SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 353

Optimiranje neuronskih mreža u računalnim igrama korištenjem evolucijskih algoritama

Marta Knežević

Zagreb, svibanj 2022.

SADRŽAJ

1. Uvod **1**

2. Primjena neuroevolucije u računalnim igrama **2**

2.1. Faktori razvoja strategije igre 4

2.1.1. Funkcija dobrote 4

2.1.2. Faktor raznolikosti 5

2.1.3. Podaci neuronske mreže 5

2.1.4. Parametri genetskog algoritma 5

3. Genetski algoritam NEAT **6**

3.1. Kodiranje genoma 7

3.2. Mutacije 9

3.3. Križanje 10

3.4. Očuvanje inovacije 11

3.5. Minimizacija topologije 14

4. Programska knjižnica NEAT-Python **16**

4.1. Konfiguracijska datoteka 16

5. Programska knjižnica Pygame **18**

5.1. Grafičko sučelje 18

6. Praktičan rad **19**

6.1. Računalna igra *Snake* 19

6.1.1. Programsko ostvarenje 19

6.1.2. Parametri i treniranje 20

6.1.3. Rezultati 22

6.2. Računalna igra *Flappy bird* 29

6.2.1. Programsko ostvarenje 29

6.2.2. Parametri i testiranje 30
6.2.3. Rezultati 31

7. Zaključak **34**

Literatura **35**

1. Uvod

Od samih početaka razvoja umjetne inteligencije i evolucijskog programiranja, njihova primjena započela je i u području računalnih igara. Težnja da se omogući igranje protiv „pametnog“ igrača doprinijela je napretku umjetne inteligencije u tom području. Uz kontinuiran proces optimizacije i nadograđivanja stvoreni su brojni programi koji imitiraju ljudsko igranje računalnih igrara.

U okviru ovog rada prikazani su osnovni koncepti primjene neuroevolucije u području računalnih igara te je dan uvid u algoritam NEAT (engl. *NeuroEvolution of Augmenting Topologies*).

Drugo poglavlje pod nazivom *Primjena neuroevolucije u računalnim igrama* opisuje korištenje neuroevolucije i način rada u području računalnih igara te prednosti i mane korištenja neuroevolucije. Treće poglavlje pod nazivom *Genetski algoritam NEAT* detaljno opisuje funkcionalnosti algoritma NEAT. Navode se izazovi s kojima se susreću algoritmi za razvoj topologija neuronske mreže te načini kako iste rješava algoritam NEAT. Opisan je način evoluiranja neuronske mreže kroz funkcionalnosti mutacije, križanja i očuvanja inovacije. U četvrtom poglavlju pod nazivom *Programska knjižnica NEAT-Python* opisuje knjižnicu programskog jezika Python pod nazivom NEAT-Python. Knjižnica obuhvaća metode za evoluiranje proizvoljnih neuronskih mreža. Dan je pregled konfiguracijske datoteke te je opisan način rada temeljnih funkcija knjižnice. Peto poglavlje *Programska knjižnica Pygame* opisuje knjižnicu programskog jezika Python pod nazivom Pygame koja se koristi pri izradi računalnih igara. Šesto poglavlje *Eksperimentalni dio* opisuje programski dio završnog rada. Opisane su dvije računalne igre napravljene korištenjem prethodno navedenih knjižnica. Cilj izrade računalnih igara je primjena neuroevolucije kroz algoritam NEAT kako bi se napravio računalni igrač koji samostalno i uspješno igra računalne igre. Opisane su primjene metoda evolucijskog programiranja u konkretnim računalnim igrama te je dan pregled rezultata u ovisnosti o modificiranim parametrima korištenima u računalnim igrama.

2. Primjena neuroevolucije u računalnim igrama

Prvotna primjena umjetne inteligencije u računalnim igrama bila je u klasičnim igrama na ploči poput šaha te je obuhvaćala pretežito različite tehnike i metode pretraživanja prostora stanja kako bi odredila optimalan potez. Daljnjim istraživanjem i tehnološkim napretkom počele su se koristiti i druge metode poput neuronskih mreža, Bayesove tehnike te genetskih algoritama kako bi se poboljšalo igranje računalnih igara od strane računala.

Cilj evolucijskih algoritama najčešće je pronaći optimalno rješenje nekog problema. Primjenom istih na neuronske mreže nastoji se izgraditi neuronska mreža koja će rješavati problem na optimalan način odnosno na temelju ulaza izračunati optimalan izlaz koji će se koristiti u specifičnoj situaciji.

Neuroevolucija je nastala primjenom evolucijskih algoritama na neuronske mreže s ciljem razvoja neuronske mreže kroz koncepte evolucije. Kao jedna od tehnika strojnog učenja, neuroevolucija ima značajnu primjenu u računalnim igrama. Od nastanka neuroevolucije prije otprilike trideset godina, bilježi se rast popularnosti i razvoj neuroevolucije u različitim područjima računalnih igara. Koristi se u dizajniranju igrača u igrama na ploči poput šaha, kao i u video igrama s kompleksnim VR elementima.

Primjenom neuroevolucijskih metoda, moguće je ostvariti složena ponašanja NPC-a (engl. *Non-Player Character*) te njegovo poboljšanje kroz igru. NPC-ovi su likovi odnosno igrači računalne igre koji se ponašaju inteligentno na temelju vlastitih izračuna te oponašaju ljudsko igranje igre. U računalnim igrama rješavanje problema na optimalan način odnosi se na optimalno igranje igre koje kao posljedicu ima opstanak u računalnoj igri, tj. duže igranje s pogodnostima i ostvarivanjem ciljeva igre bez poraza. Neuronska mreža koja je evoluirala kroz proces neuroevolucije prima parametre koji opisuju stanje igre, te na temelju njih nastoji izračunati optimalan potez u igri.

Postavlja se pitanje na koji način računalo trenira i poboljšava svoje sposobnosti igranja računalne igre. Po uzoru na biološki proces evolucije, evolucijski algoritmi

nastoje pronaći rješenje na temelju procesa iz prirode poput križanja, mutacija i selekcije. U kontekstu neuroevolucije, navedeni procesi primjenjuju se na skupu neuronskih mreža koje se, kao jedinke populacije, nastoje izgraditi te evoluirati od jednostavnih neuronskih mreža do onih složenijih koje mogu rješavati složenije probleme.

Inteligentni igrač ili agent obuhvaća neuronsku mrežu koja je trenirana uz pomoć evolucijskog algoritma. Neuronska mreža trenira se vlastitim iskustvom. Neuronska mreža koja se trenira za specifičan problem rješavanja određene računalne igre iskustvo stječe igrajući igru više puta. Igranjem igre neuronska mreža uči dobivajući podatke o uspješnosti vlastitih performansi. Proces učenja sastoji se od mijenjanja parametara i strukture neuronske mreže s ciljem poboljšanja budućeg igranja.

Kao i kod evoluiranja organizama u prirodi, za uspješan razvoj jedinki i populacije potrebno je imati raznolik skup gena i veliku populaciju kako bi se jedinke što bolje evoluirale na temelju raznolikosti gena. Kroz koncepte neuroevolucije ne trenira se jedan igrač već cijela populacija jedinki koja igra računalnu igru. Iz generacije u generaciju opstaju jedinke čije su se performanse pokazale najuspješnijima. Na temelju njihovih karakteristika, odnosno gena, razvijaju se nove jedinke s boljim genima koji doprinose poboljšanju strategije igranja igre. Neuronska mreža najboljeg igrača, koji je uspio ostvariti složeno ponašanje i ciljeve igre, može se koristiti kao protivnik u igranju igara protiv računala.

Još jedan razlog primjene neuroevolucije u računalnim igrama je testiranje neuroevolucijskih algoritama. Računalne igre jedan su od najjednostavnijih testnih okruženja za ispitivanje svojstava algoritama jer nude jednostavnu vizualizaciju te simulaciju rezultata. Stoga mnogi znanstvenici i inženjeri koriste jednostavne računalne igre kao područje u kojem ispituju performanse neuroevolucije i evaluiraju genetske algoritme za optimizaciju neuronskih mreža koje u igrama nastoje pronaći dobre taktike igranja ili ponašanja inteligentnog igrača [7].

Iako se strategije igranja računalnih igara mogu ostvariti različitim metodama strojnog učenja, u određenim situacijama postoje prednosti korištenja neuroevolucije za razvoj i optimiziranje strategije igranja te neuroevolucija ostvaruje najefikasnije rezultate rješavanja problema. Neuroevolucijom NPC-ova računalnih igara smanjuje se predvidljivost jer se njihovo ponašanje temelji na nedeterminističnosti. To doprinosi povećanju zanimljivosti i dinamičnosti računalne igre. Dodatno, takvo ponašanje doprinosi i realističnosti igre, pogotovo u situacijama kada računalna igra simulira stvarni svijet. Pogodnosti koju nude nedeterminističke tehnike, pa tako i neuroevolucija je prilagodba ponašanja NPC-a na postupke koje ljudski igrač obavlja u računalnoj igri. Primjer takvih igara su složene videoigre koje imitiraju događaje iz stvarnog svijeta u

kojima NPC-ovi imitiraju ljudske radnje.

Prednosti korištenja neuroevolucije je skalabilnost. Pokazuje se da neuroevolucija bolje obrađuje velike prostore stanja ili akcija u odnosu na ostale tehnike podržanog učenja, posebice u situacijama kada se koristi za direktan odabir akcije igrača. Zahvaljujući raznolikosti metoda koje obuhvaća, neuroevolucija može postići nekoliko različitih oblika rješenja kojim se mogu ostvariti smislene različite strategije, modeli i sadržaj igre [7].

Neuroevolucija u računalnim igrama može se primjenjivati na različite načine. Pretežito se koristi za učenje igranja računalne igre ili kontroliranja ponašanja NPC-a u igri. U tim situacijama neuronska mreža ili ocjenjuje vrijednost stanja ili akcije kako bi drugi algoritam donio odluku o odabranoj akciji ili direktno bira koju akciju poduzeti na temelju stanja igre.

Proceduralno generiranje sadržaja (engl. *Procedural Content Generation*) jedna je od aktualnih primjena neuroevolucije u računalnim igrama gdje se neuronska mreža koristi kao reprezentacija sadržaja. Uz to, neuroevolucija se može koristiti s ciljem predviđanja iskustva ili prioriteta igrača koji igraju igru.

Unatoč tome postoje nedostatci primjene neuroevolucije kod igara. Jedan od temeljnih nedostataka je činjenica da je teško predvidjeti ponašanje koje će biti naučeno kroz proces evolucije, a ponekad i pronaći razlog naučenog ponašanja [7].

2.1. Faktori razvoja strategije igre

Prilikom razvoja strategije igranja igre potrebno je u obzir uzeti nekoliko ključnih faktora koji imaju velik utjecaj na rezultate treniranja [4].

2.1.1. Funkcija dobrote

Funkcija dobrote (engl. *fitness function*) je funkcija koja pridjeljuje jedinkama populacije brojčanu vrijednost na temelju njihove uspješnosti. Pridjeljena vrijednost naziva se *dobrota* (engl. *fitness*). Dobrota pojedine jedinke može se temeljiti na sposobnosti igranja igre u odnosu na fiksnog protivnika ili u odnosu na populaciju koja se razvija. Potonji slučaj primjer je koevolucije koja je detaljnije objašnjena u trećem poglavlju. Koristeći principe koevolucije pri izračunu dobrote jedinki nije potrebno implementirati ljudske strategije [4] koje imaju ograničen nivo složenosti.

2.1.2. Faktor raznolikosti

Uspjeh razvoja neuronskih mreža temelji se na modificiranju genoma neuronske mreže te je stoga vrlo bitno osigurati mogućnost mijenjanja i modificiranja neuronske mreže. To se može postići definiranjem dovoljno velike populacije ili pak povećanjem vjerojatnosti mutacija. Velika populacija kao i mutacije osigurat će raznolikost pretraživanja prostora strategija te veće mogućnosti kombiniranja gena s ciljem optimiranja ponašanja neuronske mreže.

2.1.3. Podaci neuronske mreže

Velik utjecaj na rezultate i uspješnosti igranja igre od strane neuronske mreže imaju ulazni i izlazni podaci. Neuronska mreža na temelju ulaznih podataka računa izlaz na temelju kojeg donosi odluku o akciji koja se poduzima kao sljedeći korak u igri. Nedovoljno podataka o trenutnom stanju, kao i loše definirane akcije na temelju izlaznih podataka ograničavaju rad neuronske mreže i ne doprinose uspjehu jedinke pri igranju igre.

2.1.4. Parametri genetskog algoritma

Vjerojatnosti poput mutacija gena, dodavanja gena i sl. kao i parametri poput inicijalne topologije, veličine neuronske mreže i veličine populacije utječu na razvoj neuronskih mreža u procesu neuroevolucije. Negativan utjecaj na uspjeh razvoja može biti velika vjerojatnost mutacije gena, jer geni neće opstati dovoljno dugo da se optimiraju i pokažu korisnima. S druge strane velika populacija na temelju raznolikosti gena može doprinijeti uspjehu razvoja neuronskih mreža.

3. Genetski algoritam NEAT

Pri prvotnom pristupu neuroevoluciji, fiksna topologija neuronske mreže koja uobičajeno obuhvaća jedan sloj skrivenih neurona odabire se na samom početku učenja neuronske mreže. Cilj primjene genetskog algoritma na neuronsku mrežu u tom slučaju je pronaći optimalne težine veza između neurona pri kojima će neuronska mreža najbolje djelovati. Djelujući na fiksnoj topologiji, genetski algoritam kroz postupke križanja i mutacije djeluje isključivo na težine neuronskih veza.

Uz težine veza između neurona neuronske mreže, velik utjecaj na rad neuronske mreže ima upravo i njena struktura.

Postavlja se pitanje smisla izmjene strukture neuronske mreže s obzirom na činjenicu da, prema univerzalnom teoremu aproksimacije [5], bilo koja kontinuirana funkcija može biti aproksimirana neuronskom mrežom s jednim slojem skrivenih neurona. Stoga je prirodno zapitati se doprinosi li dodatno razvoj strukture neuronske mreže i na koji način.

Pokazuje se da izmjena strukture neuronske mreže uz izmjenu težina neuronskih veza smanjuje vrijeme potrebno za analizu i izgradnju neuronske mreže, odnosno broja skrivenih neurona, vezane uz specifičan problem. Dodatno, paralelnim izmjenama strukture neuronske mreže kao i težina veza između čvorova, tj. dodavanjem i oduzimanjem neuronskih veza i neurona, moguće je poboljšati performanse neuroevolucije [9].

Glavni izazovi koji se javljaju kod neuroevolucije pri razvoju topologija neuronske mreže su:

1. Odabir genetske reprezentacije koja će omogućiti križanje različitih topologija;
2. Očuvanje nove topologije, kojoj je potrebno nekoliko generacija da bi postigla optimalne performanse, unutar populacije;
3. Minimizacija topologije tijekom evolucije bez uvođenja funkcije koja mjeri složenost mreže.

Primjer neuroevolucijskog algoritma kojim se uz težine neuronskih veza razvija i topologija mreže je algoritam NEAT (engl. *NeuroEvolution of Augmenting Topologies*) koji pripada skupini TWEANN (engl. *Topology and Weight Evolving Artificial Neural Networks*) metoda. Algoritam NEAT osmišljen je 2002. godine od strane Kenneth Stanleyja i Risto Miikkulainena.

Algoritam NEAT prethodno definirane probleme rješava implementacijom metode koja provjerava homogenost gena, specijacije jedinki populacije s ciljem očuvanja novostvorene topologije neuronske mreže te minimiziranjem topologije prve generacije populacije.

3.1. Kodiranje genoma

Osnovno svojstvo genetskih algoritama je razvoj jedinki unutar populacije, sve dok se ne postigne traženo ponašanje ili zadovoljavajuća svojstva populacije. Svaka jedinka unutar populacije kodirana je nizom bitova koji označavaju njezina svojstva odnosno predstavljaju njezin genom. Genom sadrži informacije o pojedinoj jedinki koje utječu kako na samo ponašanje jedinke tako i na njezin uspjeh, tj. ostvarenu vrijednost funkcije dobrote.

U neuroevoluciji populacija sadrži skup neuronskih mreža. Svaka mreža jedinka je u populaciji koja se optimira genetskim algoritmom. Karakteristika po kojoj se jedinke populacije međusobno razlikuju je genom jedinke. On u neuroevoluciji obuhvaća dijelove pojedine mreže, njezine neurone i neuronske veze.

Kodiranje genoma je proces pretvorbe neuronske mreže u njenu genetsku reprezentaciju. Postoje različiti načini kodiranja genoma. Prilikom odabira genetske enkripcije za jedinke populacije koje se razvijaju TWEANN algoritmima, potrebno je uzeti u obzir činjenicu da genom mora sadržavati informacije i o strukturi mreže i o neuronskim vezama između neurona jer sam genetski algoritam razvija odnosno mijenja, mutira, dodaje i oduzima oboje, čvorove i veze neuronske mreže.

Genom se može kodirati takozvanim direktnim kodiranjem (engl. *direct encoding*). Direktno kodiranje obuhvaća definiranje svakog pojedinog neurona te svake neuronske veze koje će se pojaviti u neuronskoj mreži odnosno jedinki populacije. Kod takvog kodiranja, svi parametri neuronske mreže mogu se dobiti iz same genetske reprezentacije jedinke, tj. njenog genoma [2].

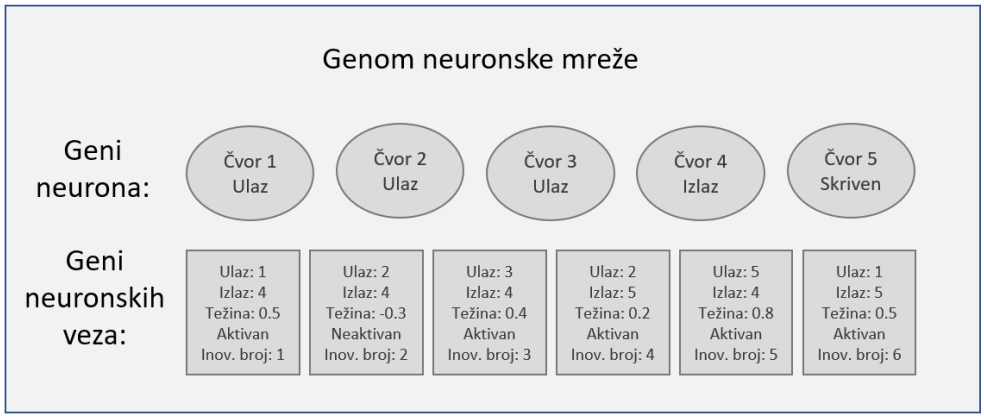
Primjer takvog genetskog kodiranja je binarna enkripcija genoma jedinke unutar matrice neuronskih veza, gdje bitovi '1' unutar matrice predstavljaju konekciju između neurona numeriranih rednim brojevima retka i stupca matrice. Nadalje direktno kodi-

ranje moguće je ostvariti pomoću grafova u koje se pohranjuju specifične informacije o neuronima i njihovim međusobnim vezama.

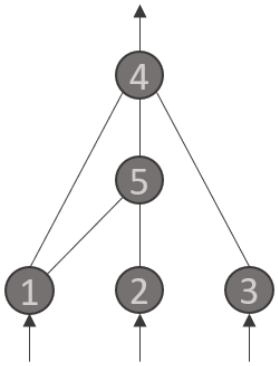
Drugačiji pristup genetskom kodiranju je indirektno kodiranje (engl. *indirect encoding*). Takva vrsta kodiranja specifična je za reprezentaciju gena neuronske mreže koja se koristi za rješavanje složenijih problema koji zahtijevaju visoku razinu složenosti neuronske mreže koja nadilazi onu s direktnim kodiranjem [2]. Karakteristika takvog kodiranja je da stvoreni genom ne specificira pojedine neuronske veze i neurone, iako se oni mogu saznati na temelju genoma [9]. Iako složeniji, takvim načinom moguće je ostvariti regularnost, tj. svojstvo kompresije informacija o genima u strukturi genoma koja nadalje doprinosi ponovnom iskorištavanju istih u svrhu opisivanja različitih dijelova jedinke populacije [1].

Algoritam NEAT koristi direktno kodiranje genoma. Dizajniran tako, NEAT omogućava korespondentnim genima jednostavnu kombinaciju u procesu križanja jedinki [8]. Geni populacije obuhvaćaju takozvane gene neuronske veze i gene neurona, dok se genom jedinki populacije sastoji od liste gena neuronske veze. Svaki gen neuronske veze sadrži informaciju o genima čvorova, tj. neurona koje povezuje. Osim toga, gen veze sadrži informacije o težini veze, aktiviranosti veze i jedinstvenog inovacijskog broja veze. Kako bi se poboljšala kompleksnost neuronske mreže u procesu optimiranja, duljina genoma nije ograničena te je omogućen proizvoljan broj skrivenih čvorova neuronske mreže.

Slika 3.1 prikazuje primjer genoma neuronske mreže koji se sastoji od čvorova i veza između njih. Slika 3.2 prikazuje neuronsku mrežu koja odgovara genomu sa slike 3.1.



Slika 3.1: Genom neuronske mreže



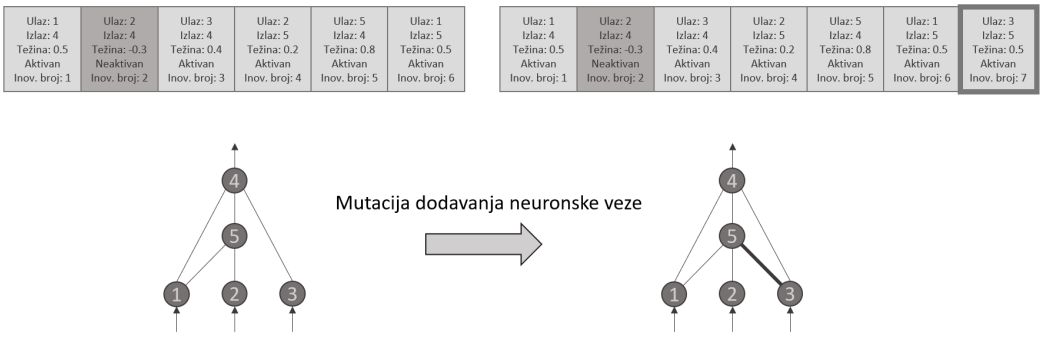
Slika 3.2: Neuronska mreža - fenotip genoma

3.2. Mutacije

Mutacije u algoritmu NEAT događaju se na neuronskim vezama i strukturi mreže. Mutacije na postojećim neuronskim vezama obuhvaćaju promjenu težine veze. Mutacije koje utječu na strukturu mreže, takozvane strukturalne mutacije moguće je podijeliti na dvije grupe mutacije: dodavanje veze između postojećih neurona i dodavanje novog neurona u neuronsku mrežu. Karakteristika obje strukturalne mutacije je povećanje veličine neuronske veze jer svaka mutacija dodaje nove gene.

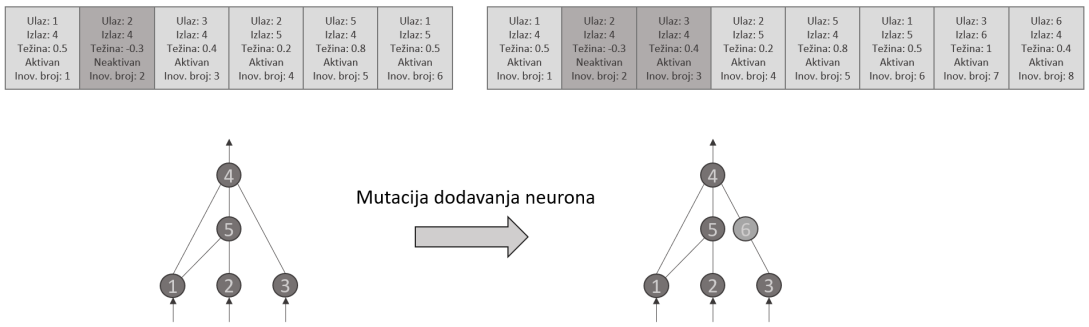
Mutacija dodavanja veze je metoda koja dodaje novi gen neuronske veze koji povezuje dva postojeća neurona. Težina novostvorene neuronske veze odabrana je slučajno.

Na slikama 3.3 i 3.4 prikazane su mutacije strukture neuronske mreže.



Slika 3.3: Mutacija dodavanja neuronske veze

Mutacija dodavanja novog neurona ponešto je složenija metoda. Novi neuron dodaje se na poziciju postojeće neuronske veze. Neuronska veza koju on zamjenjuje ostaje unutar genoma ali gubi funkcionalnost tj. prestaje biti aktivna. Njezin bit ak-



Slika 3.4: Mutacija dodavanja neurona

tivnosti postavlja se u nulu. U sam genom dodaju se još dvije neuronske veze koje povezuju novi neuron s neuronima koje je povezivala deaktivirana neuronska veza. Težina veze koja ulazi u novi neuron postavlja se na 1, dok se težina neuronske veze koja je prethodno bila aktivirana prenosi na izlaznu vezu novog neurona. Na taj način minimiziraju se početni efekti mutacije [8].

3.3. Križanje

Križanje jedinki populacije s različitim topologijama jedan je od tri temeljna problema neuroevolucije. Kod takvog križanja dolazi do problema natjecateljskih konvencija (engl. *Competing Conventions*) ili permutacijskog problema. Uzrok takvog problema je mogućnost kodiranja neuronske mreže na različite načine. Zbog toga rezultat križanja jedinki koje, iako različito kodirane, imaju istu funkcionalnost, može biti nova jedinka koja je oštećena odnosno koja je izgubila dio gena ili kod koje je došlo do duplikacije gena.

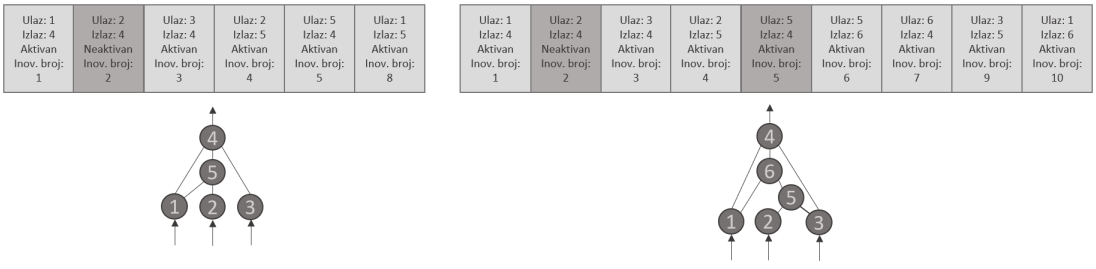
Dodatan problem predstavlja činjenica da rezultat križanja jedinki različitih topologija nije intuitivan, već je kod određivanja nasljednih gena potrebno znati koji dio genoma predstavlja određenu funkcionalnost.

NEAT navedenom problemu doskače na način da prati gene kroz proces evolucije od trenutka njihova nastanka. Permutacije gena u tom slučaju ne predstavljaju problem jer se istovjetnost gena može saznati na temelju podrijetla odnosno nastanka. Uz to,

NEAT ima mogućnost provjere zajedničkih svojstava različitih genoma, što omogućuje sprječavanje križanja topologija koje su potpuno neusklađene.

Križanje u algoritmu NEAT temeljeno je na principu homologije koje nalaže da se označavanjem gena na temelju njihovog nastanka (engl. *historical marking*) postiže identifikacija jednakosti gena. Takvo označavanje postiže se kroz inovacijski broj gena koji jedinstveno određuje njegovo vrijeme nastanka te ne zahtijeva analizu topologije fenotipa jedinke kako bi se ustvrdila jednakost gena [10].

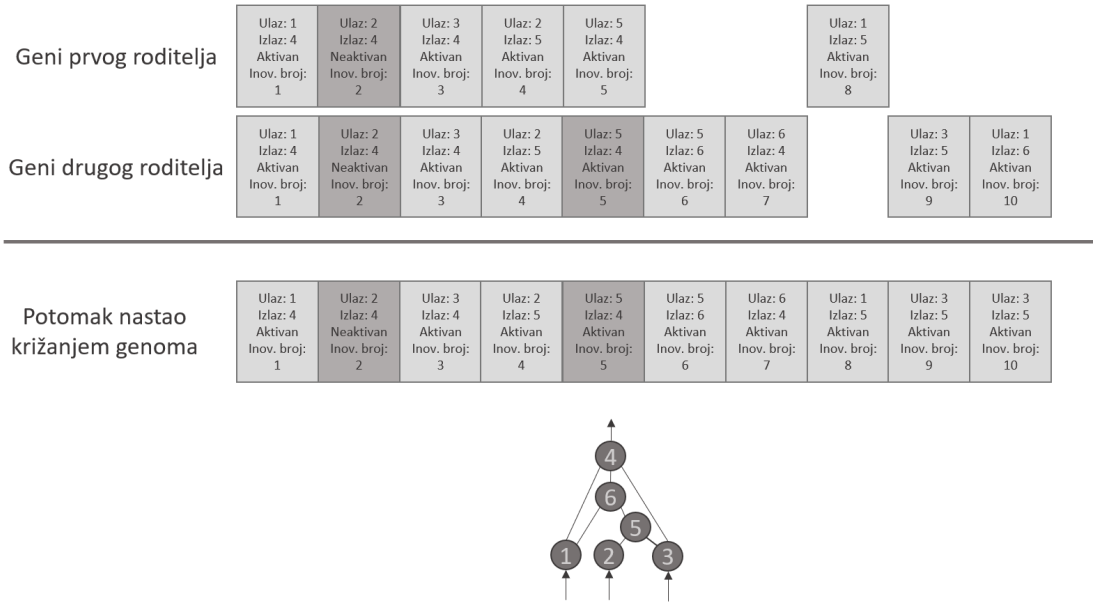
Na slikama 3.5 i 3.6 su prikazani geni neuronskih veza i topologije različitih neuronskih mreža s obzirom na strukturu i proces križanja tih mreža. Proces križanja pokazuje neuronsku mrežu nastalu kao rezultat križanja neuronskih mreža različitih topologija. Geni koji ne dijele zajedničko podrijetlo ni sa jednim genom druge neuronske mreže nazivaju se razdvojeni geni (engl. *disjoint genes*). Geni koji su se u jednom roditelju evolucijski pojavili nakon svih gena drugog roditelja nazivaju se viška geni (engl. *excess genes*). Geni s inovacijskim brojevima 6 i 7 na slici 3.5 su razdvojeni geni dok su geni s inovacijskim brojevima 9 i 10 na slici 3.5 takozvani viška geni. U danom primjeru potomak je naslijedio sve gene pod pretpostavkom o jednakosti vrijednosti funkcije dobrote roditeljskih neuronskih mreža. Svi razdvojeni geni i viška geni koje je potomak naslijedio naslijeđeni su od jednog slučajno odabranog roditelja. U slučaju nejednakosti vrijednosti dobrote, potomak nasljeđuje gene uspješnijeg roditelja.



Slika 3.5: Neuronske mreže različitih topologija

3.4. Očuvanje inovacije

Očuvanje inovacije bitan je faktor uspjeha napredovanja genetskog algoritma. U TWEANN metodama, pa tako i u algoritmu NEAT, inovacija se postiže strukturalnom mutacijom, tj. dodavanjem novih gena jedinki. S obzirom na to da se neuronska mreža



Slika 3.6: Proces i rezultat križanja neuronskih mreža različitih topologija

postepeno razvija i optimira počevši od minimalne strukture, u većini slučajeva je potrebno nekoliko generacija populacije kako bi se novouvedena topologija optimirala i počela doprinositi poboljšanju jedinki populacije. Problem koji se javlja kod očuvanja inovacije je smanjenje vrijednosti funkcije dobrote prilikom uvođenja nove strukture.

Posljedica pada vrijednosti funkcije dobrote jedinke s novim genom je smanjenje vjerojatnosti preživljavanja u sljedećim generacijama te je zbog toga potrebno očuvati novitete struktura unutar populacije kako bi se jedinke imale priliku razviti, optimirati i poboljšati svoju vrijednost dobrote, a potencijalno i ukupnu vrijednost dobrote populacije, s novom strukturom.

Navedeni problem riješen je po uzoru na prirodne procese, gdje različiti organizmi najčešće pripadaju različitim vrstama koje se evoluiranjem diferenciraju i stvaraju nove vrste.

Kako bi se inovacija strukture zaštitila, NEAT je baziran na principu zaštite inovacije koji nalaže specijaciju populacije, tj. izoliranje jedinki s novim svojstvima u vlastitu vrstu koja će se optimirati unutar nje same neovisno o starijim vrstama koje zbog veće vrijednosti funkcije dobrote potencijalno mogu ugroziti preživljavanje novih jedinki.

Prednosti korištenja specijacije su održavanje jedinki različitih topologija unutar populacije, što dozvoljava paralelni razvoj različitih struktura jedinki. Pokazano je da

optimiranje populacije bez specijacije može značajno smanjiti mogućnost dodavanja i razvoja novih topologija unutar populacije [10].

Korištenjem inovacijskih brojeva određuje se povijest gena, a samim time moguće je odrediti koliko gena dijele različiti genomi i tu informaciju iskoristiti u svrhu stvaranja vrsta unutar populacije. Razlika između dva genoma neronske mreže računa se kao linearna kombinacija viška gena (E), razdvojenih gena (D) te prosječne težine zajedničkih gena ($\overline{W}$) formulom 3.1. Koeficijenti c_1, c_2 i c_3 određuju utjecaj navedenih triju faktora, a faktor *N*, tj. broj gena većeg genoma koristi se s ciljem normaliziranja veličine genoma.

$$\delta = \frac{c_1 E}{N} + \frac{c_1 E}{N} + c_3 \cdot \overline{W} \tag{3.1}$$

Usporedbom genoma neuronske mreže s prethodno odabranim reprezentantom svake pojedine vrste određuje se pripadnost vrsti i obavlja proces specijacije. U slučaju kada je razlika genoma manja od δ_t, tj. praga kompatibilnosti, genom se smješta u tu vrstu. Ako je pak genom nekompatibilan sa svakom postojećom vrstom, kreira se nova vrsta.

Specijacija u algoritmu NEAT temelji se na eksplicitnom dijeljenju vrijednosti dobrote. Metoda eksplicitnog dijeljenja podrazumijeva podjelu dobiti vrijednosti funkcije dobrote između sličnih individualnih jedinki populacije. Sličnost između jedinki određena je funkcijom koja vraća vrijednost 0 u slučaju kada razlike jedinki prelaze neku graničnu vrijednost δ_t, a 1 kada je razlika genoma manja od granične vrijednosti, tj. kada se uspoređuje razlika genoma iz iste vrste. Funkcija *sh* koja mjeri sličnost jedinki temelji se na broju nehomolognih neurona i neuronskih veza te određuje koliko se isti razlikuju u odnosu na zajedničko podrijetlo. Konačna vrijednost funkcije dobrote računa se u odnosu na različitost genoma svih jedinki populacije formulom 3.2 što omogućuje opstanak različitih vrsta.

$$f_i' = \frac{f_i}{\sum_{j=1}^{n} sh(\delta(i,j))} \tag{3.2}$$

Zanimljiva činjenica je da implementacija dijeljenja dobrote unutar sličnih vrsta kod algoritma NEAT nije nova ideja ili bolje implementirana u odnosu na prethodne verzije algoritma NEAT, već da se korištenjem metode dijeljenja vrijednosti dobrote po prvi puta u TWEANN metodama nastojalo postići očuvanje topoloških inovacija. Ono što specificira algoritam NEAT je činjenica da omogućuje metriku za razdvajanje topologija kakva nije bila dostupna prije te koja omogućuje razvoj populacije u skladu s principom zaštite inovacije [10].

3.5. Minimizacija topologije

Prije nego što je algoritam NEAT postojao, započinjanje procesa evolucije neuronskih mreža s uniformnom populacijom minimalne strukture nije bilo moguće jer nije postojala specijacija vrsti. Inicijalna populacija sastojala se od topološki različitih struktura jedinki iz koje bi opstale one bolje. Započinjanje algoritma sa slučajno odabranim topologijama jedinki nije dovodilo do optimalnog rješenja u smislu minimalnosti mreže jer su jedinke inicijalne populacije već sadržavale mnogo nepotrebnih gena. Kontrola veličine i složenosti evoluiranih topologija neuronskih mreža kontrolirala se uvođenjem kazni unutar funkcije dobrote, tj. smanjivanjem vrijednosti funkcije dobrote za složenije jedinke. Takav pristup imao je negativne posljedice u performansama jedinki jer se definiranjem kazne uvodio rizik da će neuronska mreža razviti drugačije performanse u odnosu na one koje bi razvila bez uvođenja kazni [6].

Problemi s kojima se suočavalo doveli su do zaključka da se utjecaj veličine i složenosti neuronske mreže može izostaviti iz izračuna funkcije dobrote u slučaju da se populacija razvija postepeno od minimalne strukture. Upravo taj zaključak doveo je do principa topološke inovacije na kojem se temelji implementacija algoritma NEAT.

Minimizacijom topologije kroz cjelokupni proces evolucije NEAT nastoji minimizirati prostor pretraživanja optimalne mreže i ubrzati proces optimizacije, tj. učenja. Upravo tom minimizacijom omogućava započinjanje algoritma s jednostavnom minimalnom inicijalnom populacijom.

Algoritam NEAT definira minimalnu arhitekturu kao "uniformnu populaciju mreža s nula skrivenih čvorova" u kojoj su svi ulazni čvorovi direktno spojeni s izlaznima [3]. NEAT započinje razvoj populacije postavljajući inicijalnu topologiju jedinki koja obuhvaća jedan *bias* čvor na ulazu te potpunu povezanost ulaznih i izlaznih neurona. Takva struktura mreže definirana je kao minimalna struktura. Početne težine neuronskih veza odabrane su slučajno.

Algoritam NEAT postepeno dodaje nove strukture koje opstaju u slučajevima kad se pokaže da doprinose poboljšanju vrijednosti funkcije dobrote. Tako se smanjuje broj generacija potrebnih za pronalazak rješenja te se garantira da neuronska mreža neće postati nepotrebno složena [10]. Proces postepenog povećanja složenosti neuronske mreže naziva se kompleksifikacija (engl. *complexification*). Izuzetno bitan doprinos kompleksifikacije je kontinuirana koevolucija odnosno kontinuirana inovacija kroz natjecanje s drugim jedinkama. Ona omogućuje otkrivanje novih rješenja.

Inicijalno, populacija evoluira pretragom rješenja u prostoru s malo parametara jer inicijalne mreže sadrže minimalan broj gena. Dodavanjem novih gena kroz pro-

ces strukturalne mutacije, prostor pretraživanja raste, no s obzirom na to da su neki geni iz prethodnih topologija već optimirani da obavljaju određen funkcionalnosti, njihova adaptacija novom prostoru može biti jednostavnija te se manji broj parametara mora paralelno optimirati. Posljedica toga je omogućeno pretraživanje prostora velikog broja dimenzija.

Topološke inovacije koje se uvode u populaciju očuvane su u slučaju kada poboljšavaju performanse neuronske mreže. S obzirom na to da neuronska mreža strukturalno raste od minimalne inicijalne topologije, algoritam NEAT nastoji razviti topologije manje strukture.

Principi na kojma počiva algoritam NEAT izuzetno su bitni te su uzor i temelji za izgradnju različitih varijanti algoritama NEAT i metoda u genetskom programiranju te se mogu primjenjivati u razne svrhe i područja koje nadilaze neuronske mreže.

4. Programska knjižnica NEAT-Python

U programskom jeziku Python postoji knjižnica NEAT-Python koja sadrži implementaciju algoritma NEAT. Knjižnica se temelji isključivo na standardnoj knjižnici Pythona.

Trenutna implementacija algoritma NEAT nastoji evoluirati populaciju individualnih genoma koji se sastoje od dva seta gena – čvorova koji označavaju neurone neuronske mreže i veza koje te čvorove povezuju.

Prilikom korištenja algoritma NEAT u NEAT-Python knjižnici, korisnik navodi funkciju dobrote pomoću koje se računa uspješnost jedinki unutar populacije i koja time doprinosi razvoju i optimizaciji ponašanja jedinki. Dodatno navodi se i broj generacija populacije kroz koje će se genomi razvijati. Jedinke koje se pokažu najuspješnijima križat će se i mutirati, te će rezultat razmnožavanja njihovih genoma biti jedinke sljedeće generacije.

Algoritam se zaustavlja kada se dostigne zadani broj generacija ili kada neka od jedinki unutar populacije dostigne maksimalnu vrijednost funkcije dobrote definiranu u konfiguracijskoj datoteci.

4.1. Konfiguracijska datoteka

Specifične postavke algoritma NEAT predaju se kroz konfiguracijsku datoteku čija je struktura unaprijed definirana. Konfiguracijska datoteka sastoji se od četiri dijela, od kojih je nužno definirati barem jedan.

Odjeljak [NEAT] definira kriterij funkcije dobrote te graničnu vrijednost funkcije dobrote nakon čijeg dostizanja algoritam prestaje raditi. Kroz ovaj odjeljak moguće je definirati da uvjet zaustavljanja bude isključivo dostizanje maksimalnog broja generacija neovisno o vrijednosti funkcije dobrote. Dodatno definira se veličina populacije i uvjet resetiranja populacije u slučaju izumiranja svih jedinki.

Odjeljak [DefaultStagnation] definira kriterije odbacivanja jedinki iz populacije u slučaju stagnacije odnosno ne pokazivanja napretka kroz generacije te kriterij izračuna vrijednosti funkcije dobrote pojedine vrste.

Odjeljak [DefaultReproduction] definira broj individualnih genoma koji se ne mijenjaju unutar generacije, minimalni broj genoma nakon križanja, te dozvoljen udio jedinki za razmnožavanje. Svi parametri u ovom odjeljku odnose se na pojedine vrste unutar populacije.

Odjeljak [DefaultGenome] sadrži najviše parametara koji se mogu mijenjati unutar algoritma. Parametri tog odjeljka odnose se na strukturu i rad genoma. Kroz njih moguće je postaviti parametre i vrijednosti aktivacijske funkcije, pristranosti, granične vrijednosti za izračun sličnosti jedinki, vjerojatnosti dodavanja i brisanja čvorova i njihovih veza, inicijalnu strukturu jedinki prve generacije koja obuhvaća broj ulaznih, skrivenih i izlaznih čvorova te način povezanosti istih, vjerojatnosti mutacija, maksimalnu vrijednost težina neuronskih veza te brojne druge parametre.

Knjižnica NEAT-Python omogućuje definiranje vlastitih funkcija aktivacije koje se mogu konfigurirati proizvoljno i neovisno o konfiguracijskoj datoteci.

Statistički podaci o uspješnosti generacije odnosno genoma koji joj pripadaju dobivaju se kroz klasu `StatisticsReporter()`.

5. Programska knjižnica Pygame

Pygame je multimedijska knjižnica programskog jezika Python namijenjena izradi računalnih igara i multimedijskih aplikacija. Knjižnica Pygame nastala je kao nadogradnja SDL (engl. *Simple DirectMedia Layer*) knjižnice.

Programsku knjižnicu Pygame karakterizira mogućnost korištenja više jezgri, optimizacija koda korištenjem C i asemblerskog koda za temeljne funkcije, portabilnost prema operacijskim sustavima te jednostavnost korištenja.

Kako bi se knjižnica mogla koristiti, potrebno ju je uključiti u program naredbom `import pygame`. Prije samog korištenja metoda iz knjižnice potrebno je inicijalizirati uključene module knjižnice pygame naredbom `pygame.init()`. Uobičajeno je da se glavna petlja obavlja bezuvjetno te da se unutar nje analiziraju svi događaji nastali od strane korisnika za vrijeme igranja igre. Listu događaja moguće je dobiti naredbom `pygame.event.get()`. Glavna petlja prestaje s radom kada korisnik inicira događaj QUIT, te je u tom slučaju potrebno obraditi prekid rada igre. Kako bi aplikacija uspješno završila s radom potrebno je pozvati funkciju `pygame.quit()`.

5.1. Grafičko sučelje

Prozor u kojem se prikazuje računalna igra moguće je postaviti koristeći naredbu `pygame.display.set_mode()` s argumentom veličine prozora. Na nastalom `Surface` objektom iscrtavaju se grafički elementi. Funkcija `pygame.draw()` omogućava crtanje jednostavnih oblika (poligona, pravokutnika, kruga i elipse) na površinu. Nakon crtanja oblika na površinu pomoću navedene funkcije potrebno je ažurirati objekt `Surface` nad kojim se napravila promjena pozivom metode `pygame.display.update()`.

Kako bi se na površini prikazali tekstualni elementi potrebno je napraviti objekt `Font`, s tim fontom renderirati odabrani tekst u novi objekt `Surface` koristeći metodu `render()` te postaviti novostvoreni objekt s tekstom na proizvoljnu površinu koristeći metodu `blit()`.

6. Praktičan rad

U sklopu ovog rada, napravljene su dvije računalne igrice čije se igranje nastoji optimirati pomoću neuroevolucije koristeći NEAT algoritam. Cilj primjene NEAT algoritma na igrice Snake i Flappy bird je pronaći optimalne parametre za razvoj populacije te usporediti ovisnost parametara s performansama genoma koji se treniraju.

6.1. Računalna igra *Snake*

Jedna od najpoznatijih računalnih igara je igra Snake. Cilj igre je, pomičući zmiju u različitim smjerovima, skupljati nagrade koje se pojavljuju u igri. Svako uspješno prikupljanje nagrade doprinosi povećanju uspjeha i broja bodova stečenih u igri (engl. *score*) te povećava duljinu zmije za 1. Cilj primjene neuroevolucije na računalnoj igri Snake je izgraditi neuronsku mrežu koja će samostalno i uspješno igrati računalnu igru. U ovom slučaju neuronska mreža trenira se s ciljem donošenja odluke koju akciju poduzeti s obzirom na trenutno stanje igre.

Kroz računalnu igru Snake, testirali su se parametri genetskog algoritma na utjecaj performansi neuronskih mreža pri igranju igre. Svi navedeni parametri mijenjani su u konfiguracijskoj datoteci `config-snake.txt`.

6.1.1. Programsko ostvarenje

Računalna igra napravljena u sklopu završnog rada napisana je u programskom jeziku Python. Programske knjižnice koje su se koristile pri izradi igre su knjižnica NEAT-Python i knjižnica Pygame.

Temeljna funkcija programa je `eval_genoms()` koja se poziva prilikom evoluiranja svake generacije populacije. Ta funkcija izračunava vrijednost dobrote genoma i izlazne podatke neuronskih mreža na temelju kojih optimira jedinke za igranje igre.

Funkcija `eval_genoms()` u sebi sadrži tri polja:

– Polje neuronskih mreža

– Polje igara

– Polje genoma

Igra obuhvaća ploču, zmiju i nagradu koja se slučajno postavlja na slobodno polje ploče. Svaka igra ima pripadajuću neuronsku mrežu koja se optimira genetskim algoritmom te na temelju koje se izračunava ponašanje zmije u igri te genom kojem se pridjeljuje vrijednost dobrote na temelju uspjeha igranja igre.

Sve dok neuronska mreža igra računalnu igru, što se ispituje funkcijom `play()` klase `Game`, igra ostaje u polju igara te njezina pripadajuća neuronska mreža ostaje kao jedinka populacije koja se nastoji optimirati. Kada igra prestane, tj. kada se zmija u igri sudari s preprekom ili iskoristi maksimalan broj koraka između prikupljanja nagrada, igra prestaje i izbacuje se iz polja igara, kao i pripadajuća neuronska mreža iz populacije.

Sve jedinke populacije testiraju se na fiksnoj veličini ploče čije su dimenzije 10 x 10 polja.

6.1.2. Parametri i treniranje

Postavke genetskog algoritma kojim se trenirala računalna igra definirane su u konfiguracijskoj datoteci `config-snake.txt`. Kao kriterij preživljavanja jedinki iz generacije u generaciju razmatra se maksimalna vrijednost funkcije dobrote jedinki, te stoga opstaju one jedinke s najvećom dobrotom. U konfiguracijskoj datoteci postavljena je gornja vrijednost dobrote na 100, te algoritam prestaje s radom u slučaju dostizanja iste. Kao aktivacijska funkcija čvorova neuronske mreže, odabrana je funkcija tangens hiperbolni *tanh*. Koeficijenti koji se koriste u izrazu za računanje udaljenosti genoma dviju neuronskih mreža, kao i parametri za pristranost neuronske mreže, također su postavljeni na fiksne vrijednosti:

$$koeficijent\_kompatibilnosti\_razlicitih\_gena = 1$$

$$koeficijent\_kompatibilnosti\_tezina\_homolognih\_gena = 0.5$$

$$standardna\_devijacija\_vrijednosti\_mutacije\_cvora\_pristranost = 0.5$$

$$vjerojatnosti\_mutacije\_cvora\_pristranost = 0.7$$

$$vjerojatnost\_zamjene\_cvora\_pristranost = 0.1$$

Neuronske mreže koje se treniraju rade na principu ulančavanja unaprijed te nema povratnih veza između čvorova. Inicijalna struktura neuronskih mreža prve generacije postavljena je na potpunu povezanost ulaznih čvorova s izlaznima.

Neuronske mreže sastoje se od šest ulaznih čvorova i četiri izlazna neurona. Skrivene neurone generira evolucijski algoritam u procesu evoluiranja neuronskih mreža. Ulazni podaci neuronske mreže su:

1. Broj koraka do prve prepreke s desne strane ploče
2. Broj koraka do prve prepreke s lijeve strane ploče
3. Broj koraka do prve prepreke s gornje strane ploče
4. Broj koraka do prve prepreke s donje strane ploče
5. Razlika udaljenosti položaja nagrade i glave zmije na x osi
6. Razlika udaljenosti položaja nagrade i glave zmije na y osi

gdje broj koraka do prepreke označava broj slobodnih polja u određenom smjeru koji se računa u odnosu na trenutni položaj glave zmije. Prepreke koje se pritom uzimaju u obzir je granica ploče na kojoj se igra te tijelo zmije.

Na temelju izlaznih podataka neuronske mreže funkcija `set_new_orientation()` iz klase `Snake`, koja kao argument prima polje izlaznih podataka, postavlja novi smjer kretanja zmije na ploči.

Funkcija dobrote

Vrijednost dobrote jedinki populacije računa se na temelju broja sakupljenih nagrada te na temelju približavanja objektu nagrade. Svaka sakupljena nagrada povećava vrijednost dobrote jedinke za 1. Ako se zmija približi nagradi, tj. ako je udaljenost između položaja nagrade i glave zmije manja od iste udaljenosti prethodnog koraka, vrijednost dobrote jedinke povećava se za 0.02, dok se u slučaju udaljavanja smanjuje za 0.01.

Dodatno ograničenje koje je postavljeno na broj koraka zmije između skupljanja dviju nagrada, uvedeno je s ciljem zaustavljanja igranja igre onih jedinki koje se kreću po istoj putanji ploče. Ograničenje je postavljeno na 50 koraka.

Simulacija igranja najbolje jedinke iz generacije

Nakon evaluiranja svih jedinki unutar generacije, uzima se jedinka koja je u toj generaciji poprimila najveću vrijednost dobrote te se prikazuje njezino ponašanje pri igranju igre u novom okruženju. U sklopu ovog rada vizualno se ne prikazuju jedinke prilikom treniranja već se pokreće nova igra u kojoj se ponašanje zmije izračunava na temelju neuronske mreže koja je bila najuspješnija prilikom treniranja.

6.1.3. Rezultati

Razvoj neuronske mreže koja će igrati računalnu igru Snake testirana je s obzirom na različite parametre genetskog algoritma. Statistički podaci prikupljeni su korištenjem klase `StaticticsReporter()` iz programske knjižnice NEAT-Python.

Utjecaj veličine populacije na razvoj jedinki populacije

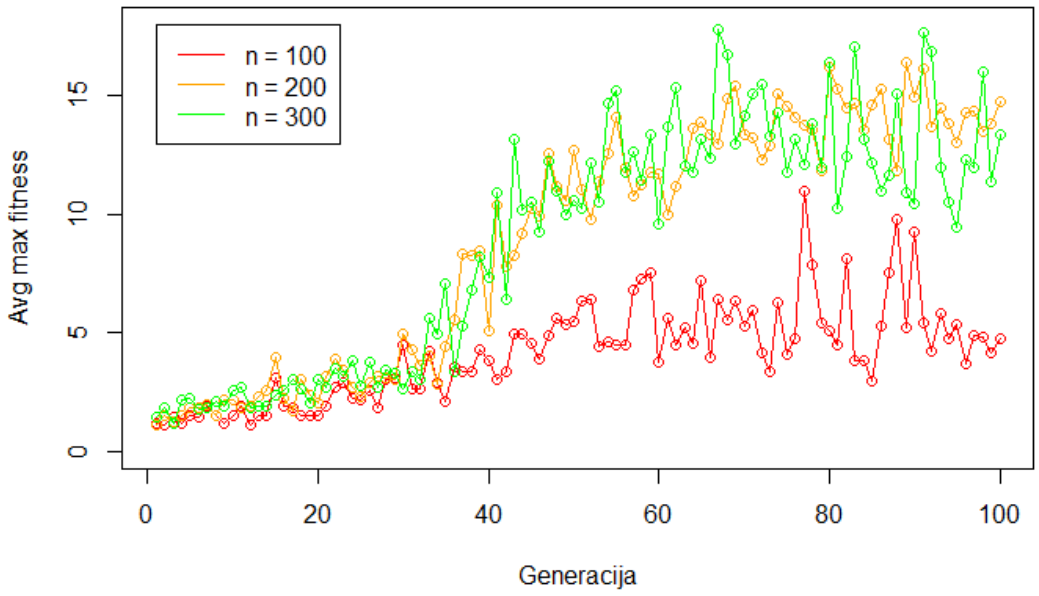
Utjecaj veličine populacije na uspješnost performansi neuronske mreže u igranju računalne igre ispitan je eksperimentom s tri različite veličine populacije od $n = 100$, $n = 200$ i $n = 300$ jedinki. Broj generacija u eksperimentu postavljen je na fiksnu vrijednost 100 te je algoritam prestajao s radom nakon evoluiranja 100 generacija populacije.

Evolucijski algoritam pokrenut je sveukupno devet puta, po tri puta za svaku veličinu populacije. Ostali parametri genetskog algoritma postavljeni su na fiksne vrijednosti. Pokazalo se da je veličina populacije bitno utječe na uspješnost razvoja neuronske mreže koja igra računalnu igru Snake. Prosječna maksimalna dobrota jedinki po generaciji prikazana je na slici 6.1 te je vidljivo da populacije s više jedinki pri inicijalizaciji postižu bolje rezultate. Iako je prosječna maksimalna vrijednost dobrote populacija od 200 i 300 jedinki slična, populacije s više jedinki su, pri pokretanju simulacije najboljeg genoma u generaciji, ostvarile bolje kretanje prema nagradi, za razliku od populacija s manjim brojem jedinki koje su svoje dohvaćanje nagradi temeljile na postupnom približavanju nagradi, pretražujući slijedno polja na putu do nje.

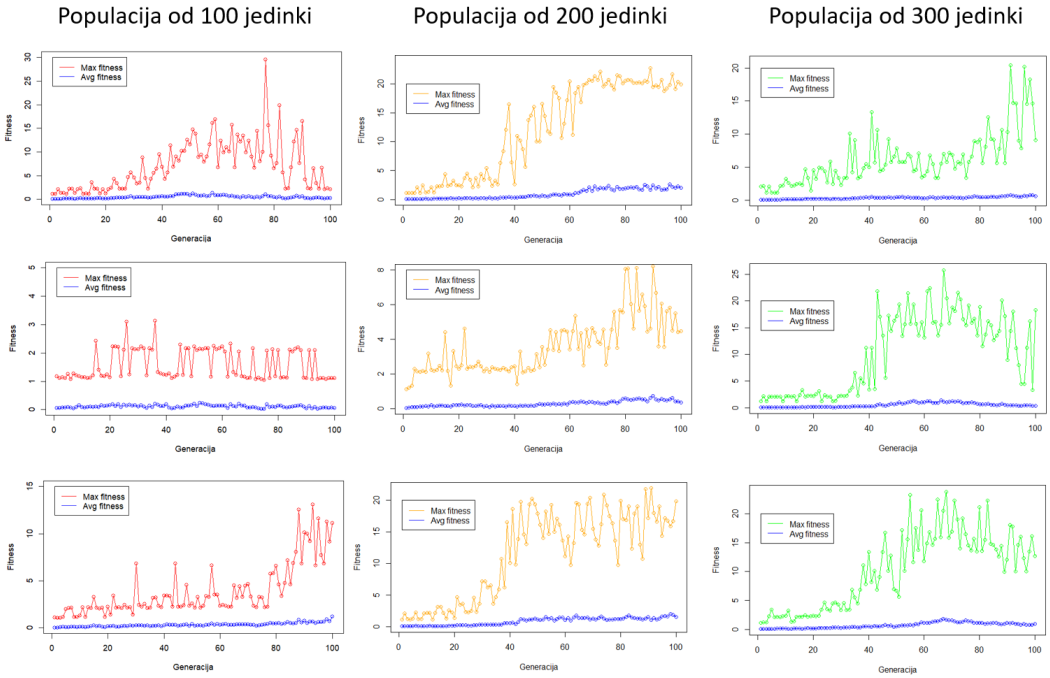
Populacije s 300 jedinki su pri svakoj evoluciji postigle vrijednost maksimalne dobrote veću od 20, populacije s 200 jedinki su u dva od tri puta ostvarile istu vrijednost dok je samo jedna populacija od 100 jedinki ostvarila vrijednost dobrote veću od 20. Rezultati evaluacije pojedinačnih populacija prikazani su slikom 6.2.

Na temelju analize strategije igranja igre u simulaciji igre s najboljim genomom iz generacije, pokazalo se da se jedna populacija od 100 jedinki uopće nije razvila, te se uspješnost jedinki te populacije u igranju igre temeljila na slučajnosti dohvaćanja nagrade. U toj populaciji došlo je do velike specijacije genoma po vrstama, te se zbog malog broja gena i genetske raznolikosti inicijalne populacije nije moglo optimirati jedinke populacije, što je dovelo do niske maksimalne vrijednosti funkcije dobrote. Frekvencije vrijednosti maksimalne dobrote prema broju jedinki populacije prikazane su na slici 6.3.

S druge strane kod evaluacije strategije igranja igre populacija s 200 jedinki, u većini slučajeva ustanovljeno je strateško ponašanje igrača koje se temeljilo na kretanju

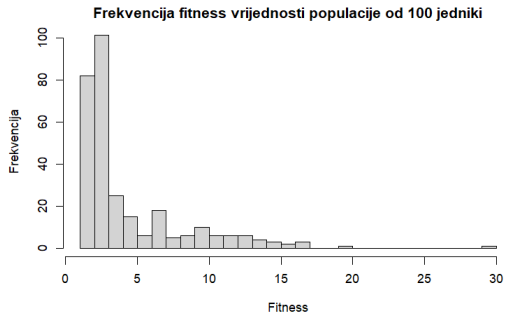


Slika 6.1: Graf prosječne maksimalne vrijednosti dobrote prema veličini populacije

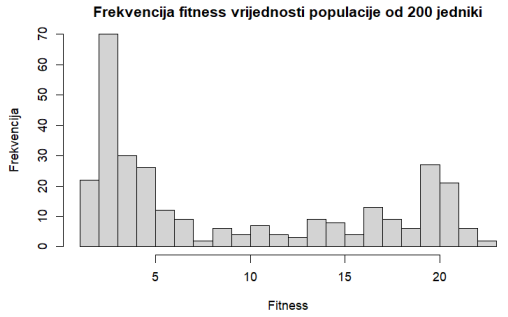


Slika 6.2: Grafovi maksimalne i prosječne vrijednosti dobrote po svakoj generaciji populacija s različitim brojem jedinki

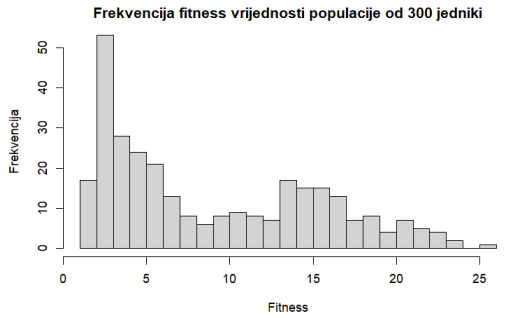
prema nagradi. Kretanje prema nagradi evoluiralo je tako da se agent kreće korak po korak po x osi pritom prolazeći kroz nekoliko polja na y osi ploče. Kod svih popula-



(a) Populacija od 100 jedinki



(b) Populacija od 200 jedinki



(c) Populacija od 300 jedinki

Slika 6.3: Frekvencije maksimalne vrijednosti funkcije dobrote

cija, pri početnim generacijama simulacije ustanovljeno je cirkularno kretanje po već posjećenim poljima ploče.

Pri testiranju utjecaja ostalih parametara populacije odabrana je fiksna veličina populacije od 300 jedinki.

Utjecaj vjerojatnosti dodavanja i brisanja neuronskih veza na razvoj jedinki populacije

Nadalje, ispitan je utjecaj vjerojatnosti dodavanja i brisanja gena neuronskih veza pri evoluiranju jedinki populacije. Populacije s istom vjerojatnosti pokrenute su po tri puta

svaka te je uzet prosjek maksimalno ostvarene vrijednosti dobrote po generaciji.

Testne vrijednosti dodavanja i oduzimanja neuronskih veza tijekom razvoja jedinki su $p = 0.2$, $p = 0.5$ i $p = 0.8$. Pri testiranju su vrijednosti ostalih parametara postavljene na fiksnu vrijednost:

$$vjerojatnost\_mutacije\_aktivnosti\_neuronske\_veze = 0.01$$

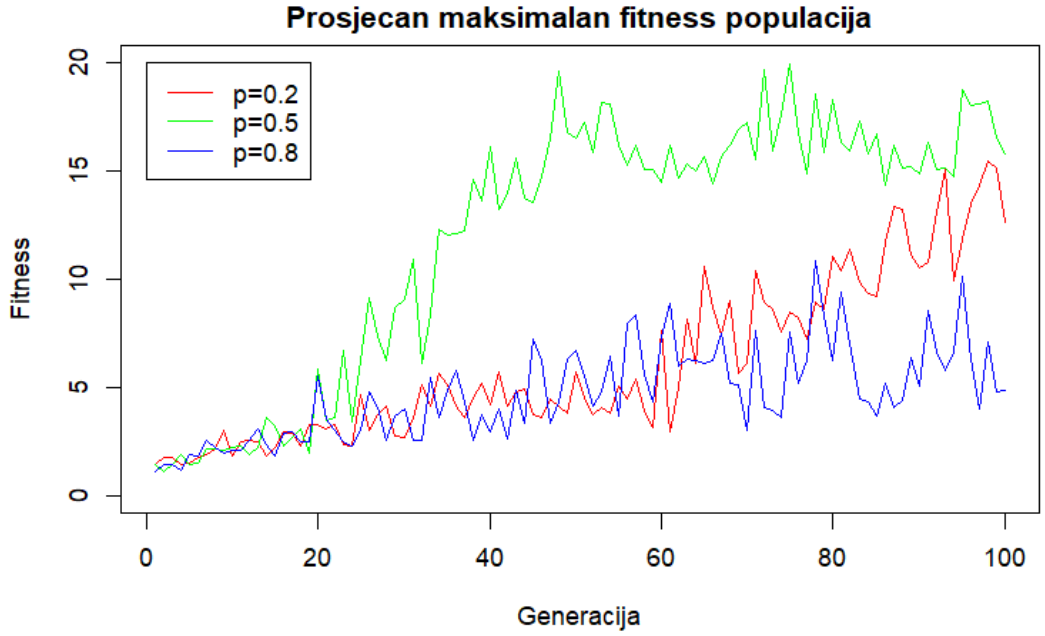
$$vjerojatnost\_dodavanja\_neurona = 0.3$$

$$vjerojatnost\_brisanja\_neurona = 0.3$$

$$standardna\_devijacija\_vjerojatnosti\_mutacije\_neuronske\_veze = 0.5$$

$$vjerojatnost\_mutacije\_neuronske\_veze = 0.3$$

$$vjerojatnost\_zamjene\_neuronske\_veze\_novom\_vezom = 0.1$$



Slika 6.4: Graf prosječne maksimalne vrijednosti dobrote populacija s različitom vjerojatnošću dodavanja i oduzimanja neuronskih veza

Rezultati testiranja prikazani su na slici 6.4. Pokazuje se da su najbolje performanse populacije ostvarene kada je vjerojatnost dodavanja i oduzimanja neuronskih veza postavljena na $p = 0.5$. Iako je razvoj i poboljšanje uspješnosti igranja igre bio vidljiv i pri nižoj vjerojatnosti dodavanja i oduzimanja gena, taj razvoj bio je mnogo sporiji. U slučaju postavljana visoke vjerojatnosti dodavanja i oduzimanja gena, populacija se ne može razviti jer su prečeste promjene genoma jedinki te nije moguće očuvati gene koji doprinose uspješnosti igranja igre.

Dodatno, pokazuje se da vjerojatnosti dodavanja i oduzimanja gena neuronskih veza utječu na specijaciju populacije te da se povećavanjem vjerojatnosti dodavanja i oduzimanja gena povećava broj vrsta stvorenih tijekom evoluiranja populacije. Podaci o specijaciji pojedinih populacija prikazani su na slici 6.5.

Broj vrsta populacija

	p=0.2	p=0.5	p=0.8
1.	1	6	11
2.	2	7	12
3.	4	7	16
	2.33	6.67	13

Slika 6.5: Broj vrsta populacija s obzirom na mijenjanje parametra vjerojatnosti dodavanja i oduzimanja neuronskih veza mreže

Karakteristično ponašanje u kojem se inteligentni igrač kreće postupno po x osi pritom prolazeći vertikalno kroz različita polja na y osi uočeno je pri simulaciji igranja igre s najboljim genomom iz generacije onih populacija čija je vjerojatnost dodavanja i brisanja gena postavljena na $p = 0.5$.

Utjecaj vjerojatnosti dodavanja i brisanja neurona na razvoj jedinki populacije

Napravljen je sličan eksperiment u kojem je testiran utjecaj vjerojatnosti dodavanja i brisanja gena neurona pri evoluiranju jedinki populacije. Kao i u prethodnom primjeru, populacije s istom vjerojatnosti pokrenute su po tri puta svaka te je uzet prosjek maksimalno ostvarene vrijednosti dobrote po generaciji.

Testne vrijednosti dodavanja i oduzimanja neuronskih veza tijekom razvoja jedinki su $p = 0.2$, $p = 0.5$ i $p = 0.8$. Pri testiranju su vrijednosti ostalih parametara postavljene na fiksnu vrijednost:

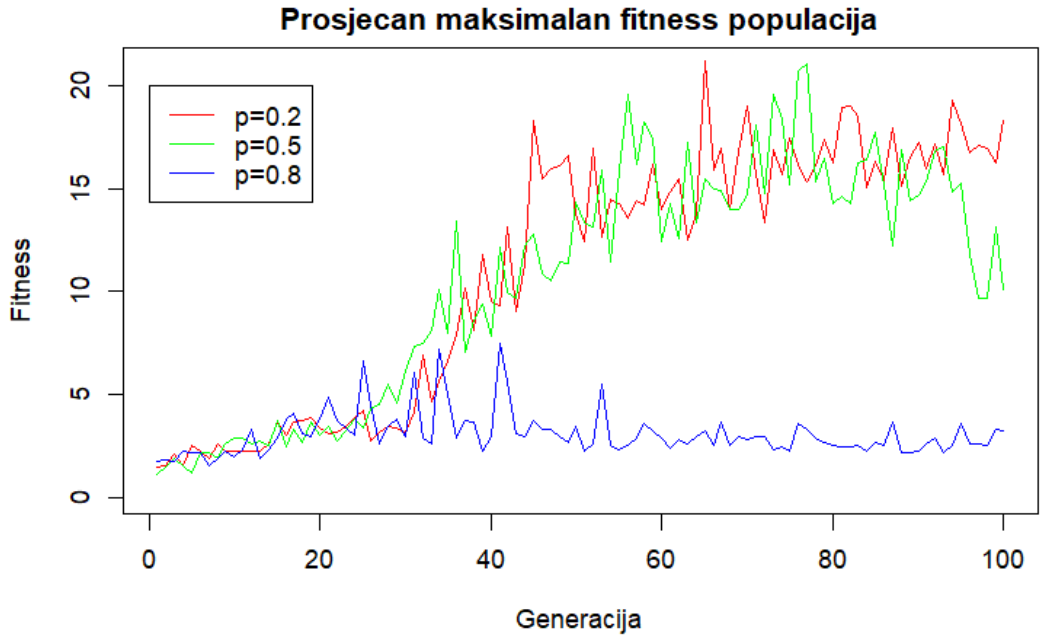
$$vjerojatnost\_mutacije\_aktivnosti\_neuronske\_veze = 0.01$$

$$vjerojatnost\_dodavanja\_neuronske\_veze = 0.5$$

$$vjerojatnost\_brisanja\_neuronske\_veze = 0.5$$

$$standardna\_devijacija\_vjerojatnosti\_mutacije\_neuronske\_veze = 0.5$$

$$vjerojatnost\_mutacije\_neuronske\_veze = 0.3$$



Slika 6.6: Graf prosječne maksimalne vrijednosti dobrote populacija s različitom vjerojatnošću dodavanja i oduzimanja neurona

$$vjerojatnost\_zamjene\_neuronske\_veze\_novom\_vezom = 0.1$$

Rezultati su prikazani na slici 6.6. Pokazuje se da su najbolje performanse populacije ostvarene kada je vjerojatnost dodavanja i oduzimanja neuronskih veza manja. Kao i kod dodavanja i brisanja neuronskih veza, u slučaju postavljana visoke vjerojatnosti dodavanja i oduzimanja neurona, populacija se ne može razviti jer su prečeste promjene genoma jedinki.

Sličan trend s obzirom na specijaciju vrsta uočen je kao i pri mijenjanju parametra vjerojatnosti dodavanja i brisanja neuronskih veza te je prikazan tablicom 6.7.

Utjecaj vjerojatnosti mutacije težina neuronskih veza na razvoj jedinki populacije

Napravljen je eksperiment u kojem je testiran utjecaj vjerojatnosti dodavanja i brisanja gena neuronskih veza pri evoluiranju jedinki populacije. Populacije s istom vjerojatnosti mutacije pokrenute su po tri puta svaka te je uzet prosjek maksimalno ostvarene vrijednosti dobrote po generaciji.

Testne vrijednosti mutacije težina neuronskih veza tijekom razvoja jedinki su $p = 0.2$, $p = 0.5$ i $p = 0.8$. Pri testiranju su vrijednosti ostalih parametara postavljene na

Broj vrsta populacija

	p=0.2	p=0.5	p=0.8
1.	3	5	7
2.	4	7	11
3.	7	8	12
	4.67	6.67	10

Slika 6.7: Broj vrsta populacija s obzirom na mijenjanje parametra vjerojatnosti dodavanja i oduzimanja neurona mreže

fiksnu vrijednost:

$$vjerojatnost\_mutacije\_aktivnosti\_neuronske\_veze = 0.01$$

$$vjerojatnost\_dodavanja\_neurona = 0.2$$

$$vjerojatnost\_brisanja\_neurona = 0.2$$

$$vjerojatnost\_dodavanja\_neuronske\_veze = 0.5$$

$$vjerojatnost\_brisanja\_neuronske\_veze = 0.5$$

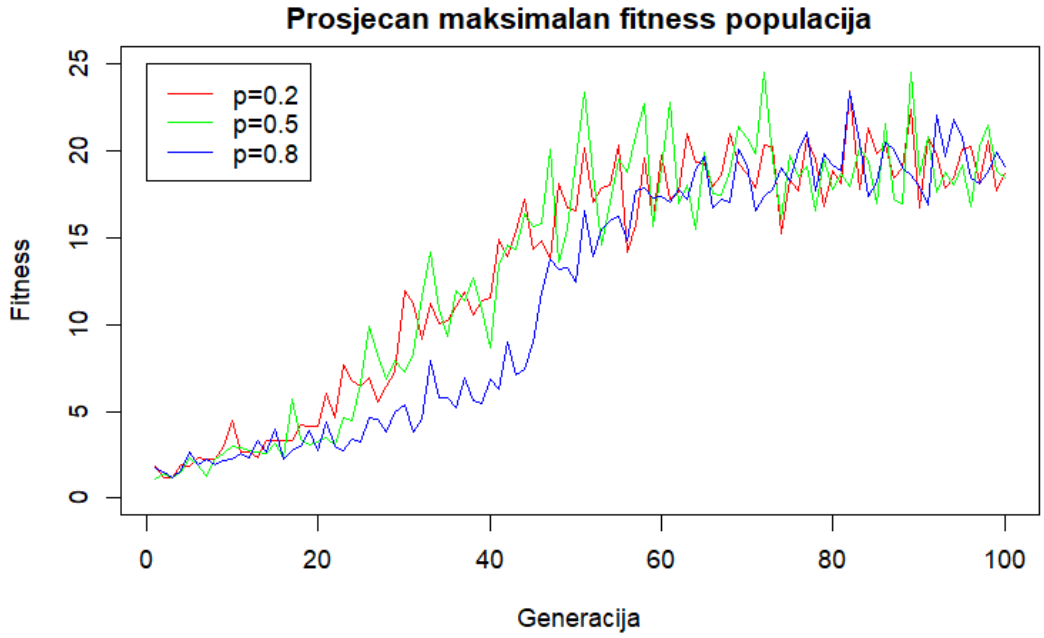
$$standardna\_devijacija\_vjerojatnosti\_mutacije\_neuronske\_veze = 1$$

$$vjerojatnost\_zamjene\_neuronske\_veze\_novom\_vezom = 0.05$$

Vjerojatnosti dodavanja i brisanja gena neuronske mreže postavljene su na one vrijednosti koje su se u prethodnim primjerima pokazale najboljima. Inicijalne težine neuronskih mreža početne generacije biraju se iz normalne distribucije.

Na temelju performansi neuronske mreže u igranju računalne igre, vidljivo je da je kombinacija onih vjerojatnosti dodavanja i brisanja gena koje su se u prethodnim primjerima pokazale najboljima, doprinijela poboljšanju strategije inteligentnog igrača pri dohvaćanju nagrade.

Rezultati testiranja prikazani su na slici 6.8. Pokazalo se da su performanse populacije ostvarene pri mijenjanju vjerojatnosti mutacije težina neuronskih veza vrlo slične, tj. da taj faktor ne utječe mnogo na uspješnost igranja računalne igre. Strategije igranja igre neuronskih mreže bile su različite varirajući od kretanja prema nagradi bez nepotrebnih koraka postupnog približavanja do postepenog pretraživanja praznog prostora do nagrade. Rezultati su pokazali da su populacije s većom vjerojatnošću mutacije



Slika 6.8: Graf prosječne maksimalne vrijednosti dobrote populacija s različitom vjerojatnošću mutacije težina neuronskih veza

težina neuronskih veza nešto kasnije evoluirale u odnosu na one s nižim vrijednostima iste vjerojatnosti.

6.2. Računalna igra *Flappy bird*

Druga računalna igra napravljena u sklopu rada je *Flappy bird*. Cilj ove računalne igre je prolaziti kroz prepreke koje nailaze tako da igrač u pravom trenutku "skoči" u prema gore. Tijekom igranja računalne igre, igrač pada prema dnu ekrana pod utjecajem prividne sile gravitacije. U slučaju kada padne do dna ili pak dodirne vrh ekrana, igra prestaje.

Cilj primjene neuroevolucije u ovoj računalnoj igri je trenirati populaciju neuronskim mreža koje igraju računalnu igru kako bi se razvile u samostalne inteligentne igrače. Kroz računalnu igru testiran je utjecaj ulaznih parametara neuronskih mreža na uspješnost igranja igre.

6.2.1. Programsko ostvarenje

Funkcija `eval_genoms()` koja je zadužena za evoluiranje populacije neuronskih mreža izračunava vrijednosti funkcije dobrote za jedinke populacije, brine o tome koje

jedinke populacije u danom trenutku igraju računalnu igru te mijenja ponašanje igrača koji se trenira na temelju izlaznih podataka pripadne neuronske mreže.

Kroz vizualizaciju računalne igre, prikazan je postupak treniranja cijele populacije istovremeno. Jedinke populacije treniraju se u istom okruženju, tj. na istim preprekama koje su ostvarene kao objekti klase `Obstacle`. Genomi i neuronske mreže odgovaraju objektu loptice iz klase `Ball`.

Vrijednost dobrote jedinki populacije određena je vremenom koje pojedina jedinka provede igrajući igru. Sve dok igra ne prestane, igraču koji se trenira uvećava se vrijednost dobrote za 0.01.

Aktivacijska funkcija neurona je funkcija *sigmoid*. Neuronske mreže koje se treniraju imaju jedan neuron u izlaznom sloju neurona te se na temelju izlazne vrijednosti tog neurona mijenja ponašanje igrača. Računalni igrač "skače" prema gore kada je izlazna vrijednost neuronske mreže veća od 0.5.

6.2.2. Parametri i testiranje

Pri testiranju utjecaja ulaznih podataka na performanse računalnih igrača, parametri genetskog algoritma vezani uz evoluiranje neuronske mreže postavljeni su na fiksne vrijednosti:

$$velicina\_populacije = 100$$

$$vjerojatnost\_dodavanja\_neurona = 0.3$$

$$vjerojatnost\_brisanja\_neurona = 0.2$$

$$vjerojatnost\_dodavanja\_neuronske\_veze = 0.5$$

$$vjerojatnost\_brisanja\_neuronske\_veze = 0.4$$

$$vjerojatnost\_mutacije\_neuronske\_veze = 0.7$$

$$vjerojatnost\_mutacije\_aktivnosti\_neuronske\_veze = 0.01$$

Ostali parametri vidljivi su u datoteci `config-params.txt`. Broj generacija koje će se razvijati u procesu neuroevolucije postavljen je na trideset.

Pri testiranju utjecaja ulaznih parametara na performanse računalnih igrača, populacije s istim ulaznim podacima trenirane su po tri puta te su promatrane prosječne vrijednosti maksimalno ostvarene dobrote i prosječne vrijednosti dobrote po generaciji.

U prvom slučaju neuronske mreže primaju šest parametara:

1. Poziciju loptice na x osi

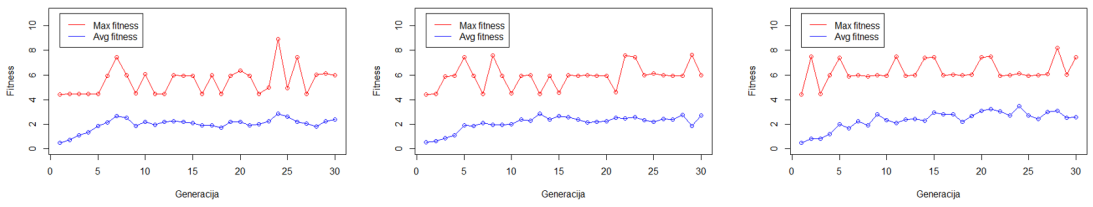
2. Poziciju loptice na y osi
3. Brzinu loptice na y osi
4. Visinu donje pregrade
5. Visinu prolaza u pregradi
6. Poziciju pregrade na x osi

U drugom slučaju neuronske mreže primaju pet parametara:

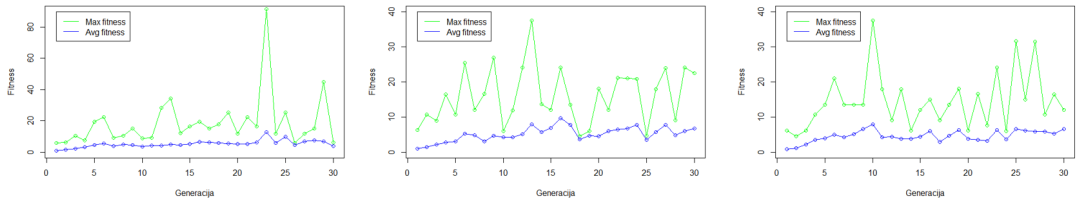
1. Poziciju loptice na y osi
2. Brzinu loptice na y osi
3. Udaljenost desnog ruba loptice do lijevog ruba prepreke na x osi
4. Udaljenost donjeg ruba loptice do vrha donje prepreke na y osi
5. Udaljenost gornjeg ruba loptice do dna gornje prepreke na y osi

6.2.3. Rezultati

Rezultati testiranja performansi neuronskih mreža dani su grafovima na slikama 6.9 i 6.10.

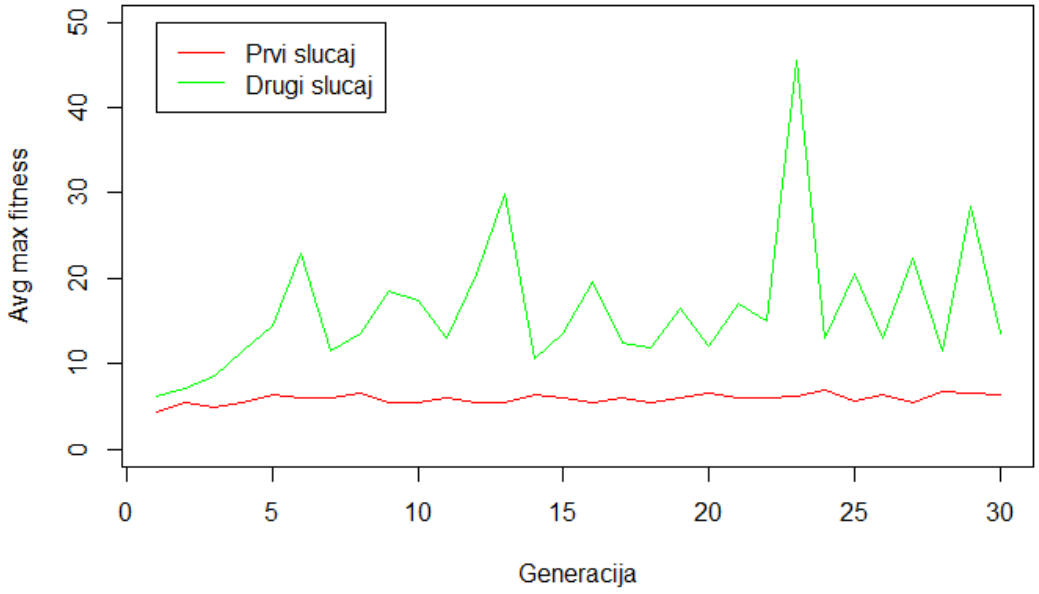


Slika 6.9: Maksimalna i prosječna vrijednost dobrote po generaciji populacija iz prvog slučaja



Slika 6.10: Maksimalna i prosječna vrijednost dobrote po generaciji populacija iz drugog slučaja

Na temelju rezultata vidljivo je da se populacije jedinki iz prvog slučaja nisu razvile te da ulazni parametri nisu dobro opisivali stanje igre u kojem se igrač nalazi te se na temelju njih nije moglo računati optimalno ponašanje. Uspjeh tih jedinki temeljio se pretežito na slučajnosti prolaza kroz slijedne prepreke koje imaju sličan položaj prolaza. S druge strane, populacije iz drugog slučaja, čije su jedinke primale pet parametara koji su bili već izračunati podaci, puno su brže evoluirale i razvile smisleno ponašanje i dobru strategiju igre.

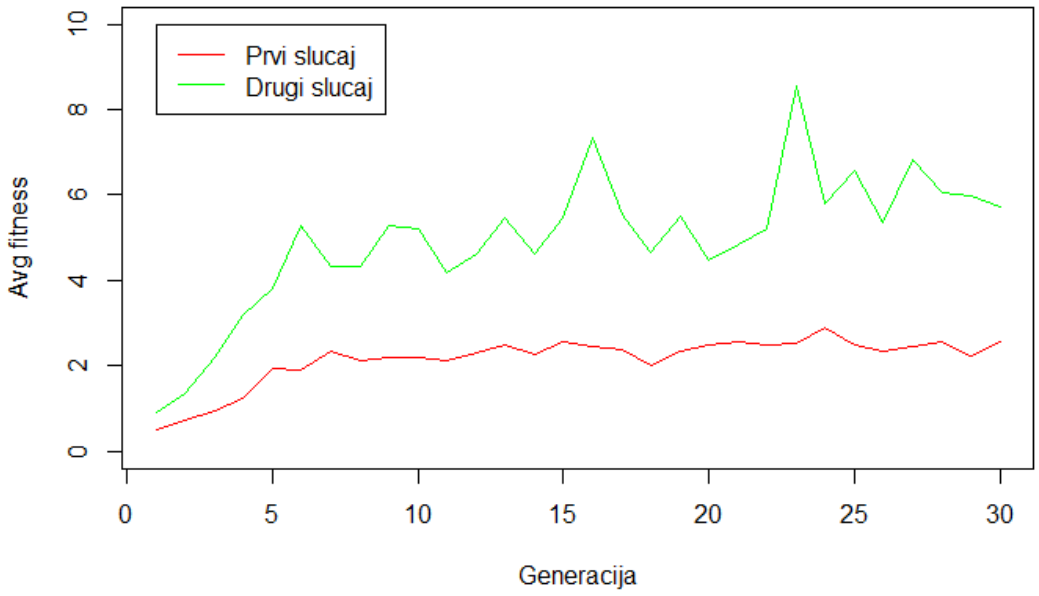


Slika 6.11: Prosječne vrijednosti maksimalne dobrote

Slike 6.11 i 6.12 prikazuju prosječne vrijednosti maksimalne dobrote i prosječne dobrote testiranih populacija po generaciji.

Vidljivo je da jedinke populacija iz drugog slučaja razviju smislenu strategiju u svega nekoliko generacija. Na temelju promatranja simulacije treniranja jedinki populacije pokazuje se da drastične oscilacije vrijednosti maksimalne dobrote između generacija proizlaze iz veličine i položaja prolaza pojedine prepreke. Jedinke populacija ponekad ne prođu kroz prepreku s malom veličinom prolaza, a ako na istu prepreku naiđu pri početku igre maksimalna vrijednost dobrote te generacije bude niska.

Na temelju rezultata može se zaključiti da je utjecaj ulaznih parametara neuronske mreže, pomoću kojih se izračunava akcija računalnog igrača, od ključne važnosti za kvalitetno evoluiranje strategija jedinki populacije. Iako se podaci predani populacijama iz drugog slučaja mogu izračunati na temelju podataka predanih populacijama u prvom slučaju, izračun istih uz pomoć evolucijskog algoritma kroz neuronsku mrežu



Slika 6.12: Prosječne vrijednosti dobrote

predstavlja dugotrajan i težak proces tijekom neuroevolucije. Stoga je bitno kao ulazne podatke postaviti vrijednosti koje bolje opisuju stanje računalne igre u kojem se igrač nalazi. Primjerice, neuronska mreža postiže bolje rezultate ako joj se kao ulazni parametar predaju izračunate udaljenosti dvaju objekta, a ne njihovi položaji u igri, iako se na temelju položaja može odrediti udaljenosti. Upravo takav primjer pokazan je u ovom eksperimentu.

7. Zaključak

Primjena neuroevolucije u računalnim igrama ostvarila je velik uspjeh. Računalne igre predstavljaju zanimljivo područje za napredak i razvoj koje privlače ne samo igrače već brojne znanstvenike i istraživače. Unatoč dugoj prošlosti, računalne igre predstavljaju inovativno i neistraženo područje u kojem se nude nova istraživanja i napredovanja.

Proteklih godina znanstvenici su primjenjivali metode evolucijskog programiranja kako bi poboljšali ponašanje različitih vrsta igrača u igrama kao što su arkadne igre u stvarnom vremenu te igre u konzoli poput Pac-Man-a. Brojni su ciljevi istraživanja razvoja igranja igre. Jedan takav trenutno aktualan je korištenje evolucijskog programiranja kako bi se generirali protivnici protiv kojih će biti zanimljivo i zabavno igrati igru, a koji neće nužno biti superiorni u odnosu na čovjeka. [4]

Iako se neuroevolucija primjenjuje u mnogim računalnim igrama, brojne su računalne igre neistražene u kontekstu neuroevolucije, a mnoge su i u nastajanju. Razvojem igrača kroz neuroevoluciju moguće je testirati i samu igru koja se razvija te uvidjeti ponašanje i uspjeh igrača. Izazovi koje donosi to područje je učenje NPC-ova različitim strategijama koje će poboljšati igranje igre i povećati privid realnosti i ljudskog ponašanja.

U eksperimentalnom dijelu ovog rada pokazao se utjecaj različitih parametara genetskog algoritma na uspješnost razvoja strategije igranja računalnih igara. Potvrđena su prednost raznolikosti gena inicijalne populacije jedinki koja se temelji na broju jedinki populacije. Također, pokazano je da su dobro definirani ulazni podaci neuronske mreže, koji opisuju stanje računalne igre, od presudne važnosti za uspjeh razvoja inteligentnih igrača.

Daljnje proširenje ovog rada moguće je ostvariti kroz testiranje karakteristika i ponašanja igrača na temelju drugačijih podražaja te mijenjanje parametara koji utječu na razvoj neuronske mreže s ciljem dodatnog optimiranja ponašanja igrača. Varijabilnost početnih struktura neuronske mreže, funkcije dobrote te ostalih parametara genetskog algoritma mogla bi doprinijeti razvoju i poboljšanju NPC-ova u računalnim igrama Snake i Flappy bird.

LITERATURA

[1] Jeff Clune, Kenneth O. Stanley, Robert T. Pennock, and Charles Ofria. On the performance of indirect encoding across the continuum of regularity. *IEEE Transactions on Evolutionary Computation*, 15(3):346–367, 2011.

[2] Jozef Fekiač, Ivan Zelinka, and Juan Burguillo. A review of methods for encoding neural network topologies in evolutionary computation. 06 2011.

[3] Derek James and Philip Tucker. A comparative analysis of simplification and complexification in the evolution of neural network topologies. 05 2022.

[4] S.M. Lucas and G. Kendall. *Evolutionary computation and games*, 2006.

[5] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[6] Evgenia Papavasileiou, Jan Cornelis, and Bart Jansen. A Systematic Literature Review of the Successors of "NeuroEvolution of Augmenting Topologies". *Evolutionary Computation*, 29(1):1–73, 03 2021.

[7] Sebastian Risi and Julian Togelius. Neuroevolution in games: State of the art and open challenges. *IEEE Transactions on Computational Intelligence and AI in Games*, PP(99), 2015. Publisher Copyright: © 2015 IEEE.

[8] Soroosh Sohangir and Bidyut Gupta. Neuroevolutionary feature selection using neat. *Journal of Software Engineering and Applications*, 07:562–570, 01 2014.

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

[10] Kenneth O. Stanley and Risto Miikkulainen. Efficient evolution of neural networks through complexification. 2004.

Optimiranje neuronskih mreža u računalnim igrama korištenjem evolucijskih algoritama

Sažetak

U ovom radu dan je uvid u primjenu neuroevolucije u računalnim igrama. Detaljno je opisan algoritam NEAT (NeuroEvolution of Augmenting Topologies) i principi na kojima se algoritam temelji. Kroz primjer izrade inteligentnih igrača i igranja jednostavnih računalnih igara prikazani su način korištenja i karakteristike evolucijskog algoritma NEAT pri kreiranju i treniranju neuronske mreže s ciljem optimalnog igranja računalne igre. Napravljen je pregled i analiza parametara genetskog algoritma te usporedba rezultata dobivenih izmjenama parametara genetskog algoritma.

Ključne riječi: neuroevolucija, računalne igre, NEAT algoritam, neuronske mreže

Optimisation of neural networks in computer games with evolutionary algorithms

Abstract

This paper provides an insight into the application of neuroevolution in computer games. The NEAT (NeuroEvolution of Augmenting Topologies) algorithm and the principles on which the algorithm is based are described in detail. Usage and characteristics of the evolutionary algorithm NEAT in creating and training a neural network with the aim of optimal computer game play are presented by evolving intelligent agents and playing simple computer games. Additionally, an overview and analysis of the parameters of the genetic algorithm and a comparison of the results obtained by changes in the parameters of the genetic algorithm were made.

Keywords: neuroevolution, computer games, NEAT algorithm, neural networks