Prilagodljiva kontrola raznolikosti pomoću genetskog algoritma s dvostrukom populacijom

Licht, Bruno

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni diplomski studij Računarstvo

PRILAGODLJIVA KONTROLA RAZNOLIKOSTI POMOĆU GENETSKOG ALGORITMA S DVOSTRUKOM POPULACIJOM

Diplomski rad

Bruno Licht

Osijek, 2024

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za ocjenu diplomskog rada na sveučilišnom diplomskom studiju

Ocjena diplomskog rada na sveučilišnom diplomskom studiju

Ime i prezime pristupnika:	Bruno Licht
Studij, smjer:	Sveučilišni diplomski studij Računarstvo
Mat. br. pristupnika, god.	D1299R, 07.10.2022.
JMBAG:	0165082019
Mentor:	izv. prof. dr. sc. Časlav Livada
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	prof. dr. sc. Krešimir Nenadić
Član Povjerenstva 1:	izv. prof. dr. sc. Časlav Livada
Član Povjerenstva 2:	izv. prof. dr. sc. Alfonzo Baumgartner
Naslov diplomskog rada:	Prilagodljiva kontrola raznolikosti pomoću genetskog algoritma s dvostrukom populacijom
Znanstvena grana diplomskog rada:	**Umjetna inteligencija (zn. polje računarstvo)**
Zadatak diplomskog rada:	U ovom radu potrebno je proučiti teoriju i implementirati genetski algoritam. Fokus će biti na stvaranju virtualnog svijeta u Unityju, gdje će se entiteti, podijeljeni prema svojim genetskim karakteristikama, natjecati za resurse.. Treba razviti aplikaciju koja koristi principe umjetne inteligencije za donošenje odluka entiteta. Iterativnim ciklusima natjecanja i stvaranja novih grupa temeljenih na pobjedničkim entitetima treba demonstrirati evoluciju u virtualnom okruženju. Tema rezervirana za studenta: Bruno Licht
Datum ocjene pismenog dijela diplomskog rada od strane mentora:	23.09.2024.
Ocjena pismenog dijela diplomskog rada od strane mentora:	Vrlo dobar (4)
Datum obrane diplomskog rada:	03.10.2024.
Ocjena usmenog dijela diplomskog rada (obrane):	Vrlo dobar (4)
Ukupna ocjena diplomskog rada:	Vrlo dobar (4)
Datum potvrde mentora o predaji konačne verzije diplomskog rada čime je pristupnik završio sveučilišni diplomski studij:	08.10.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 08.10.2024.

Ime i prezime Pristupnika:	Bruno Licht
Studij:	Sveučilišni diplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D1299R, 07.10.2022.
Turnitin podudaranje [%]:	1

Ovom izjavom izjavljujem da je rad pod nazivom**: Prilagodljiva kontrola raznolikosti pomoću genetskog algoritma s dvostrukom populacijom**

izrađen pod vodstvom mentora izv. prof. dr. sc. Časlav Livada

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

SADRŽAJ:

1. UVOD

Problem koji se koristi kako bi se na jednostavan način demonstrirala korisnost genetskog algoritma se opisuje na idući način:

Za proizvoljan broj populacija (u ovom slučaju dvije) u virtualnom svijetu, potrebno je omogućiti interakciju s okolinom. Glavni zadatak jedinki populacije je skupiti što više bodova a to mogu postići na dva načina: prikupljanjem resursa koji su razbacani po virtualnom svijetu ili napadanjem članova drugih populacija. Jedinke imaju pravo napadati jedinke druge populacije ako se nalaze na određenoj udaljenosti. Članovi populacije koji preostanu na kraju generacijskog ciklusa su sortirani po bodovima i oni s najviše bodova su prvi na redu za stvaranje novih jedinki. Karakteristike jedinki su definirane kroz gene: snaga – koliko resursa može prenijeti i koliko je dobar u borbi, brzina – koliko se brzo kreće i vid – koliko daleko mete zahtjeva mogu biti da bi jedinka reagirala, te kroz prioritete koji diktiraju način ponašanja: napad, obrana i resursi. Jedinke na početku svake generacije odlučuju na koji će resurs ići tijekom trajanja jednog ciklusa. Jedinke također imaju mogućnost tražiti od drugih jedinki da joj pomognu u slučaju napada, a neuronska mreža koja upravlja jednom populacijom ima sposobnost naređivati svojim jedinkama ovisno o situaciji u ciklusu. Nakon što ciklus završi, preživjele jedinke unutar generacije se križaju s kontrolnom populacijom na temelju genetskog algoritma dvostruke populacije te se ciklusi ponavljaju sve dok se program ne zaustavi.

Kako ovo nije vrsta problema koji se inače koristi za uspoređivanje performansi različitih algoritama iste vrste, nego ručno osmišljen zadatak autora, ne postoji rad koji se može koristiti kao mjera uspjeha implementacije algoritma. Rad dakle služi kao demonstracija rješavanja zadatka koji uključuje traženje najboljeg rješenja.

U ovom radu će se dublje dotaći tema genetskog algoritma i neuronske mreže, spomenuti osnovna ideja o njihovom radu, te provesti detaljna analiza o razvoju rješenja problema. Također će se prikazati rezultati izvedbe genetskog algoritma i navesti moguće utjecaje na njih na temelju implementacije.

2. PREGLED PODRUČJA TEME

Za genetski algoritam postoji velik broj pokušaja optimizacija i rješavanje ograničenja originalne ideje, primjerice za rješavanje problema višestrukih rezervoara i kontrolu toka vode po cjevovodima isprobane su varijante algoritma poput poboljšanog GA (Improved GA), NSGA-II i MOGA te je zaključak eksperimenata bio da su neke varijante bile bolje od originalne ideje a neke su davale lošije rezultate, što upućuje na to da za svaku varijantu genetskog algoritma postoji raspon problema koji će se bolje riješiti tim algoritmom [1].

Genetski algoritam dvojne populacije (DPGA) predstavlja oblik genetskih algoritama koji se razlikuje od prve verzije algoritma po održavanju dvije različite populacije tijekom procesa optimizacije. Ova struktura poboljšava mogućnosti istraživanja i iskorištavanja algoritma, što rezultira učinkovitijim rješenjima za složene probleme optimizacije. Najveća prednost takvog pristupa je poboljšana raznolikost kromosoma što sprječava preuranjenu konvergenciju na lokalnim optimumima i pomaže u pregledu cijelog područja pretrage.

Za DPGA postoji nekoliko istraživanja, jedan od njih se bavi analizom primjene u sustavu planiranja punjenja višestrukih električnih vozila gdje se koristi izmijenjena verzija DPGA-a kako bi se zaobišla ograničenja, dok se u drugom radu predlaže novu poboljšanu verziju genetskog algoritma dvostruke populacije (IDPGA) te ga se uspoređuje s drugim algoritmima [2,3]. Jedan rad također objašnjava utjecaj korištenja DPGA za određivanje granice *fuzzy* kontrolera [4]. Kako je algoritam nastao još 2007, prošlo je dovoljno vremena da se uoče njegove osnovne mane i da se na temelju njih izradi novi Improved DPGA koji uključuje faktore poput elitizma odnosno važnijih jedinki [5].

Ali Alajmi se bavio problemom odabira najefikasnijih svojstava genetskih algoritama kako bi riješio problem neograničene optimizacije građenja. Rad je pokazao kako faktor mutacije i križanja gena imaju najmanji utjecaj na performanse genetskog algoritma dok je veličina populacije davala najveće promjene na performanse i ubrzala dosezanje optimalnog rješenja. [6]

Također postoji rad koji se temelji na primjeni genetskog algoritma nad neuronskom mrežom u svrhu optimizacije parametra procesa [7]. Takva mreža bi bila ranije trenirana s podacima koji bi predviđali vrijednosti karakteristika proizvoda koji bi nastali kao rezultat procesa.

Unity je prepoznat kao dobar odabir u vizualizaciji problema poput problema putujućeg trgovca gdje se problem sastoji od koordinata čije je rješenje puno lakše interpretirati u vidljivim koracima i uz pomoć dodatnih metrika [8].

3. TEORETSKA PODLOGA

Kako bi se što bolje razumjela problematika ovog rada potrebno je obratiti pozornost na dva ključna pojma: **genetski algoritam** koji predstavlja srž ove teme i **neuronska mreža.** Ova dva pojma iako nisu nužno povezana , ako se pravilno koriste mogu dati zanimljive rezultate.

3.1 Genetski algoritam

Genetski algoritmi spadaju pod optimizacijske i pretraživačke algoritme, a temeljni princip rada je inspiriran prirodnim odabirom i genetikom. Veoma su pogodni za rješavanje kompleksnih optimizacijskih problema gdje nije u potpunosti poznat utjecaj vrijednost ulaznih podataka (bez obzira na njihovu veličinu) na dani problem.

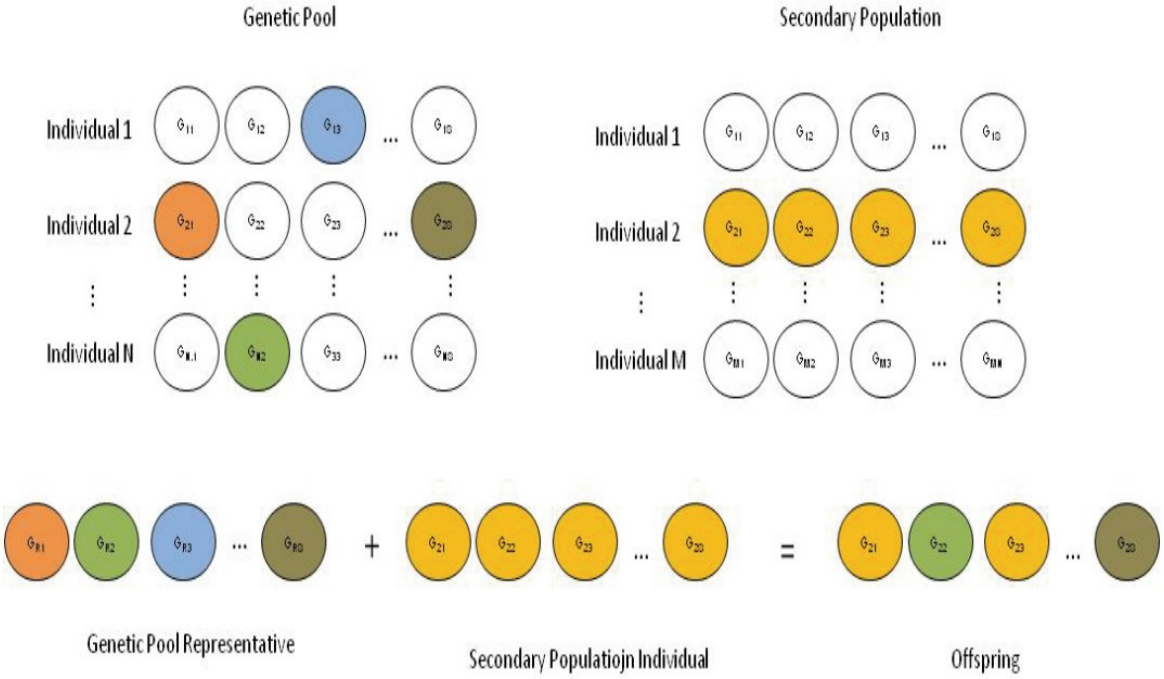
Kao što se može zaključiti iz imena, genetski algoritam se zasniva na skupini gena odnosno kromosomu koji zapravo predstavlja niz vrijednosti koji čine potencijalno rješenje problema. Skup takvih rješenja se naziva **populacija**. Određivanje kvalitete rješenja se odvija pomoću *fitness* funkcije**,** gdje se za svaki novi problem mora dizajnirati nova funkcija.

Genetski algoritam se izvodi na idući način: prvo se stvara čitava populacija ili nasumično ili na temelju heuristike te im se računaju vrijednosti *fitness* funkcije. Zatim se odabiru jedinke unutar populacije koje će se koristiti za stvaranje jedinki iduće generacije. U većini metoda odabira prednost će imati jedinke s većom *fitness* vrijednosti. Stvaranje novih jedinki se izvršava pomoću križanja (engl. *crossover*) gena roditelja. Za proces miješanja postoji nekoliko metoda a u ovom radu je korišteno ujednačeno križanje (engl. *uniform crossover)* koji za svaki pojedinačni gen provjerava od kojeg roditelja će biti naslijeđen. Idući korak je mutacija koja mijenja gene jedinke populacije. Mutacija se može primijeniti na cijeloj populaciji ili samo na par jedinki i može izmijeniti samo dio gena ili pak čitav kromosom ovisno o stopi mutacije i pridonosi raznolikosti populacije. Nakon toga slijedi formiranje nove populacije, to se može izvršiti tako da se samo zamijeni dio populacije s novim jedinkama ili da se stvori potpuno nova populacija. Svi ovi koraci se ponavljaju sve dok se ne postigne neki uvjet.

Ovaj algoritam je popularan pri rješavanju optimizacijskih problema iz nekoliko razloga. Prvo, veoma ga je lagano prilagoditi na veliki broj problema bez obzira na ograničenja i tražena rješenja te zato što provodi pretragu po cijelom prostoru mogućih vrijednosti, puno su manje šanse da će “zaglaviti” na lokalnom minimumu i maksimumu. Naravno ovisno o veličini

područja pretrage i kompleksnosti problema takav algoritam može trošiti dosta računalnih resursa. Isto tako ovisno o parametrima može doći do smanjenja raznolikosti jedinki odnosno pretraživanja dijela područja što može dovesti do ne optimalnih rješenja.

U ovom radu se koristi varijanta genetskog algoritma zvana genetski algoritam dvojne populacije. Varijanta koristi populaciju koja predstavlja genetski bazen (engl. G*enetic Pool*), nadalje opisana kao kontrolna populacija, kako bi očuvala raznolikost prisutnih gena u glavnoj populaciji. To ostvaruje na idući način, umjesto čitavog područja pretrage, jedinke kontrolne populacije su podijeljene način da svaki gen jedinke može poprimiti podskup vrijednosti koji se nalaze u području pretrage. Tako se osigurava da kontrolna populaciju u potpunosti očuva genetsku raznolikost. Križanje gena se odvija između nasumično odabranog člana glavne populacije i jedinke kontrolne populacije koja nastaje od nasumično odabranih gena svih jedinki kontrolne populacije (Sl. 3.1.), te novo stvorene jedinke zamjenjuju roditelja iz glavne populacije ako imaju bolju *fitness* vrijednost od njega. Zatim se izvršava mutacija nad nasumičnim genom kontrolne populacije na način da se vrijednost gena povećava za prethodno definiranu vrijednost ali ako bi ta vrijednost izašla iz raspona vrijednosti tog gena onda se vrijednost vraća na drugu stranu raspona. Mutacija se neće izvršiti jedino u slučaju da je odabrani gen dio trenutno najboljeg rješenja. Također postoji i koncept adaptivne kontrole raznolikosti (engl. *Adaptive Diversity Control*) koji funkcionira na način da prilikom svakog prijelaza vrijednosti gena na drugu stranu raspona, se smanjuje inkrementalna vrijednost koja se koristi za mijenjanje vrijednosti gena za taj specifičan gen kontrolne populacije.



Slika 3.1. *Postupak križanja gena za DPGA [9].*

3.2 Neuronska mreža

Neuronske mreže su struktura za rješavanje problema zasnovane na principu rada mozga. Mogu rješavati razne probleme poput klasifikacije, prepoznavanje glasa ili slike te čak i igranje igara. Sastoje se od slojeva neurona koji predstavljaju jedinice koje obavljaju nekakav izračun. Prvi sloj je uvijek sloj ulaznih podataka i sastoji se od onoliko neurona koliko različitih vrsta podataka je potrebno analizirati. Nakon dolazi skriveni sloj koji se može sastojati od nekoliko slojeva i često sadrži puno više neurona nego izlazni i ulazni sloj zajedno. Taj sloj služi za izvođenje izračuna i izvlačenja obrazaca kako bi se došlo do rješenja. Zadnji sloj je izlazni sloj koji se sastoji od onoliko neurona koliko je mogućih ishoda i u njega se mapiraju vrijednosti iz skrivenog sloja. Prilikom treniranja neuronske mreže između neurona se podešavaju različiti utezi (engl. *weights*) koji predstavljaju utjecaj prijašnjeg neurona na idući. Još jedan parametar koji služi za bolje podešavanje mreže se zove pristranost (engl. *bias)*, njegova svrha je podešavanje aktivacijski funkcija unutar mreže. Aktivacijska funkcija se nalazi unutar svakog neurona i uvodi nelinearnost u mrežu te omogućuje učenje kompleksnih obrazaca. Neke od čestih funkcija su Sigmoid, Tanh, ReLU i Leaky ReLU.

Treniranje neuronske mreže se sastoji od nekoliko koraka:

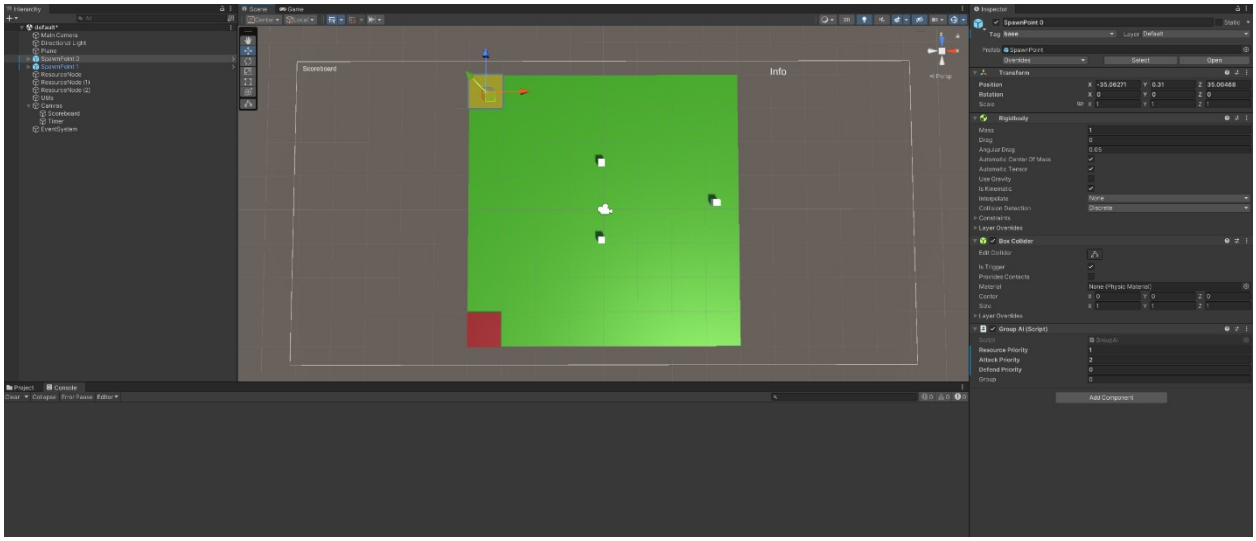
1. *Forward Propagation* – ulazni podaci prolaze kroz slojeve mreže i pri prolasku kroz svaki neuron se ulazna vrijednost množi s utezima i provlači kroz aktivacijsku funkciju kako bi se dobio izlaz
2. Funkcija greške (engl. *Loss Function*) - služi za mjerenje razlike između predviđene izlazne vrijednosti i stvarnog izlaza, često korištene funkcije uključuju srednju kvadriranu grešku (MSE) koja se koristi za regresijske zadatke i *Cross-Entropy Loss* koja je korisna za klasifikacijske probleme
3. *Backpropagation* - služi za računanje gradijenta funkcije greške uz pomoć izlaza koji su dobiveni u prvom koraku, prvo se računa koliko je svaki uteg doprinio grešci te se na temelju optimizacijskog algoritma podešavaju utezi
4. Optimizacijski algoritmi - služi za podešavanje utega i *biasa* kako bi se minimalizirala funkcija greške, među čestim algoritmima su *Gradient descent* i ADAM

Postoji nekoliko vrsta neuronskih mreža i svaka je dizajnirana za rješavanje određenih vrsta zadataka. Najjednostavnija od njih je *Feedforward Neural Network* (FNN) gdje veze između neurona ne tvore kružnu vezu nego se informacije kreću ravno od ulaza do izlaza mreže. Takva mreža je dovoljna za potrebe ovog rada jer zbog jednostavnosti problema za koji se koristi. Konvolucijske neuronske mreže (CNN) su jako efektivne u zadacima obrade slike a to postižu korištenjem konvolucijskih slojeva koji koriste filtere nad ulaznim podacima kako bi otkrile značajke slike poput rubova i tekstura. Također koristi *pooling* i potpuno povezane slojeve kako bi smanjili dimenzionalnost podataka odnosno potpuno povezali dva sloja neurona.

4. RJEŠENJE

4.1 Završna implementacija

Na slici 4.1. je dan prikaz sučelja kada se otvori Unity projekt u kojemu se nalazi implementacija genetskog algoritma. Sastoji se od nekoliko ključnih stvari: *SpawnPoint* objekta, koji se koristi za definiranje mjesta stvaranja novih jedinki unutar iste grupe i mjesto povratka resursa i *ResourceNode* objekta, koji omogućuje jedinkama prikupljanje resursa kada se nalaze dovoljno blizu. *SpawnPoint* objekti također sadrže promjenjive vrijednosti prioriteta koje će jedinke koje se stvore na njenom mjestu imati. Prioriteti resursa, napada i obrane definiraju ponašanje jedinki tijekom trajanja simulacije ali također utječe i na odluke koje donosi *GroupAI*.

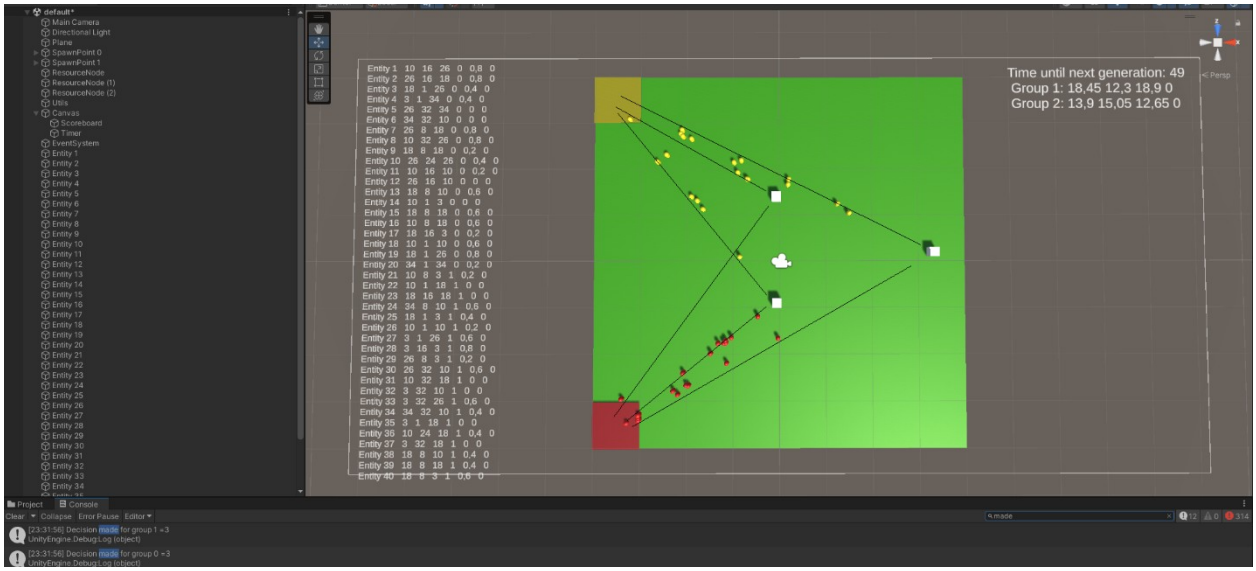


Slika 4.1. *Default sadržaj virtualnog svijeta unutar Unity projekta.*

GroupAI je neuronska mreža zadužena za praćenje stanja jedinki pod njezinom kontrolom i prosljeđivanje zahtjeva za radnjom koje dolaze jedinki. Također ima sposobnost naređivanja svojim jedinkama u određenim intervalima ako odluči da ima potrebe. Za treniranje ovog modela korišten je ručno napravljen skup podataka od 216 redova koji sadrže veliki raspon situacija koje neuronska mreže može očekivati kao ulaz. Naravno korišteni su samo češći i manje rjeđi ulazi kako bi se izbjeglo pretjerano treniranje modela a kako ne bi došlo do nestabilnih rezultata genetskog algoritma koji bi iskrivljavali sliku utjecaja pojedinih varijabli. Uz to tijekom trajanja eksperimenta se ne koriste podaci dobiveni tijekom rada kako mreža ne bi neočekivano utjecala na simulaciju. Korišteni parametri za treniranje neuronske mreže uključuju: trenutnu veličinu populacije, ukupni rezultat populacije, prosječne vrijednosti gena snage, vida i brzine unutar populacije , prioriteti ponašanja i preostalo vrijeme generacije, dakle ulazni sloj mreže se sastoji

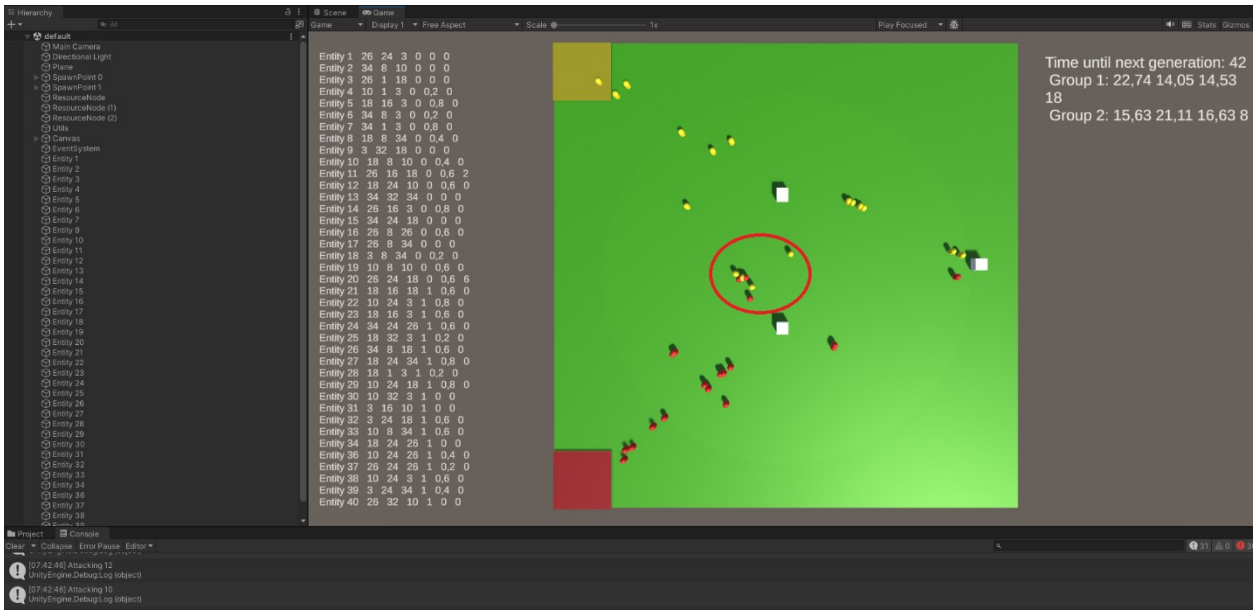
od 9 neurona. Izlazni sloj se sastoji od 4 neurona kako bi omogućio poziv funkcija namijenjene za davanje naredbi grupi, a skriveni sloj se sastoji od 7 neurona kako bi se postigla optimalna struktura mreže [10]. Unatoč dužem vremenu treniranja i boljim alternativama koristi se sigmoidna aktivacijska funkcija, prvotno zbog manje kompleksnosti neuronske mreže na koju odabir funkcije ne bi imao neki prevelik utjecaj što se tiče razina performansi.

Pokretanjem simulacije stvaraju se jedinke za svaku ne kontrolnu populaciju, dalje u tekstu grupa, koja postoji (ovisi o broju *SpawnPoint*a) te onda prate jednostavan obrazac ponašanja, odlazak do čvora resursa te povratak nazad s resursima. Na slici 4.2. može se primijetiti kako je nastalo 40 novih jedinki te pišu podaci poput vrijednosti gena, kojem timu pripadaju, prema kojem čvoru se kreću i rezultata kako bi se lakše moglo nadzirati cjelokupno stanje tijekom trajanja trenutne generacije čije se preostalo vrijeme može vidjeti na desnoj strani. Tamo se također mogu naći prosječne vrijednosti gena pojedinih grupa i njihov ukupni rezultat. Na istoj slici su također iscrtani pravci kretanja kojima jedinke putuju do pojedinih čvorova na početku generacije. Na donjem dijelu slike se mogu pratiti bilješke (engl. *logs*) koji olakšavaju shvaćanje odvijanja simulacije. Bilješka koji se nalazi na vrhu označava odluku koju je donio *GroupAI* kako bi pokušao poboljšati stanje grupe za koju je zadužen.



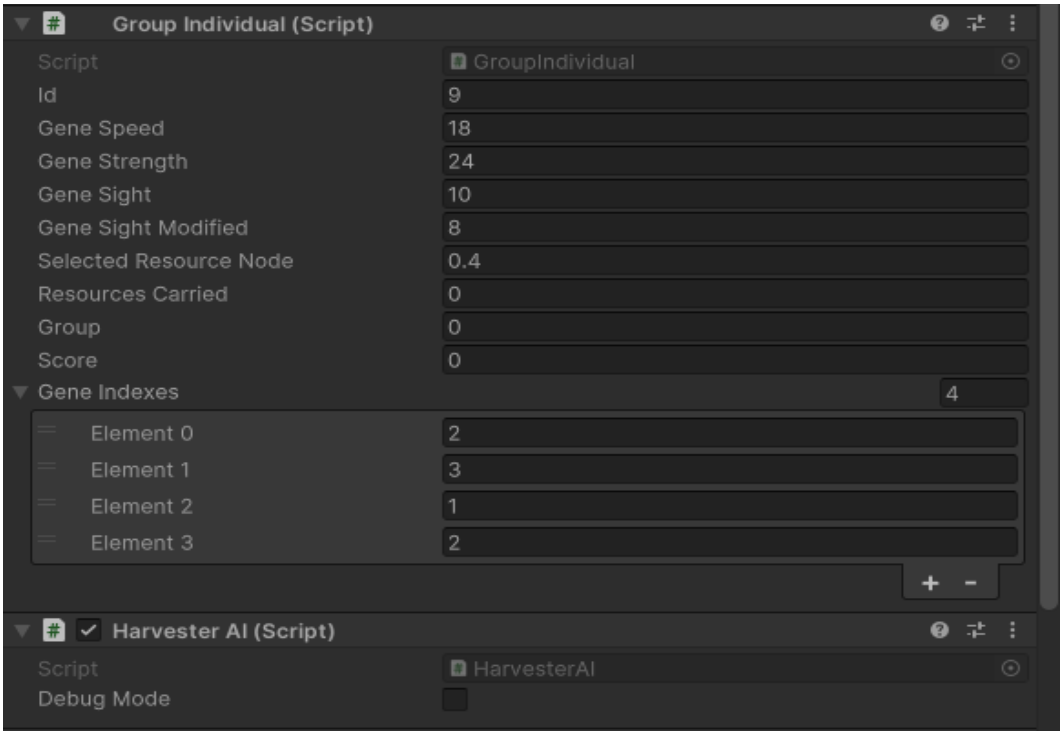
Slika 4.2. *Početak simulacije.*

Na slici 4.3. unutar crveno označenog područja može se vidjeti rezultat toga, jedinka je zbog sukoba s drugom jedinkom zatražila pomoć od svoje grupe te su se obližnje jedinke ovisno o stanju u kojima se nalaze odazvale na poziv.



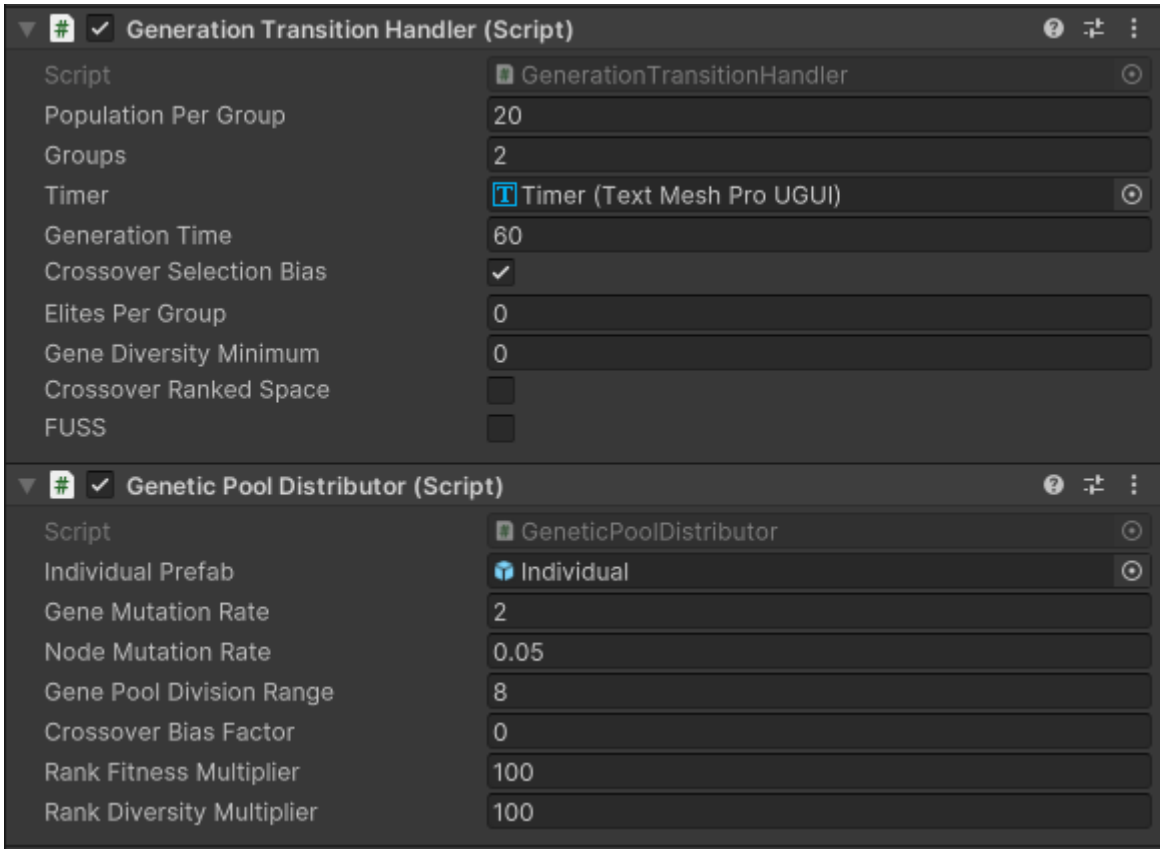
Slika 4.3. Sukob jedinki zbog prioriteta i presjeka putanja kretanja.

Na slici 4.4. se mogu naći dodatni detalji vezani uz jedinku. Osim vrijednosti gena, nije preporučljivo mijenjati nijednu vrijednost zbog nepredvidljivog ponašanja. Indeksi gena (engl. *Gene Indexes*) predstavljaju položaj gena u kontrolnoj populaciji unutar čijeg raspona bi vrijednost gena jedinke pripadala te služi za postavljanje ljestvica. Ako se naiđe na nepoželjno ponašanje, može se uključiti mod ispravljanja grešaka (engl. *Debug Mode*) opcija koja će u bilješkama ispisivati dodatne informacije o unutarnjem radu jedinki.



Slika 4.4. *Podaci o jedinki.*

Klikom na *Utils* objekt na desnoj strani sučelja se pojavljuju opcije (Sl. 4.5.) koje služe za podešavanje vrijednosti simulacije poput količine jedinki u pojedinoj grupi i koliko traje jedna generacija. Važno je naglasiti kako se može odabrati samo jedna od opcija koje su objašnjenje dalje u radu. Od najvećeg značaja su stopa mutacije gena (engl. *Gene Mutation Rate*) i stopa mutacije čvora (*engl. Node Mutation Rate*) koji direktno utječu na rad genetskog algoritma i raznolikost populacije. Raspon podjele genetskog bazena (engl. *Gene Pool Division Range*) služi za definiranje raspona vrijednosti koje jedinka kontrolne populacije može poprimiti, primjerice 0 do 8, ..., 33 do 40 (Sl. 4.6.). Sve ostale varijable će biti objašnjene s metodom u kojoj se koriste.



Slika 4.5. *Postavke simulacije.*

```
int limit = 40;
float nodeLimit = 1.0F;
int poolIndividualCount = (int)Math.Ceiling((float)limit / genePoolDivisionRange) + 1;
int[] geneValueRangeEnds = new int[poolIndividualCount];
double[] nodeValueRangeEnds = new double[poolIndividualCount];

int count = poolIndividualCount - 1;
for (int i = 0; i <= count; i++)
{
    if (i == count)
    {
        geneValueRangeEnds.SetValue(1, i);
    }
    else
    {
        geneValueRangeEnds.SetValue(limit - (genePoolDivisionRange * i), i);
    }
    nodeValueRangeEnds.SetValue(nodeLimit - (nodeLimit / count * i), i);
}
for (int i = count; i > 0; i--)
{
    geneticPoolIndividuals.Add(new GeneticPoolIndividual(geneValueRangeEnds[i] + 1, geneValueRangeEnds[i - 1], nodeValueRangeEnds[i], nodeValueRangeEnds[i - 1]));
}
```

Slika 4.6. *Podjela raspona gena za kontrolnu populaciju.*

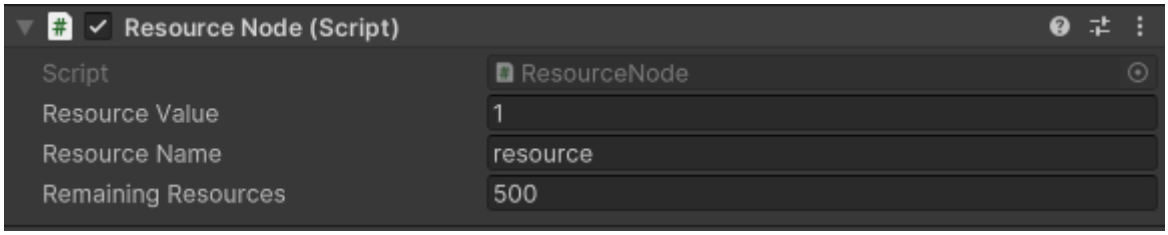
Nakon što završi trajanje generacije, ovisno o odabranoj opciji mijenja se način odabira jedinki iz grupe čiji će se geni koristiti za križanje sa jedinkama nastale iz kontrolne populacije. Uvedena je mogućnost utjecaja nad izborom koji će gen biti odabran za stvaranje novog pojedinca čime se može utjecati na raznolikost gena unutar populacije dok je u originalnom algoritmu odabir potpuno nasumičan. Adaptivna kontrola raznolikosti je također isključena zato što trajanje jedne simulacije traje jednu minutu i trebalo bi jako dugo vremena da bi se prošlo kroz sve gene po čitavom rasponu vrijednosti. Uz to, vrijednosti gena su postavljeni tako da se smanjenjem inkrementalne vrijednosti nad genom neće postići učinak osim ako gen dosegne određenu vrijednost. Ostatak algoritma se ponaša isto (Sl. 4.7.), postave se zastavice nad genima unutar kontrolne populacije za koje vrijede da su dio najboljih pojedinaca odnosno rješenja, mutiraju se nasumični geni kontrolne populacije izuzet označenih te se izvršava križanje pojedinaca iz glavne i kontrolne populacije sve dok se ne dosegne traženi broj pojedinaca te započinje nova generacija. Ako se želi izvorno ponašanje algoritma faktor pristranosti križanja (engl. *Crossover Bias Factor*) se postavlja na 0 i isključuje se opcija pristranosti odabira križanja (engl. C*rossover Selection Bias*) kako bi sve preživjele jedinke unutar grupe imale jednaku šansu da dio njihovog gena predstavlja populaciju.

```
public void StartSelection(int populationPerGroup, int groups)
{
    SetFlagsForBestGenes();
    MutateGeneticPool();
    GameObject[] individuals = individualTracker.GetIndividuals();
    List<GameObject> newIndividuals = new List<GameObject>();
    for (int k = 0; k < groups; k++)
    {
        List<GameObject> groupIndividuals = new List<GameObject>(individuals).Where(individual => individual.GetComponent<GroupIndividual>().group == k).ToList();
        if (groupIndividuals.Count == 0)
        {
            groupIndividuals.Add(CreateIndividualForNullPopulation(individuals[0], k));
        }
        int counter = populationPerGroup;
        while (counter > 0)
        {
            int random = UnityEngine.Random.Range(0, groupIndividuals.Count);
            counter--;
            newIndividuals.Add(MixIndividuals(groupIndividuals[random]));
        }

    }
    individualTracker.RemoveIndividuals(newIndividuals.ToArray());
    individualTracker.SetNewIndividuals(newIndividuals.ToArray());
}
```

Slika 4.7. *Implementacija genetskog algoritma dvojne populacije.*

Tijekom trajanja simulacije, jedinke mogu sakupljati bodove na dva načina: uzrokovanjem štete protivničkim jedinkama ili donošenjem resursa nazad u bazu. Obje radnje direktno ovise o genu snage i brzine ali zbog kompleksnosti ponašanja jedinki i putova kretanja, nije nužno da će najveće vrijednosti gena dati najbolje rezultate. Kako se samo jedinke koje su preživjele uzimaju u obzir za križanje gena, potrebna je optimalna kombinacija gena kako bi se istovremeno osigurala sigurnost cijele populacije a i same jedinke. Vrijednosti prikupljenih resursa se također mogu izmijeniti promjenom vrijednosti resursa (engl. *Resource Value*) (Sl. 4.8.) kako bi se manipuliralo ponašanjem jedinki, da li ići po veće bodove i riskirati konflikt ili igrati sigurno i skupljati bodove pomalo. Nadalje se mogu stvoriti scenariji gdje određeni čvorovi imaju jako ograničenu količinu resursa zbog čega bi potencijalno bili nepoželjni za odabir. Potrošnjom resursnog čvora jedinke odabiru novi čvor na temelju gena odabranog resursnog čvora (engl. *Selected Resource Node*) a ako nestanu svi čvorovi iz svijeta onda se zaustavljaju kretnje svih jedinki dok ne završi ciklus generacije.



Slika 4.8. *Podešavanje resursnih čvorova.*

Na kraju svake generacije se podaci o najboljim pojedincima iz svake grupe zapisuju u datoteku na temelju koje su izrađeni grafovi u potpoglavlju koje obrađuje rezultate.

4.2 Podešavanje simulacije

Stopa mutacije gena se treba postaviti na pozitivnu cjelobrojnu vrijednost i to manju nego raspon vrijednosti gena u jedinkama kontrolne populacije.

Stopa mutacije čvora se treba postaviti na pozitivnu vrijednost i to manju nego raspon vrijednosti gena čvora u jedinkama kontrolne populacije.

Raspon podjele genetskog bazena može sadržavati bilo koju pozitivnu cjelobrojnu vrijednost manju od 40 jer je to maksimalna vrijednost gena, ali će davati najbolje rezultate ako se postavi na vrijednost s kojom je broj 40 djeljiv.

Faktor pristranosti križanja je najbolje postaviti na vrijednost između -0.5 i 0.5. Kako se ta vrijednost pridodaje nasumičnom broju između 0 i 1, postavljanje vrijednosti izvan na veću od 0.5 će fiksirati odabir gena isključivo iz kontrolne populacije a ako je vrijednost manja od –0.5 isključivo iz ne kontrolne populacije.

Moguće su izmjene virtualnog svijeta na sljedeći način:

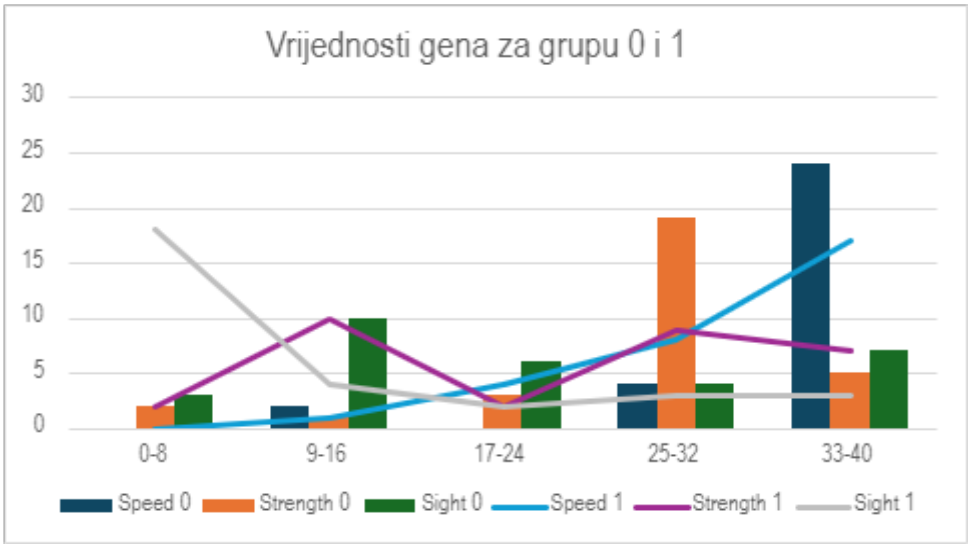
- Mogu se stvoriti dodatni resursni čvorovi jednostavnim kopiranjem postojećeg *Resource Node* objekta, mogu se i uklanjati ali je za potrebe simulacije ključno da postoji barem jedan čvor
- Mogu se dodati i dodatne grupe kopirajući *Spawn Point* objekte koji sadrže skriptu za konfiguraciju jedinki koje nastaju iz nje

4.3 Rezultati

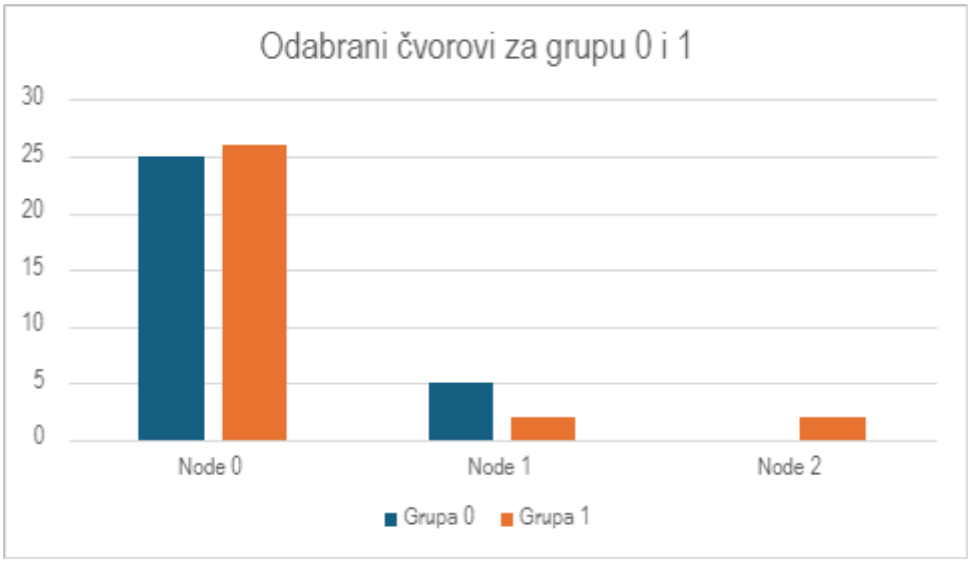
Kako bi se promatrao rezultat uporabe genetskog algoritma dvojne populacije simulacija je puštena da radi 30 generacija s 3 različite postavke prioriteta. Za svaku generaciju su zapisane najbolje jedinke iz obje grupe te su vrijednosti njihovih gena prikazani na grafovima. Postavke korištene tijekom trajanja prve tri simulacije se mogu naći na slici 4.5. te je vrijednost čvora 0 postavljena na 3.

Za slike 4.9. i 4.10. vrijedi da grupa 0 ima prioritete 1,0,2 (razina prioriteta resursa, napada i obrane) dok grupa 1 ima prioritete 1,2,0. Iz grafova se može zaključiti kako je genetski algoritam dvostruke populacije uspješno implementiran zato što se najbolje jedinke mogu naći po cijelom rasponu vrijednosti gena. Nadalje prema slikama 4.11. - 4.14. može se zaključiti da se algoritam

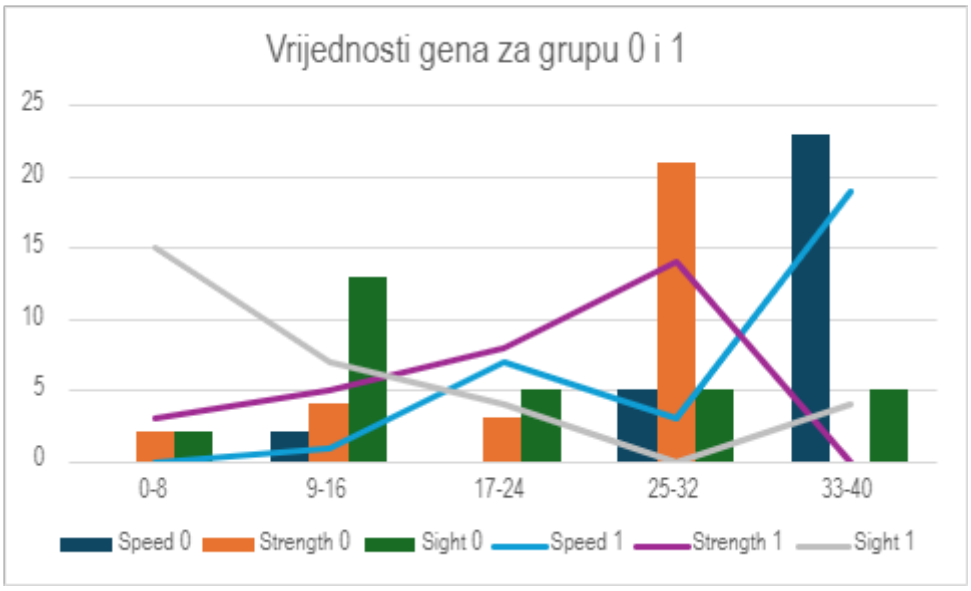
prilagođava s obzirom na prioritete koji su zadani *GroupAI-u* koja nadzire grupu. Na rezultat je također najviše utjecao odabrani čvor obje grupe jer o njemu direktno ovisi koliko će se grupe sukobljavati odnosno da li ima potrebe za određenim genima i koje vrijednosti gena daju najbolju vjerojatnost da će jedinka ostvariti najveći rezultat i preživjeti. Također prilikom promatranja grafova treba uzeti u obzir da su najboljim jedinkama proglašene one s najvećim rezultatom ali pod uvjetom da su preživjele do kraja generacije. Tako primjerice jedinke koje su jako brze i imaju slab gen vida mogu preživjeti u generacijama gdje zbog rasporeda čvorova postoji visoka šansa za sukob.



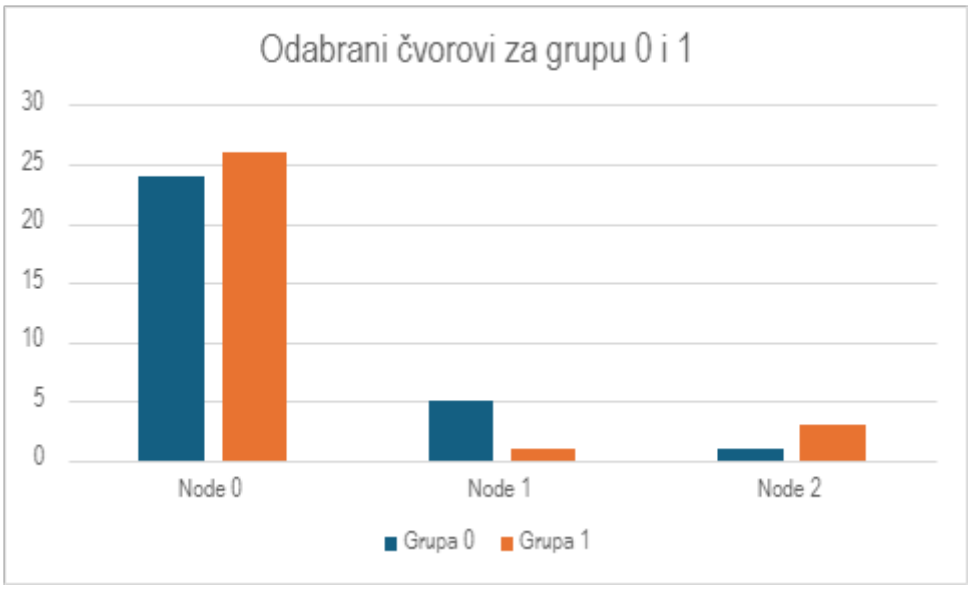
Slika 4.9. *Vrijednosti gena najboljih jedinki po grupi s prioritetima 1,0,2 (gr. 0) i 1,2,0 (gr. 1).*



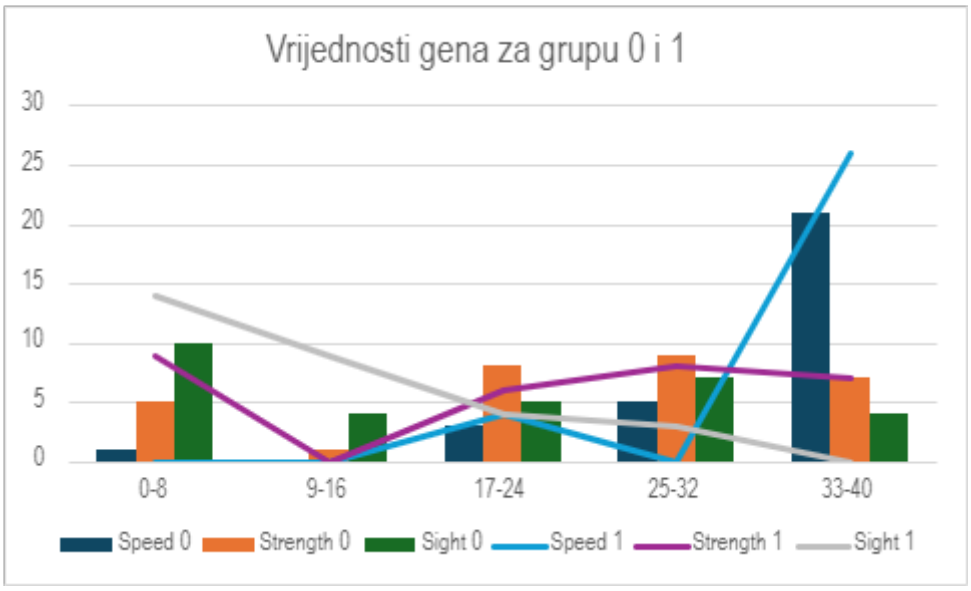
Slika 4.10. *Odabrani čvorovi najboljih jedinki po grupi s prioritetima 1,0,2 (gr. 0) i 1,2,0 (gr. 1).*



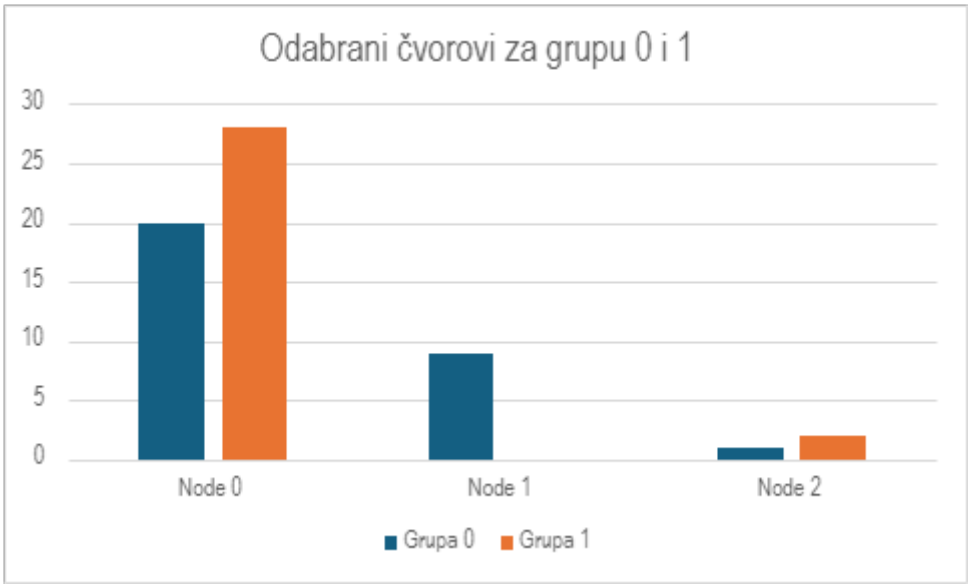
Slika 4.11. *Vrijednosti gena najboljih jedinki po grupi s prioritetima 0,1,2 (gr. 0) i 1,2,0 (gr. 1).*



Slika 4.12. *Odabrani čvorovi najboljih jedinki po grupi s prioritetima 0,1,2 (gr. 0) i 1,2,0 (gr. 1).*

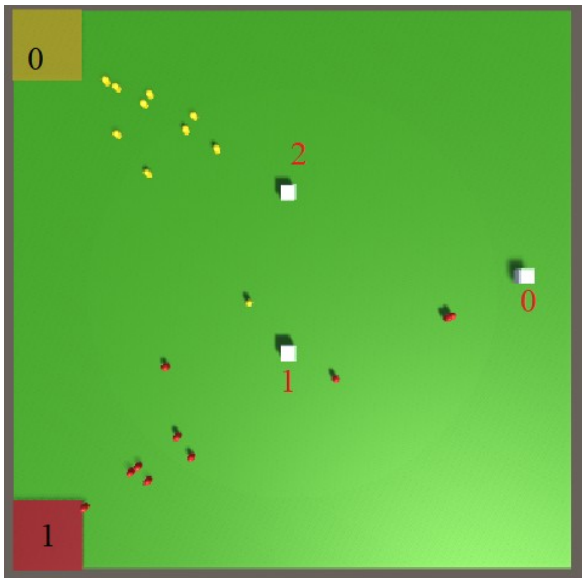


Slika 4.13. *Vrijednosti gena najboljih jedinki po grupi s prioritetima 1,2,0 (gr. 0) i 1,2,0 (gr. 1).*



Slika 4.14. *Odabrani čvorovi po grupi s prioritetima 1,2,0 (gr. 0) i 1,2,0 (gr. 1).*

Na slici 4.15. crnom bojom su označene grupe koje se spominju u rezultatima dok su crvenom bojom označeni čvorovi prema kojima se jedinke kreću. Kako je već prošlo nekoliko iteracija može se primijetiti da postoji mali broj jedinki koji se “ne drže svoje strane” odnosno većina jedinki iz grupe 0 bira čvorove 0 i 2 i ne riskiraju sukob s drugom grupom te isto vrijedi i za grupu 1 što objašnjava zašto tako veliki broj najboljih jedinki ima visoku vrijednost gena brzine.



Slika 4.15. *Peta iteracija simulacije za grupe s prioritetima 1,2,0 (gr. 0) i 1,2,0 (gr. 1).*

Kako bi pratili ima li evolucijskog napretka i kolika je raznolikost gena u svakoj generaciji rađeni su izračuni prema (4-1) i (4-2).

$$y = \sum_{i=1}^{n} \frac{x_i}{n} \tag{4-1}$$

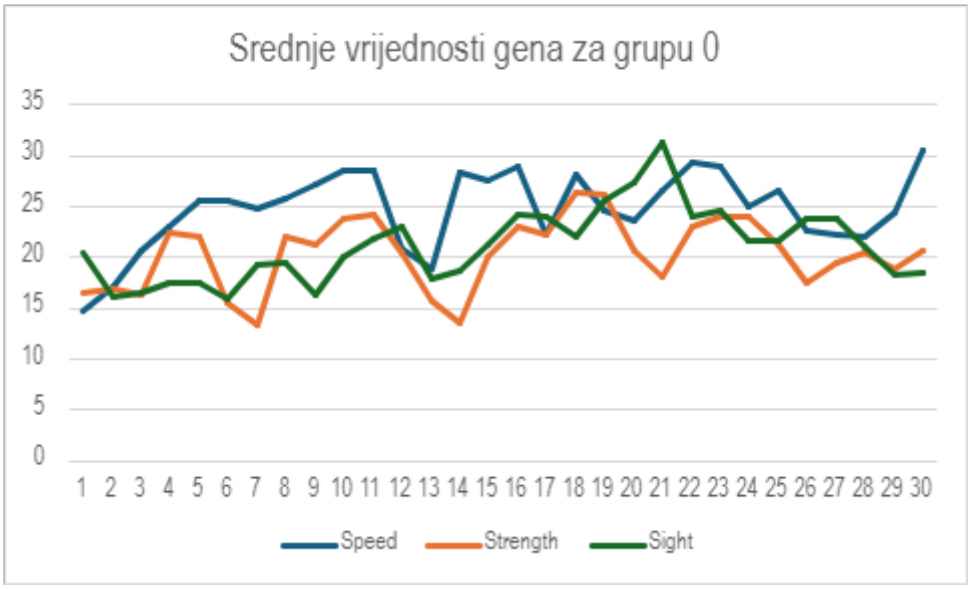
gdje je: y srednja vrijednost gena grupe jedinki, x_i predstavlja vrijednost gena jedinke dok n predstavlja broj jedinki na početku generacije

$$y = \sum_{i=1}^{n} \frac{(x_i - m)^2}{n} \tag{4-2}$$

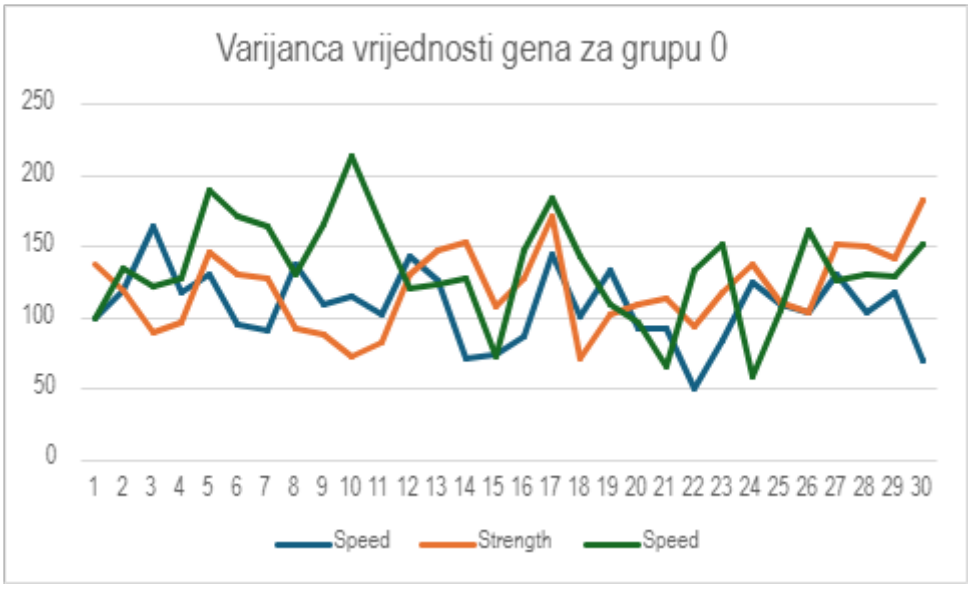
gdje je: y varijanca gena grupe jedinki, x_i predstavlja vrijednost gena jedinke, m srednju vrijednost gena unutar iste grupe dok n predstavlja broj jedinki.

Rezultati dobiveni prema slikama 4.16 - 4.19 su rezultat simulacije s postavkama prema slici 4.4. tijekom 30 generacija. Kako je vrijednost faktora pristranosti križanja postavljena na 0, ne očekuje se preveliko odstupanje vrijednosti gena od srednje vrijednosti ali također neće doći do konvergencije vrijednosti u lokalni minimum. Obje grupe su dale različite vrijednosti varijance i

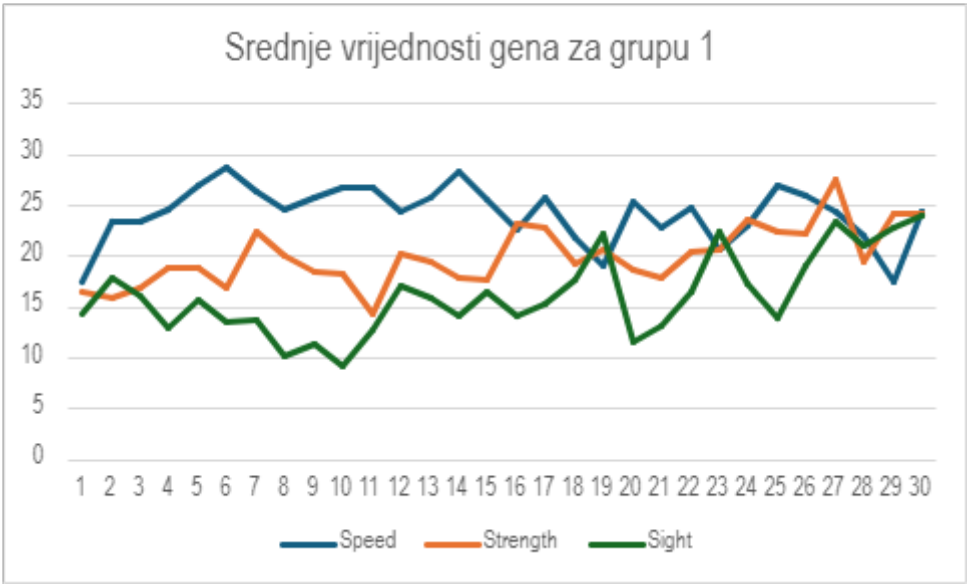
iako imaju prisutne ekstreme može se zaključiti da genetski algoritam s danim postavkama ne dopušta zadržavanje na određenom rasponu vrijednosti. To je dodatno potvrđeno činjenicom da promjene srednjih vrijednosti nisu stabilne.



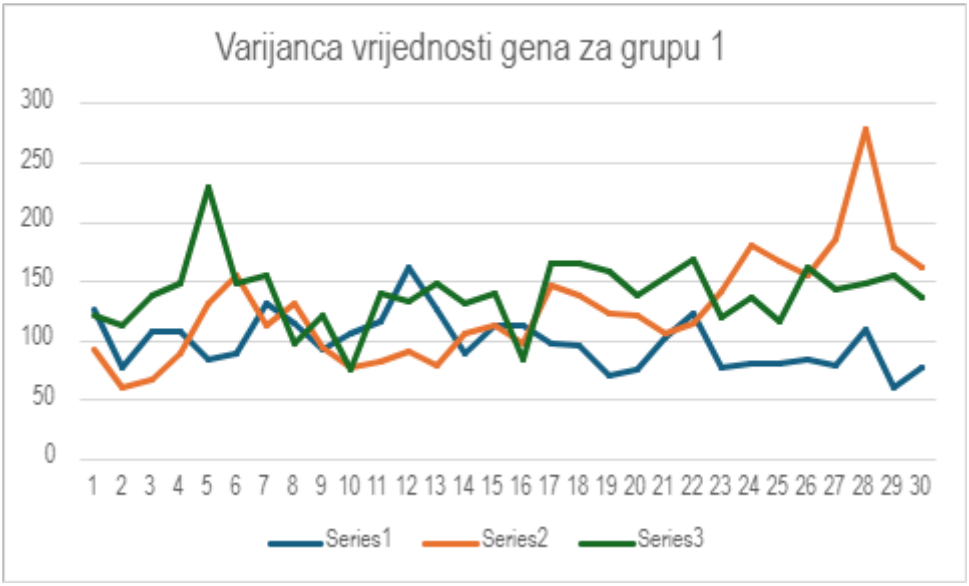
Slika 4.16. *Srednje vrijednosti gena za grupu s prioritetom 1,0,2.*



Slika 4.17. *Varijanca gena za grupu s prioritetom 1,0,2.*



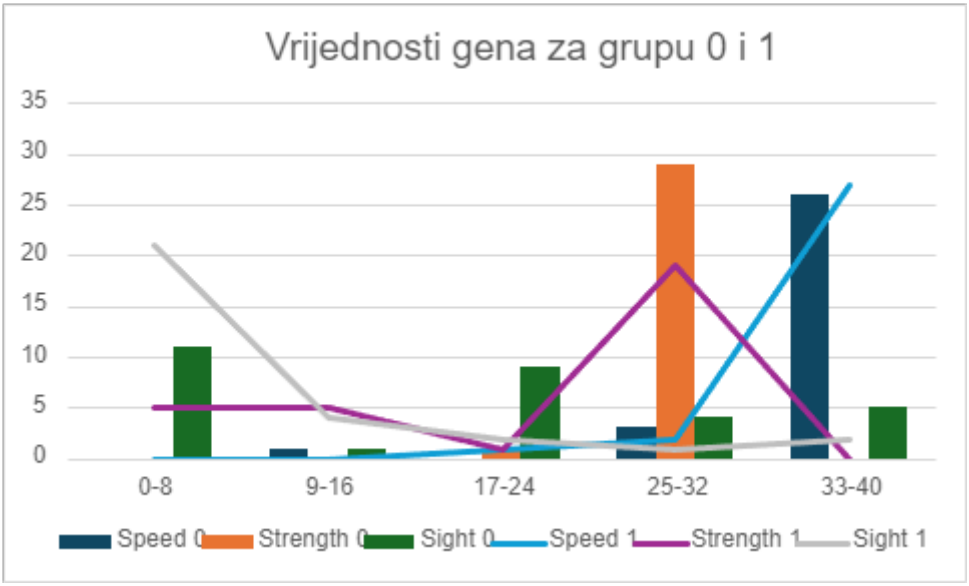
Slika 4.18. *Srednje vrijednosti gena za grupu s prioritetom 1,2,0.*



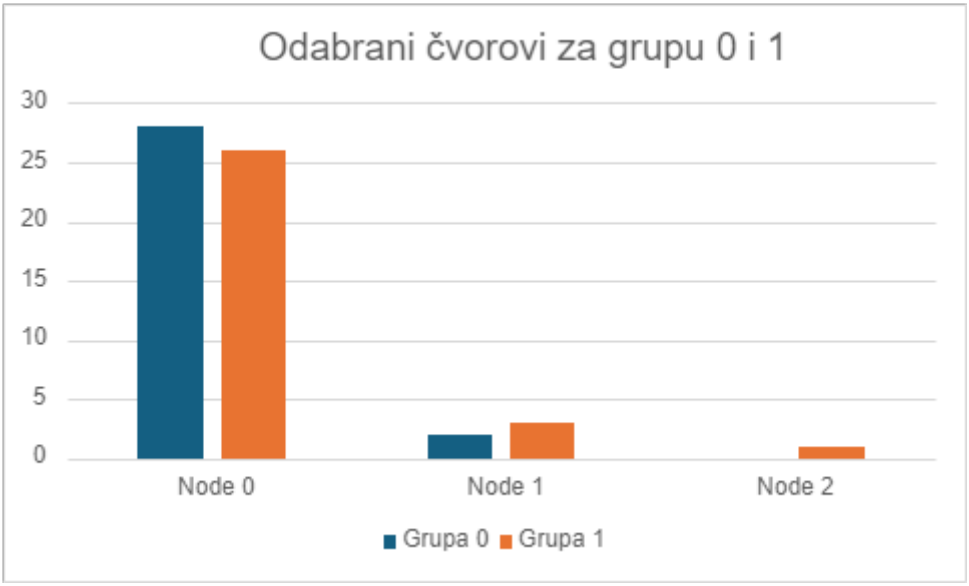
Slika 4.19. *Varijanca gena za grupu s prioritetom 1,2,0.*

Svi daljnji eksperimenti će se provoditi s prioritetima postavljenima na 1,2,0 kako bi se izbjegao utjecaj različitog ponašanja populacija na rezultat. Dodatna prednost ovakvog pristupa je i mogućnost uspoređivanja rezultata obaju grupa kako bi se moglo utvrditi da nije bilo neočekivanog ponašanja tijekom simulacije. Kao idući eksperiment, vrijednost faktora pristranosti križanja je postavljena na –0.3. Time se povećava šansa da prilikom križanja jedinki budu odabrani geni iz grupe. Rezultat toga je fokusiranje na uži raspon vrijednosti centriran oko vrijednosti gena najboljih jedinki. Prema tome s vremenom će se raznolikost vrijednosti s vremenom smanjivati i fokusirati na određenu grupu vrijednosti (Sl. 4.20.). Ta je tvrdnja dodatno potvrđena slikom 4.22. gdje se jasno vidi da je srednja vrijednost skoro svih gena porasla nakon

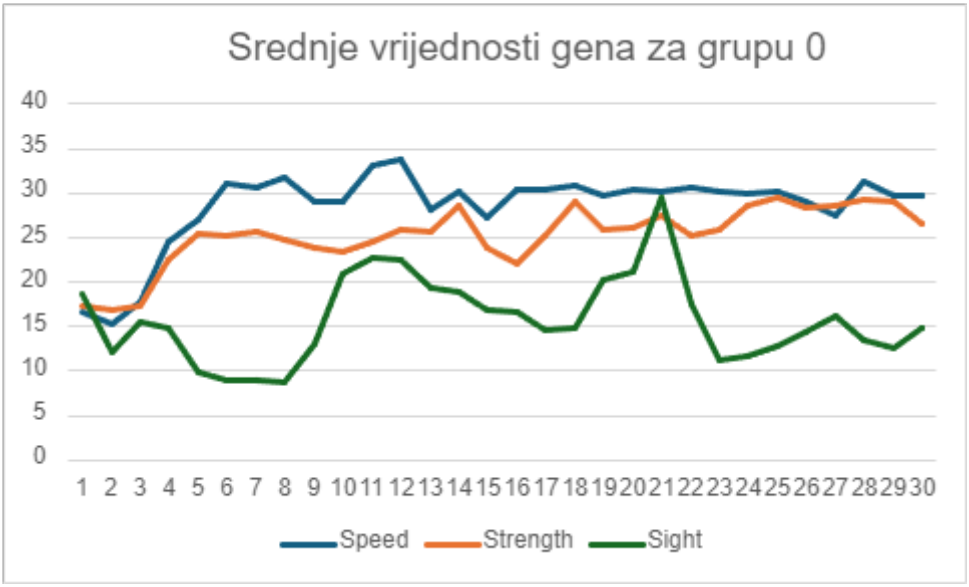
određenog broja generacija, ali rezultati prema slici 4.23. upućuju na to da još uvijek postoji velika raznolikost vrijednosti gena prilikom simulacije iako većinom prevladavaju generacije s jako grupiranim vrijednostima. Jedino gen za vid zadržava viši stupanj varijacije i nižu srednju vrijednost pretežno zato što u okruženju bez sukoba nema veliku važnost te zbog toga ne postoji optimalna vrijednost gena prema kojoj može težiti (Sl. 4.21.).



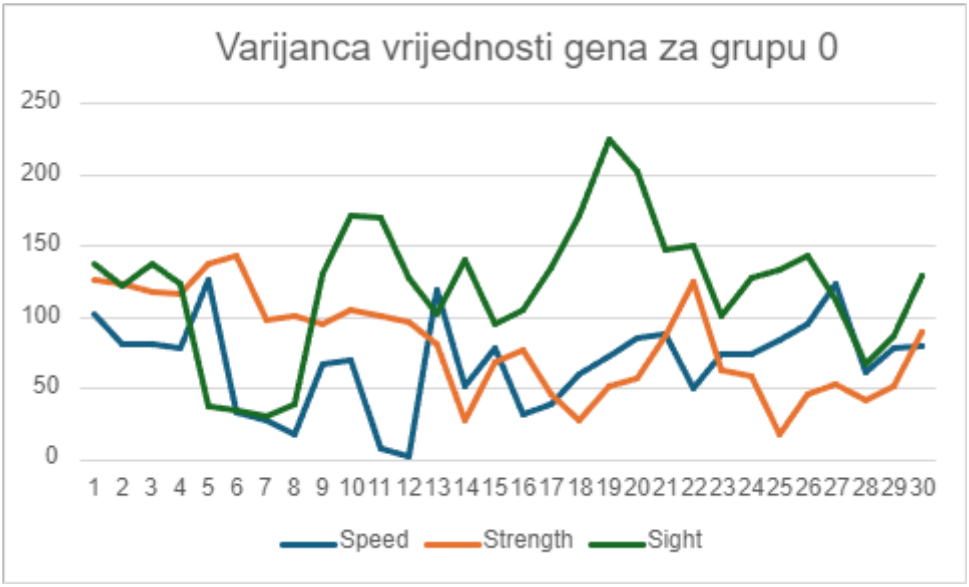
Slika 4.20. *Vrijednosti gena najboljih jedinki po grupi s prioritetima 1,2,0 (gr. 0) i 1,2,0 (gr. 1).*



Slika 4.21. *Odabrani čvorovi po grupi s prioritetima 1,2,0 (gr. 0) i 1,2,0 (gr. 1).*

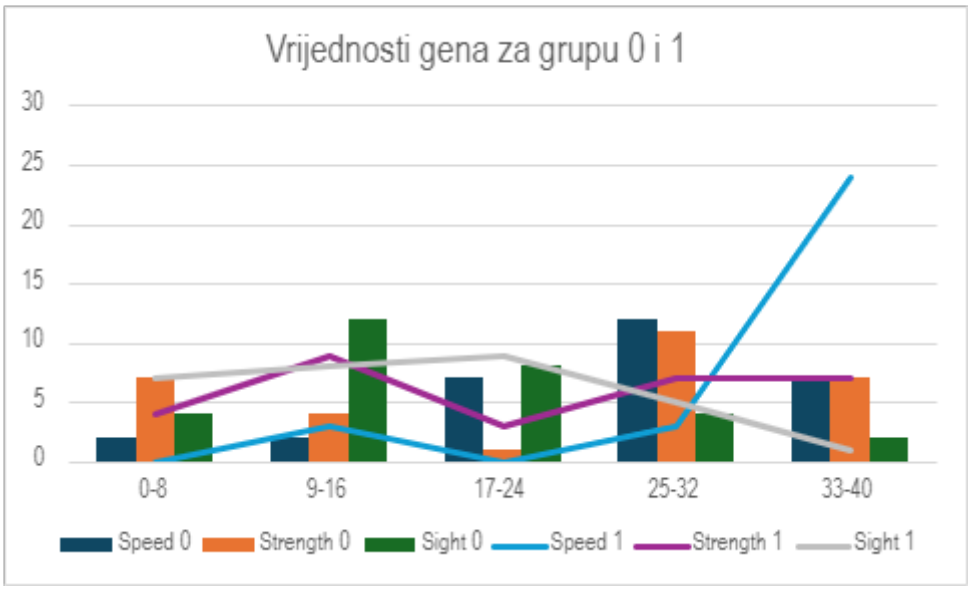


Slika 4.22. *Srednje vrijednosti gena za grupu s prioritetom 1,2,0.*

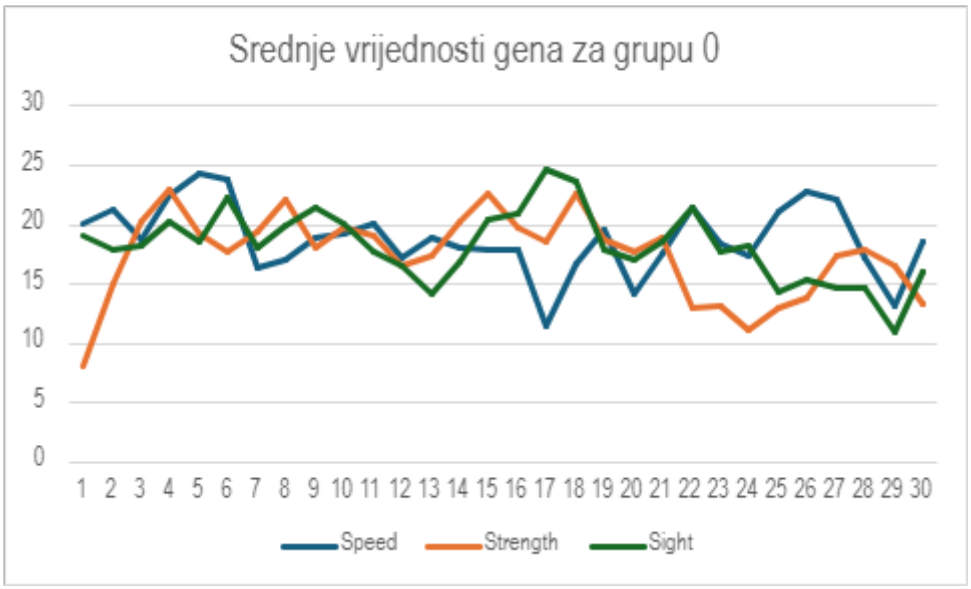


Slika 4.23. *Varijanca gena za grupu s prioritetom 1,2,0.*

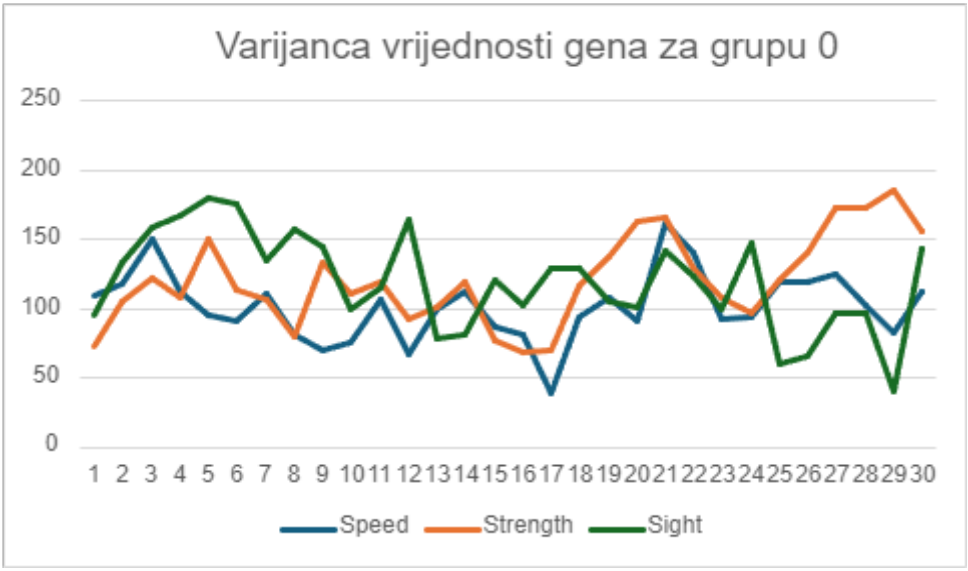
Nadalje je testirano ponašanje populacija s faktorom pristranosti križanja postavljenim na 0, a isključi se pristranost odabira križanja. Bez davanja prednosti jedinkama koje imaju veće rezultate prilikom odabira za križanje gena, dobiva se potpuno nasumičan odabir jedinki i vidi se da iako se u nekim generacijama varijacija smanji na malu vrijednost, ne ostaje dugo takva i izbjegava se konvergencija. (Sl. 4.26.). Usto, najbolje jedinke se zbog nepristranosti nalaze po cijelom rasponu vrijednosti (Sl. 4.24.). Također uspoređujući sa slikama 4.22. i 4.23. može se primijetiti da se najčešće srednje vrijednosti gena nalaze u manjem rasponu centrirane oko srednje vrijednosti raspona gena s blagim trendom pada, dok se vrijednost varijance također kaotično mijenja ali puno rjeđe se prilazi točki konvergencije (Sl. 4.25. i 4.26.).



Slika 4.24. *Vrijednosti gena najboljih jedinki za potpuno nasumičan odabir.*

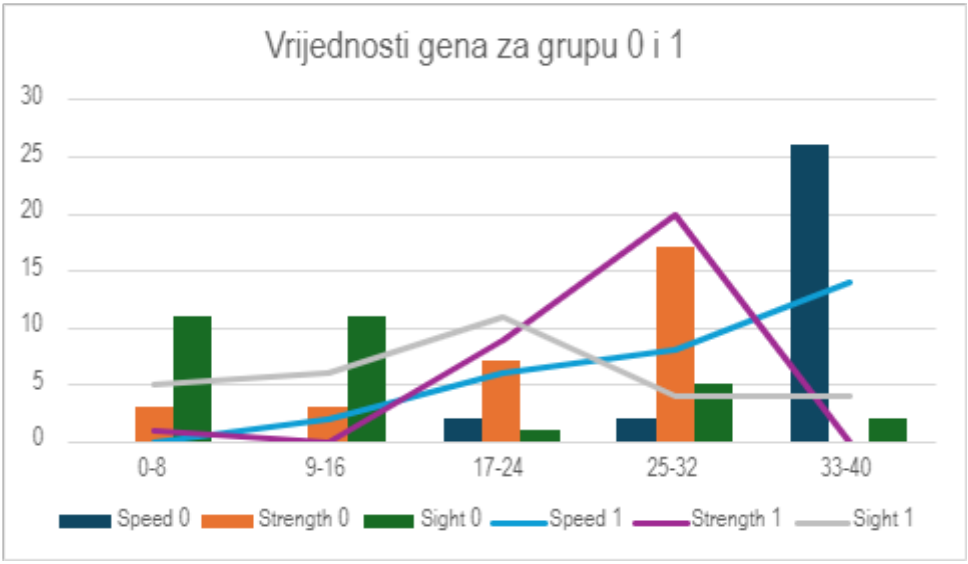


Slika 4.25. *Srednje vrijednosti gena za potpuno nasumičan odabir.*

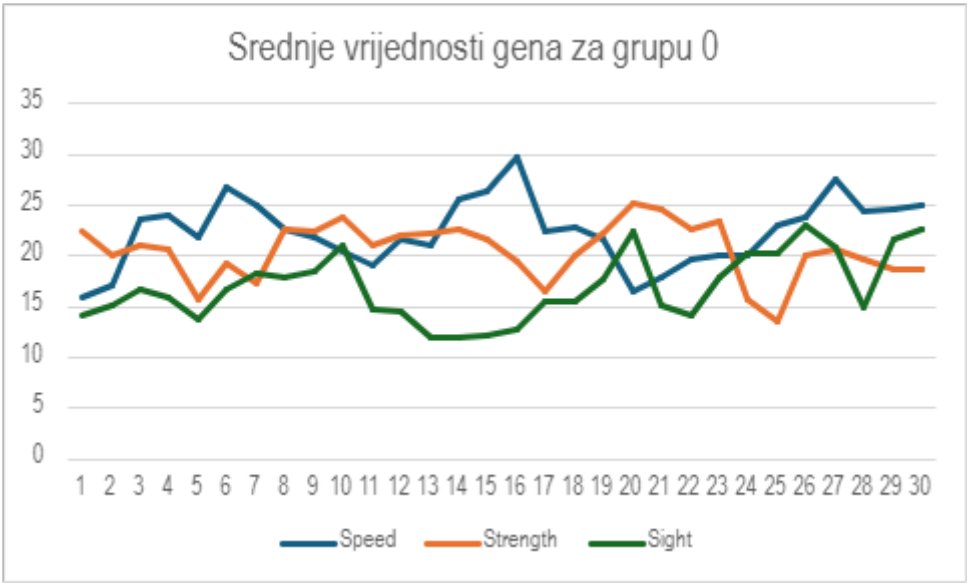


Slika 4.26. *Varijanca gena za potpuno nasumičan odabir*.

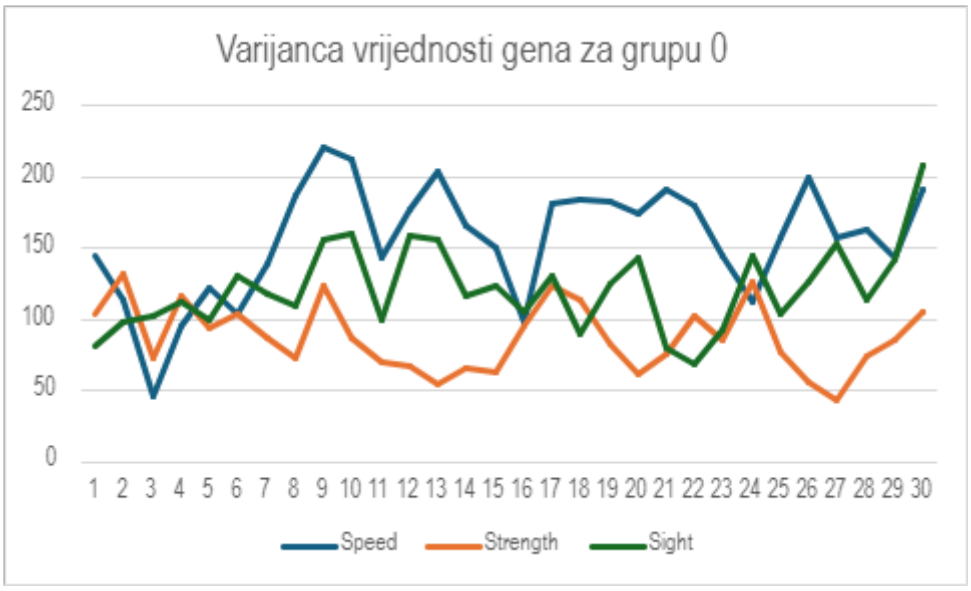
Idući eksperiment se fokusira na odabiranje dviju najboljih jedinki u svakoj grupi i njihovo prebacivanje u iduću generaciju odnosno elitizam. Odabir jedinki za križanje je u potpunosti nasumičan kao i u prethodnom eksperimentu. Kako elite čine samo 10 % grupe, težište srednjih vrijednosti se ne pomiče ni prema jednom kraju raspona vrijednosti gena a najbolje jedinke su raspoređene po cijelom rasponu vrijednosti uz blagu pristranost određenom genu (Sl. 4.27. i 4.28.). Za varijancu usporedbom slika 4.26. i 4.29. novi pristup pokazuje dosta ekstremnije promjene varijance ali u konačnoj analizi je utvrđeno da razlika nije toliko značajna. Postavljanjem broja elitnih jedinki po grupi (engl. *Elites Per Group*) na 5 (25 %) dobiva se nešto niža varijanca (Sl. 4.31.) ali je porasla srednja vrijednost (Sl. 4.30.) što upućuje na to da se područje pretrage pomiče i otežava pretraživanje vrijednosti koje se nalaze na drugom kraju. U kombinaciji sa strategijom koja potiče šire područje pretrage ovakav pristup bi zasigurno osigurao pregled čitavog područja tijekom potrage za najboljim rješenjem.



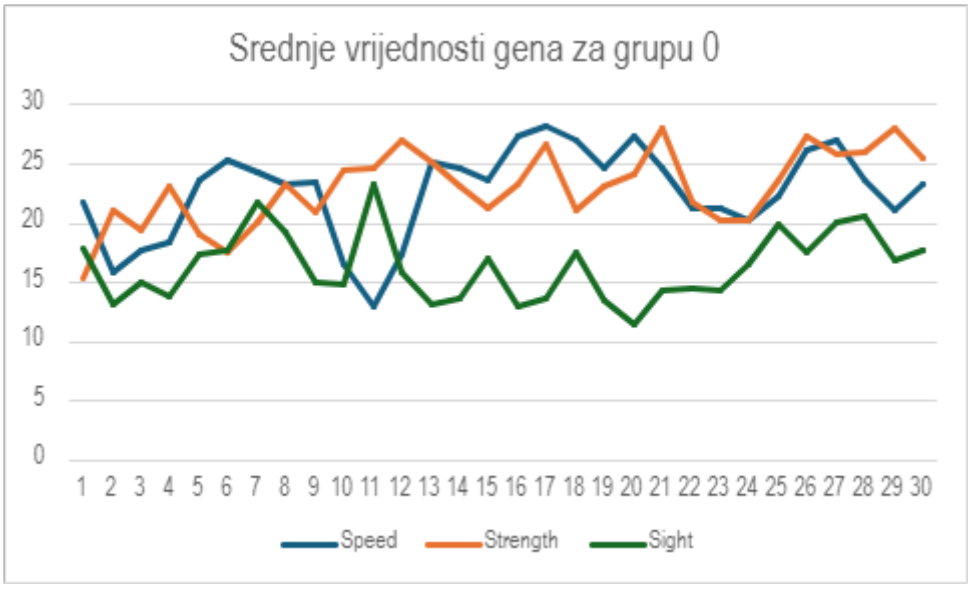
Slika 4.27. *Vrijednosti gena najboljih jedinki za dvije elitne jedinke.*



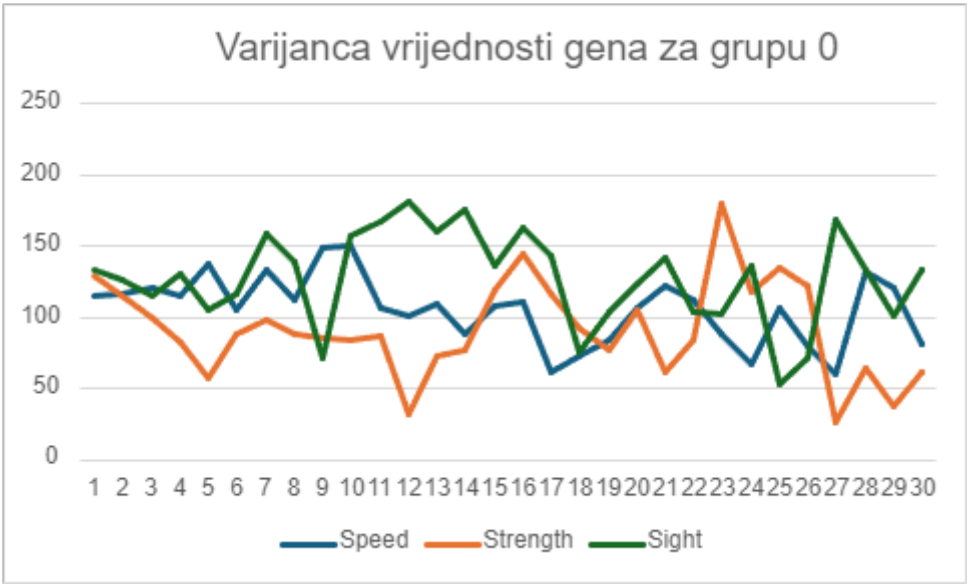
Slika 4.28. *Srednje vrijednosti gena za dvije elitne jedinke.*



Slika 4.29. *Varijanca gena za dvije elitne jedinke.*

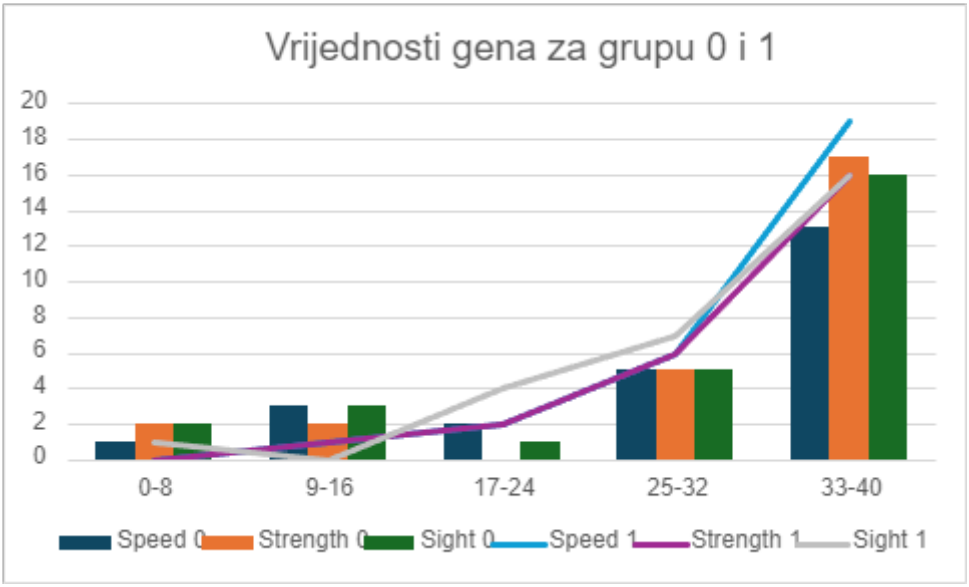


Slika 4.30. *Srednje vrijednosti gena za pet elitnih jedinki.*

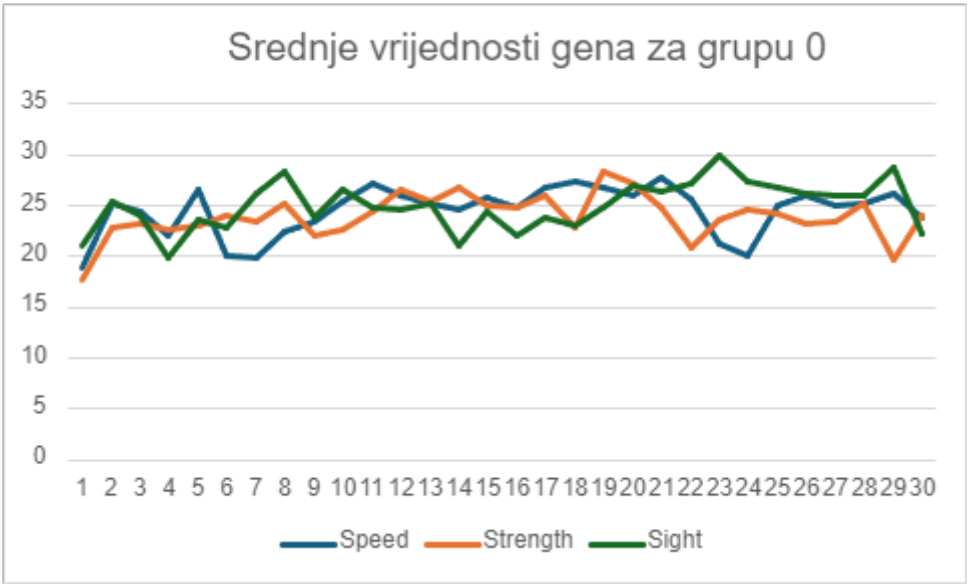


Slika 4.31. *Varijanca gena za pet elitnih jedinki.*

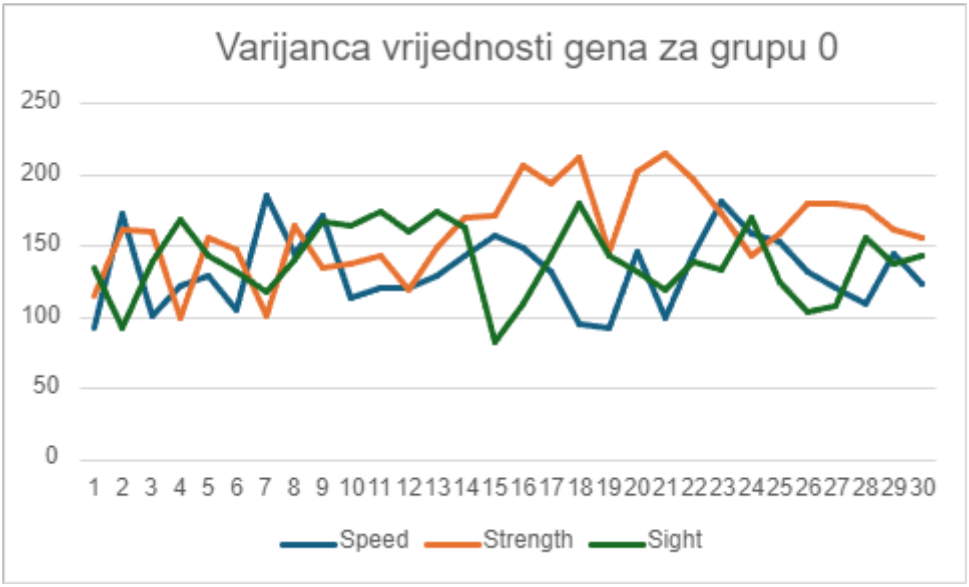
Sljedeći pristup kontroli raznolikosti gena se našao u uklanjanju sličnih gena unutar jedne grupe. Tu je ključno definirati što se smatra sličnim genom, a za potrebe ovog rada je odlučeno da će suma svih gena jedinke biti faktor za odlučivanje sličnosti. Iako to možda nije najbolja definicija, u slučaju gdje je samo bitno da se suzi mogućnost odabira gena, pokazalo se veoma djelotvornim. Simulacija je započeta vraćanjem broja elita po grupi na 0 i postavljanjem minimalne raznolikosti gena (engl. *Gene Diversity Minimum*) na 4 što znači da razlika između sume vrijednosti gena nove jedinke i postojećih jedinki unutar grupe ne smije biti manja od 4. Ako algoritam ne uspije naći takvo rješenje nakon određenog broja pokušaja, dodaje se jedinka s vrijednostima gena koje je imala prije utjecaja algoritma. Vrijednost 4 je odabrana zato što za mogućnost rasporeda vrijednosti svih gena postoji jako mala šansa da se dogodi takav slučaj (na 40 jedinki u 30 generacija se prilikom izvođenja eksperimenta dogodilo 5 puta) što je osiguralo da se područje pretrage nalazi na području cijelog raspona vrijednosti gena. Također algoritam je dizajniran da prvo istražuje sve kombinacije prema kraju raspona gena s većom vrijednosti te zatim nastavlja prema drugom kraju zbog čega rezultati vezani uz srednju vrijednost i vrijednost gena najboljih jedinki teže prema određenom kraju (Sl. 4.32. i 4.33.). S druge strane, slika 4.34. pokazuje kako je varijacija gena dosta visoka, ali ne sadržava ekstreme što upućuje na to da algoritam omogućuje kontrolirano područje pretrage.



Slika 4.32. *Vrijednosti gena najboljih jedinki pri uklanjanju sličnih gena.*



Slika 4.33. *Srednje vrijednosti gena pri uklanjanju sličnih gena.*



Slika 4.34. *Varijanca gena pri uklanjanju sličnih gena.*

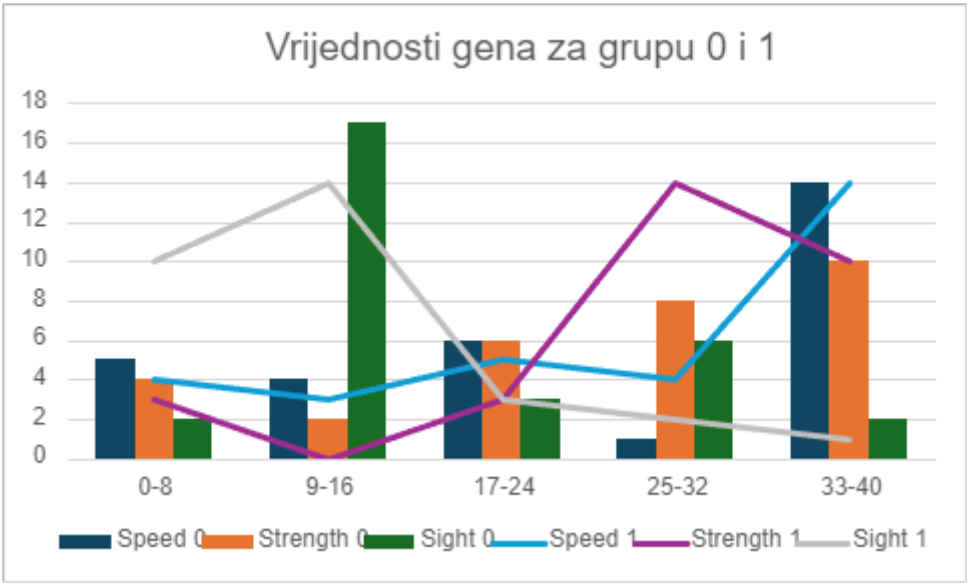
Idući pristup problemu se našao u obliku rangiranog prostora (engl. *Ranked space*). Umjesto da se za odabir koristi samo *fitness* vrijednost, može se također koristiti faktor raznolikosti gena. U ovoj implementaciji se za izračun ranga jedinke koristi se dana formula:

$$y = \frac{x_f}{x_{fm}} \cdot m_f + \sum_{i=1}^{n} \left(\frac{x_{vi}}{x_{vmi}} \cdot m_d \right) \tag{4-3}$$

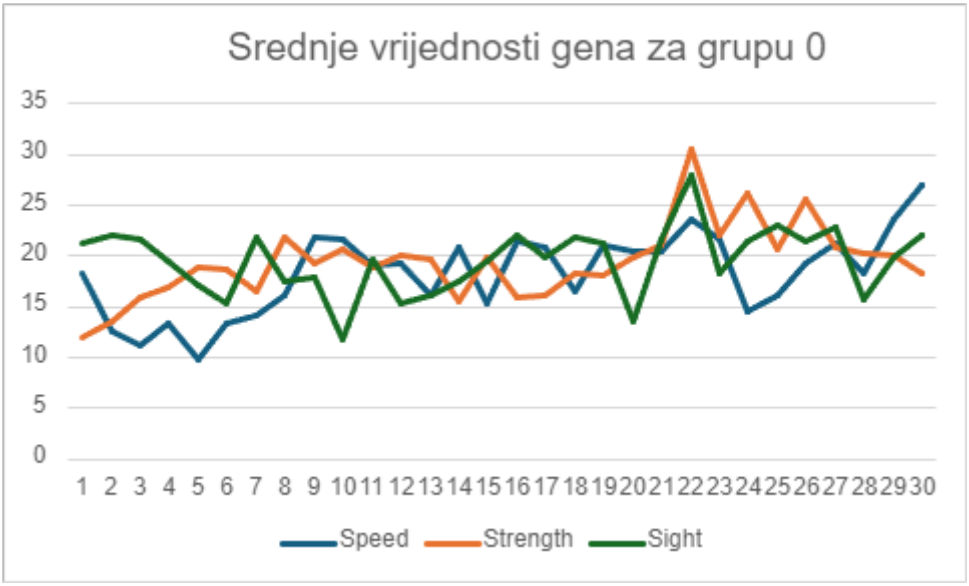
gdje je *y* - rang jedinke, x_f *fitness* vrijednost jedinke, x_{fm} maksimalna *fitness* vrijednost unutar grupe, m_f faktor množenja za *fitness* vrijednosti postavljen na 100 te se za *n* gena računa vrijednost, x_{vi} varijanca gena, x_{vmi} maksimalna varijanca gena a m_d je faktor množenja za gene koji je postavljen na 200/3.

Za formulu su korištena samo prva 3 gena, odnosno svi osim gena odabira resursnog čvora te prema (4-3) maksimalan rang koji jedinka može imati je 300 i smatra se da je omjer utjecaja *fitness-a* naspram varijance gena 1:2. Faktori se mogu promijeniti kako bi se dalo veći ili manji značaj *fitness* vrijednosti pomoću faktora *fitnessa* za rang (engl. *Rank Fitness Multiplier*) odnosno varijanci pomoću faktora raznolikosti za rang (engl. *Rank Diversity Multiplier*), ali za potrebe problema dana formula daje zadovoljavajuće rezultate. Prema slici 4.35. vrijedi da se vrijednosti gena najboljih jedinki nalaze po cijelom rasponu vrijednosti uz manje pristranosti određenim vrijednostima. Srednja vrijednost se također zadržava na određenom rasponu vrijednosti iako se vidno pomiče s vremenom (Sl. 4.36.). Varijanca dosta slična u usporedbi sa

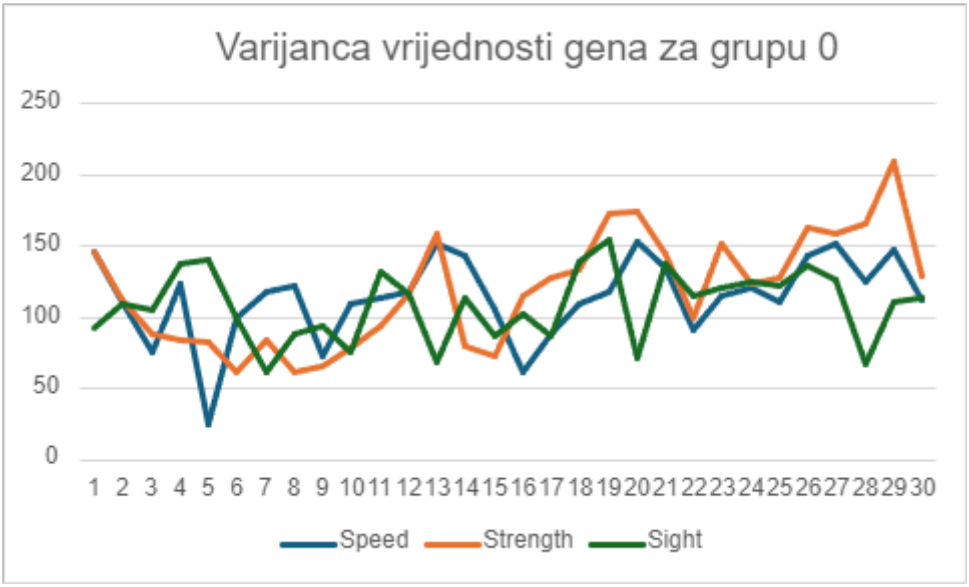
slikom 4.31., ali u ovom slučaju iako varijanca rijetko dostiže visoke vrijednosti, na temelju slike 4.35. može se zaključiti da se ipak održava raznolikost gena (Sl. 4.37.). Napravljena je dodatna simulacija gdje je omjer utjecaja 1:3 odnosno faktor raznolikosti za rang je postavljen na 100 te se može primijetiti kako je varijanca porasla što nam govori da se raznolikost gena ovim pristupom može veoma lako konfigurirati (Sl. 4.38.).



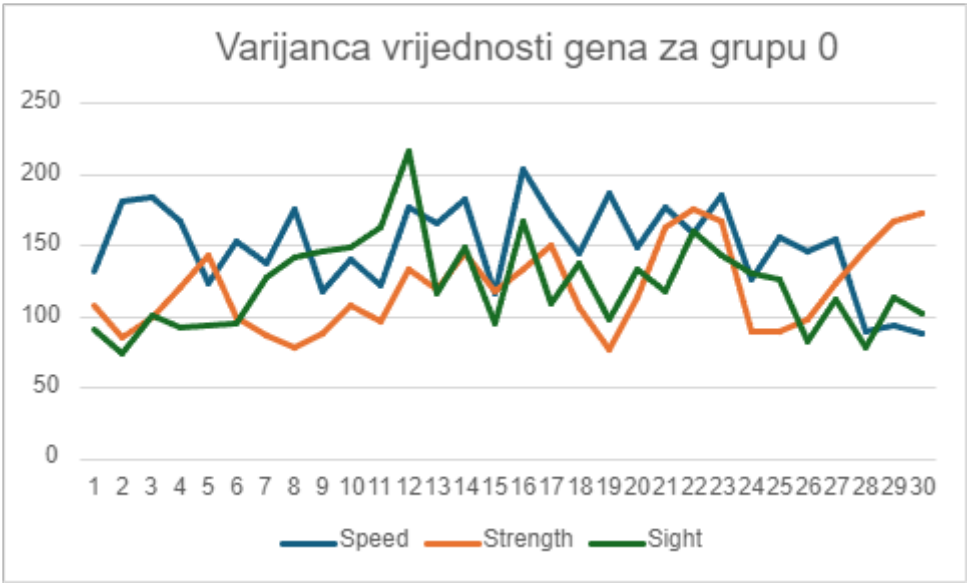
Slika 4.35. *Vrijednosti gena najboljih jedinki pri rangiranju 1:2.*



Slika 4.36. *Srednje vrijednosti gena pri rangiranju 1:2.*



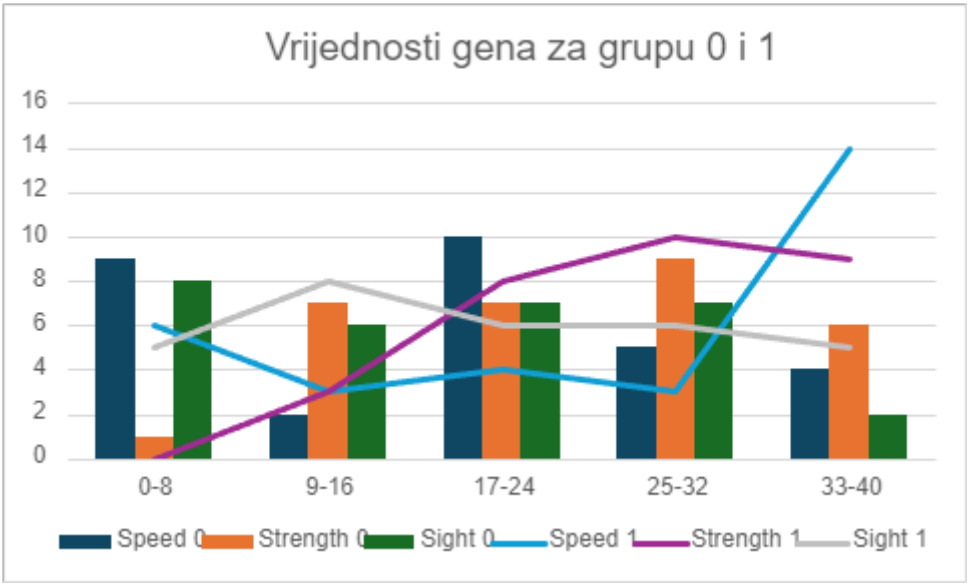
Slika 4.37. *Varijanca gena pri rangiranju 1:2.*



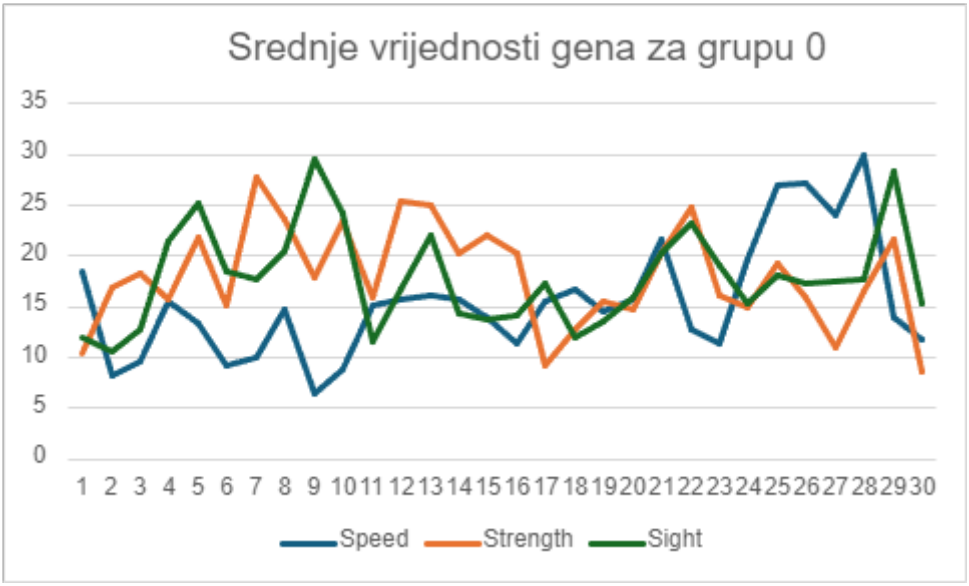
Slika 4.38. *Varijanca gena pri rangiranju 1:3.*

Za zadnji način pristupa kontroli raznolikosti gena odabran je FUSS (*Fitness Uniform Selection Scheme*) algoritam [11]. Funkcionira na način da se područje od najmanje *fitness* vrijednosti do najveće podijeli na jednake dijelove. Zatim se za svaku podijeljenu vrijednost gleda koja jedinka sadrži *fitness* vrijednost najbližu toj vrijednosti te se nju odabire za proces križanja. Nedostatak takvog pristupa je što u slučaju grupiranih jedinki u smislu *fitness* vrijednosti algoritam će stalno odabirati iste jedinke što se desilo i u ovom slučaju. Iako se vrijednosti najboljih jedinki mogu naći po cijelom rasponu vrijednosti gena (Sl. 4.39.), ta značajka u ovom slučaju dovodi do otežane kontrole srednje vrijednosti i varijance zbog grupiranog rasporeda *fitness* vrijednosti. (Sl. 4.40. i 4.41.) Na slici 4.42. je prikazan broj preživjelih jedinki u jednoj grupi po *fitness* vrijednosti na kraju trajanja jedne generacije što predstavlja upravo prethodno opisan slučaj

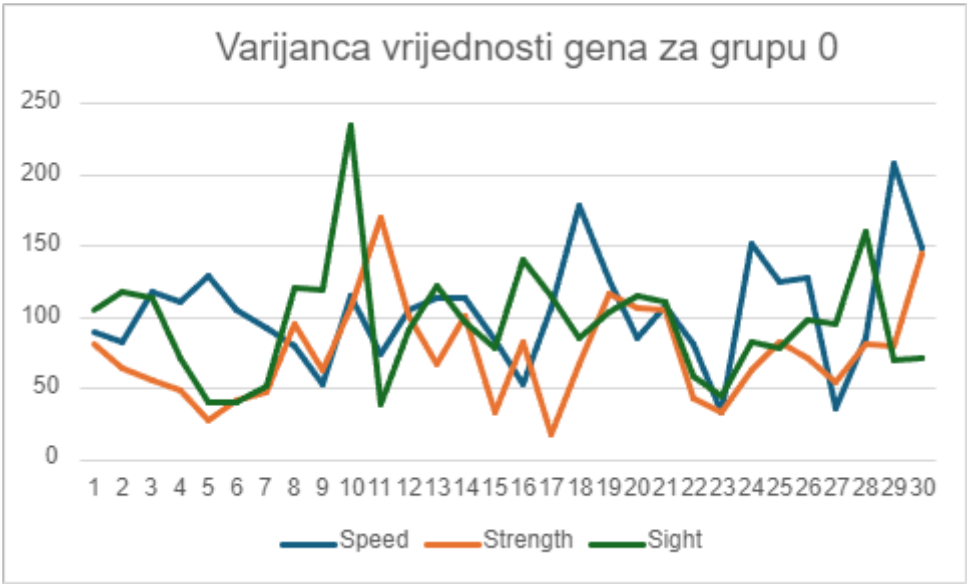
Algoritam bi se potencijalno mogao unaprijediti tako što bi u slučaju istih ili dosta sličnih *fitness* vrijednost mogao spriječiti odabir istih jedinki ako druge odgovaraju kriteriju.



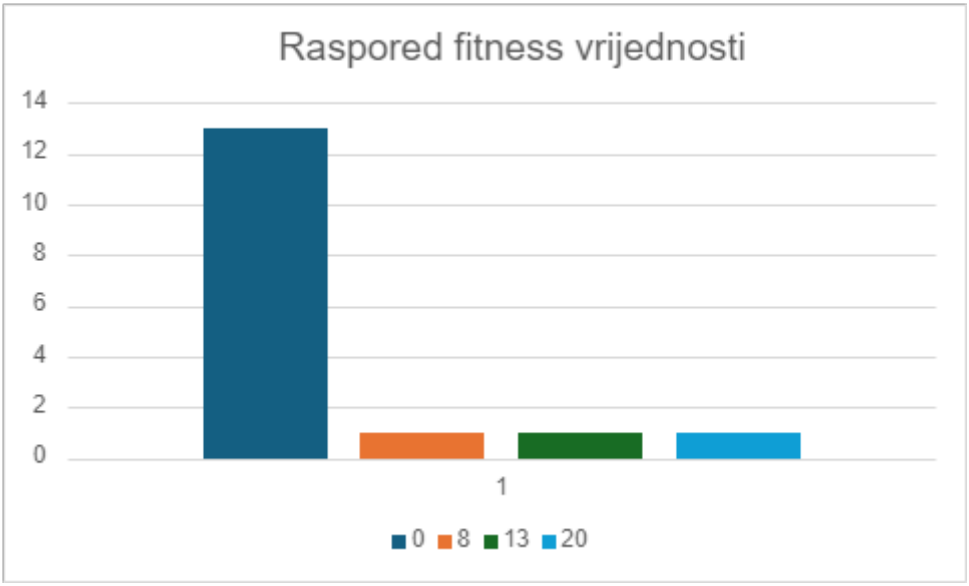
Slika 4.39. *Vrijednosti gena najboljih jedinki za FUSS.*



Slika 4.40. *Srednje vrijednosti gena za FUSS.*



Slika 4.41. *Varijanca gena za FUSS.*



Slika 4.42. *Raspored fitness vrijednosti po jedinkama za FUSS.*

Za kraj je sastavljena tablica koja prikazuje utjecaj na pomak područja pretraga odnosno srednju vrijednost i širinu područja pretrage odnosno varijancu gena. Kao osnova usporedbe je uzet pristup s nasumičnim odabirom jer se radi o izvornom ponašanju genetskog algoritma dvojne populacije. Za srednju vrijednost, vrijednost manja od 1 predstavlja pomicanje područja pretrage prema nižim vrijednostima gena. Za varijancu, vrijednost manja od 1 predstavlja sužavanje područja pretrage te manju raznolikost gena. Vrijednosti u tablici su dobivene tako što je uzet omjer srednje vrijednosti srednjih vrijednosti odnosno varijanci gena za odabrani pristup i srednje vrijednosti srednjih vrijednosti odnosno varijanci gena za pristup nasumičnog odabira odnosno izvornog ponašanja DPGA-a. Dakle tablica 4.1. upućuje na to da je uklanjanje sličnih

gena najbolji pristup za očuvanje raznolikosti gena prilikom pretrage. Suprotno tome, odabir s prednošću boljima s pomakom od –0.3 dosta sužava područje pretrage i omogućuje kontroliranije izmjene boljih jedinki kako bi se probalo naći još bolje rješenje.

	Srednja vrijednost	Varijanca
Odabir s prednošću boljima	1.073	1.166
Faktor pristranosti križanja -0.3	1.276	0.791
Nasumičan odabir	1	1
2 elitne jedinke	1.068	1.099
5 elitnih jedinki	1.13	0.95
Uklanjanje sličnih gena	1.27	1.347
Rangirani prostor 1:2	0.999	1.052
Rangirani prostor 1:3	1.048	1.149
FUSS	0.805	0.943

Tablica 4.1. *Usporedba utjecaja različitih pristupa na raznolikost gena.*

5. ZAKLJUČAK

Na temelju ovog rada može se zaključiti da je uporabom genetskog algoritma vrlo lako pronaći najbolje rješenje problema ako su njegova pravila i ciljevi kriterija jasno definirani. Iako je ovaj problem mogao biti riješen u bilo kojem okviru, Unity se pokazao kao kvalitetan alat u kojemu je moguće vizualizirati kompleksnije probleme a ujedno i izmjenjivati parametre problema tijekom izvođenja. Samom vizualizacijom lako je moguće dobiti bolji uvid u tijek izvođenja programa i ranije uočiti greške ili propuste koji se događaju. Postoji nekoliko različitih verzija genetskog algoritma što potvrđuje da postoje mnogi parametri i značajke koje se mogu uvesti i izmijeniti kako bi se postiglo optimalno ponašanje algoritma za određene vrste problema. Ni ovaj problem nije iznimka, uspješno je implementiran genetski algoritam dvojne populacije na način da se zadrži mogućnost evolucijskog napretka jedinki zajedno s kontrolom područja pretrage vrijednosti gena ovisno o odabranim opcijama unutar Unity sučelja. Kao najznačajnije opcije su se pokazali pristup s uklanjanjem sličnih gena za povećanje raznolikosti gena i pristup koji daje prednost za selekciju boljim jedinkama kako bi se pokušalo naći bolje rješenje oko područja trenutnog najboljeg rješenja. Kako izmjene ne spadaju niti pod jednu od postojećih varijanti genetskog algoritma a ni problem nije opće poznat u svijetu znanosti, dakle ne postoji zabilježen slučaj optimiziranja istog problema, teško je reći koliko je uspješno implementirana ova varijanta genetskog algoritma, ali se opisana implementacija može dodatno proširiti s karakteristikama iz drugih algoritama te se daljnje testirati koje karakteristike daju najbolje rezultate. Uporabom neuronske mreže uvedena je dodatna nasumičnost koja utječe u odabir najboljih gena što možda nije poželjno prilikom eksperimenata koji zahtijevaju stabilnost ali u ovom slučaju je pridonosilo visokom rasponu područja pretrage vrijednosti gena. Također zbog velike količine varijabli koje mogu utjecati na rezultat simulacije, najbolje vrijednosti gena koje su dobivene u jednoj generaciji se mogu daleko razlikovati od onih dobivenih u drugoj generaciji. Međutim temelji algoritama koji su korišteni u ovom radu se mogu koristiti pri razvoju modela umjetne inteligencije za jedinku koja bi bila konkurenta igračima u puno kompleksnijem virtualnom svijetu čija su pravila ponašanja detaljno definirana. S obzirom na mogućnost istraživanja različitih pristupa problemu zbog genetskog algoritma, jedinka bi mogla otkriti potpuno neočekivana rješenja koja bi davala puno bolje rezultate.

LITERATURA

[1] V. Kumar, S. M. Yadar, A state-of-the-Art review of heuristic and metaheuristic optimization techniques for the management of water resources, IWA Publishing, vol. 22, No. 4, travanj 2022

[2] S. Che, Y. Chen, L. Wang, C. Xu, Electric Vehicle Ordered Charging Planning Based on Improved Dual-Population Genetic Moth–Flame Optimization, MDPI, vol. 17, No. 3, ožujak 2024

[3] Z. Chen, X. Xu, H. Liu, Improved Dual-Population Genetic Algorithm: A Straightforward Optimizer Applied to Engineering Optimization, MDPI, vol. 15, No. 20, listopad 2023

[4] Max K.-Y. Shao; H.-Y. Zhang, The application of complex dual population genetic algorithms to optimizing the fuzzy control, IEEE, kolovoz 2011

[5] Y. Pan, Research on the Application of Two-Population Genetic Algorithm in Production Line Balancing, CSP, vol. 7, No. 2, srpanj 2023

[6] A. Alajmi Selecting the most efficient genetic algorithm sets in solving unconstrained building optimization problem, International Journal of Sustainable Built Environment, vol. 3, No. 1, srpanj 2014

[7] D.F. Cook, Combining a neural network with a genetic algorithm for process parameter optimization, Engineering Applications of Artificial Intelligence, vol. 13, No. 4, lipanj 2000

[8] smjonas(2018), Genetic Algorithm to solve the TSP in Unity, dostupno na: https://github.com/smjonas/Genetic-Algorithm-TSP-Unity

[9] M. Gestal, D. Rivero, Two-population genetic algorithm, SciTePress, 2010

[10] J. Heaton, Introduction to neural networks with Java, 2005

[11] D. Gupta, S. Ghafir, An Overview of methods maintaining Diversity in Genetic Algorithms, International Journal of Emerging Technology and Advanced Engineering, vol. 2, No. 5, svibanj 2012

SAŽETAK

Cilj ovog rada je bio analizirati različite pristupe koji se mogu primijeniti u konfiguraciji genetskih algoritama kako bi se postigla kontrola nad raznolikošću gena prilikom potrage za najboljim rješenjem. Za potrebe ovog rada, u Unityu je stvoren virtualni svijet koji podržava dvije populacije i omogućuje im natjecanje za ograničene resurse. U postavkama za simulirani svijet su ponuđene opcije kojima se lako mogu izmijeniti mnogi parametri izvođenja simulacije. Također su dane upute za izmjenu virtualnog svijeta u slučaju potrebe za testiranje različitih scenarija. Testirani su pristupi poput elitizma, uklanjanja sličnih gena, FUSS algoritma i mnogi od njih su pokazali pozitivne rezultate u smislu izbjegavanja preuranjene konvergencije zbog slabe raznolikosti gena. Neuronska mreža je također imala manji utjecaj na raznolikost, a i pokazala se prigodna za donošenje odluka na temelju cijele populacije ako jedinke imaju jasno definirana pravila ponašanja.

Ključne riječi: genetski algoritam, neuronska mreža, parametri simulacije, raznolikost gena

ADAPTIVE DIVERSITY CONTROL WITH DUAL POPULATION GENETIC ALGORITHM

ABSTRACT

The aim of this work was to analyze different approaches that can be applied in the configuration of genetic algorithms in order to achieve control over the diversity of genes when searching for the best solution. For the purposes of this work, a virtual world was created in Unity that supports two populations and allows them to compete for limited resources. In the settings for the simulated world, options are provided that can easily change many parameters of the simulation run. Instructions are also given for modifying the virtual world in case of need to test different scenarios. Approaches such as elitism, removal of similar genes, FUSS algorithm have been tested and many of them have shown positive results in terms of avoiding premature convergence due to poor gene diversity. The neural network also had less impact on diversity, and proved to be suitable for making decisions based on the entire population if the individuals have clearly defined rules of behavior.

Keywords: genetic algorithm, neural network, simulation parameters, gene diversity