thesis, U. of Illinois at Urbana-Champaign, May 1995.

[10] M. A. Potter and K. A. De Jong: "Evolving neural networks with collaborative species". In Proceedings of the 1995 Summer Computer Simulation Conference, 1995.

12. Sažetak

U radu su opisani osnovni elementi, operatori te način rada genetskih algoritama. Također je opisan način rada te različite metode učenja umjetnih neuronskih mreža. Posebna pažnja posvećena je korištenju genetskih algoritama za učenje neuronskih mreža kroz postupak učenja s podrškom. Osim korištenja genetskih algoritama za učenje neuronskih mreža fiksne strukture opisane su i specifičnosti NEAT algoritma koji kroz postupak učenja mijenja strukturu neuronske mreže.

Usporedba različitih metoda učenja neuronskih mreža pomoću genetskih algoritama izvedena je na problemu igre dama. Implementirano je učenje neuronske mreže tipa perceptron korištenjem genetskih algoritama te neuronske mreže promjenjive strukture korištenjem NEAT algoritma.

Analizirani su rezultati dobiveni korištenjem navedenih metoda te je uspoređena uspješnost implementiranih algoritama.

Rezultati pokazuju prednost korištenja NEAT algoritma na učenju neuronskih mreža promjenjive strukture.

Ključne riječi:

evolucijski algoritmi, neuronske mreže, učenje s podrškom, evoluiranje strukture, NEAT, dama

Abstract

Title:

Evolutionary neural network training

This document describes basic elements, operators and workings of genetic algorithms. There is also a brief overview of artificial neural networks and different learning methods with a special emphasis on using genetic algorithms for reinforcement learning of artificial neural networks. NEAT algorithm is used to represent evolutionary algorithms which change neural network structures during the learning process.

Playing checkers is used as a problem on which different neural network learning algorithms are used. Evolutionary algorithm for training of perceptron neural network and NEAT algorithm for training of neural networks with changing structures have been implemented.

Results analysis shows advantages of using NEAT algorithm for neural network training on reinforcement learning problems.

Keywords:

evolutionary algorithms, artificial neural networks, reinforcement learning, structural evolution, NEAT, checkers