

SVEUČILIŠTE U ZAGREBU  
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

**SEMINAR**

**Naslov seminarskog rada**

*Mario Kostelac*

Voditelj: *Domagoj Jakobović*

Zagreb, travanj, 2012.



## Sadržaj

<b>1. SAŽETAK</b> .....	<b>1</b>
<b>2. UVOD</b> .....	<b>2</b>
<b>3. GENETSKI ALGORITMI KAO LOGIČKI SKOK U RAČUNARSTVU</b> .....	<b>3</b>
3.1. IDEJA.....	3
3.2. POČETNA POPULACIJA.....	4
3.3. UMJETNA SELEKCIJA.....	4
3.4. KRIŽANJE.....	5
3.5. MUTACIJA.....	5
3.6. UVJET ZAUSTAVLJANJA.....	5
<b>4. IMPLEMENTACIJA. MODELIRANJE JEDNOSTAVNOG VOZAČA</b> .....	<b>6</b>
4.1. PATHBUILDER.....	6
4.1. SUSTAV ZA GENETIKU.....	7
4.2. SUSTAV ZA GENETIKU.....	7
4.3. MODEL VOZAČA.....	7
4.3.1. ULAZI.....	7
4.3.2. ODLUČIVANJE O SLJEDEĆOJ AKCIJI.....	8
4.3.3. MOGUĆE AKCIJE.....	9
4.3.4. ZAPIS GENOTIPA.....	9
4.4. POČETNA POPULACIJA.....	9
4.5. FUNKCIJA DOBROTE.....	9
4.6. SELEKCIJA.....	10
4.7. KRIŽANJE.....	11
4.8. MUTACIJA.....	11
4.9. PODESIVI PARAMETRI.....	12
<b>5. ANALIZA USPJEŠNOSTI</b> .....	<b>13</b>
5.1. STAZA 1.....	13
5.2. STAZA 2.....	14
5.3. STAZA 3.....	14
5.4. STAZA 4.....	17
<b>6. ZAKLJUČAK</b> .....	<b>20</b>
<b>7. LITERATURA</b> .....	<b>21</b>

## **1. Sažetak**

Evolucijsko računarstvo je novo područje računarstva temeljeno na osnovnim principima prirodne evolucije. Nastalo je 60-tih godina prošlog stoljeća, a danas se koristi kao robustan alat za rješavanje čitave familije problema za koje ne znamo pronaći učinkovite algoritme i optimalna rješenja. U seminaru je objašnjena osnovna ideja i značajke genetskih algoritama, dijela evolucijskog računarstva, kojim se u kasnijem dijelu modelira jednostavan model vozača, kojemu je u cilju naučiti savladati plošne staze. Analiza nekoliko izgrađenih staza pokazuje kako je jednostavni model, izgrađen kao konačni automat, sposoban savladati srednje komplicirane staze, ali su pokazana i ograničenja pri učenju tako izgrađenog modela.

## 2. Uvod

Igraća industrija je zasigurno jedna od vodećih industrija u svijetu, a predviđanja stručnjaka koji promatraju tržišna kretanja konvergiraju misli da se ona samo penje na ljestvici najprofitabilnijih (Caron, 1998). Crysis, Need For Speed, PES, FIFA su zasigurno svima poznati naslovi, no bi li oni bili toliko dobri da su protivnici nedorasli, da trče u krivom smjeru ili iz nekog nepoznatog razloga skaču u vodu (iako i to ponekad ima smisla), umjesto da pucaju u vas?

Umjetna inteligencija je važna i postaje sve važniji dio današnjih igara. Pitanje koje se nameće je kako programirati “mentalne sklopove” ostalih objekata u igrama? Umjetna inteligencija, kao čitava grana računarske znanosti, nudi mnoge pristupe izrade takvih modela, a jedan od njih je zasigurno i uporaba genetskih algoritama, jednostavnog koncepta prvi puta uporabljenog ranih 60-tih (Mitchell, 2009.).

Tema ovog seminara je modeliranje pojednostavljenog vozača koji upravlja vozilom u plošnom svijetu konstruiranim u obliku jednostavne staze. Generalizirani problem se naziva “Evolucija praćenja staze” (eng. Evolution of Corridor Following Behavior).

U sljedećem se poglavlju opisuju osnovni pojmovi vezani za genetske algoritme, a nakon toga se kreće u modeliranje jednostavnog vozača koji bi trebao biti u mogućnosti snaći se na stazi.

### 3. Genetski algoritmi kao logički skok u računarstvu

“Može li računalo reproducirati samo sebe?” pitanje je na koje će mnogi odgovarati negativno. Do zaključka da to nije sasvim točan odgovor, došao je von Neumann 60-tih godina 20. stoljeća. Skupina znanstvenika i entuzijasta toga doba počela je razmatrati koncept programiranja pomoću kojeg će program znati prepoznati okolinu te se ponašati u skladu s njom, ali isto tako i odrediti što bi bilo dobro učiniti u nekoj novoj situaciji (Mitchell, 2009.).

Promatranjem mehanizama evolucije, križanja, mutacija i prirodne selekcije stvoreno je novo područje računarske znanosti, spremno odgovoriti na zadani problem - evolucijsko računarstvo Genetski algoritmi, kojima se rješavaju problemi ovog seminara, samo su dio evolucijskog računarstva.

U nastavku seminara se objašnjava se osnovni koncept genetskih algoritama te pojmovi koji su neizostavni za razumijevanje daljnjeg dijela seminara.

#### 3.1. Ideja

Ideja genetskih algoritama je provođenje umjetne evolucije na našem računalu u svrhu optimiziranja određenih problema ili generiranja rješenja za probleme na koje čovjek teško daje optimalan odgovor.

U početku generiramo *početnu populaciju*, sastavljenu od više i manje dobrih jedinki koje su u određenoj mjeri spremne preživjeti situacije koje ju okružuju.

Jedinke opisujemo *kromosomima*, nakupinom svojstava jedinke populacije, koji u potpunosti opisuju ponašanje jedinke u određenoj situaciji.

Najbolje jedinke (*funkcija dobrote*) se snalaze i preživljavaju (*prirodna selekcija*), preživljene se *križaju*, a tijekom križanja može doći do određene *mutacije*. Cijeli proces generiranja se odvija dok ne dobijemo zadovoljavajuće rješenje ili dok ne prođe zadano vrijeme/broj koraka (*uvjet zaustavljanja*) (Mitchell, 2009.).

### 3.2. Početna populacija

Početna populacija je slučajno generirana populacija jedinki koja kreće u rješavanje zadanog problema. Veličina populacije (najčešće je fiksna tijekom generacija) ovisi o prirodi problema i bitan je faktor u brzini konvergencije optimalnom rješenju. Manji broj jedinki u populaciji smanjuje raznolikost početnih rješenja, što osigurava bržu konvergenciju, ali ne osigurava nužno dobar smjer. Manje populacije mogu vrlo brzo konvergirati, ali generirati rješenja čija je korelacija s optimumom vrlo mala.

Povećanje broja jedinki osigurava veću raznolikost i mogućnost dobre konvergencije, ali isto tako povećava vrijeme izvođenja algoritma. Optimum ovog parametra se dobiva eksperimentalno i svojstven je prirodi problema i kombinaciji ostalih parametara algoritma.

Ako znamo u kojem se prostoru nalazi optimum, početna populacija se može generirati tako da ne bude sasvim slučajna, već da zadovoljava egzistenciju u okolišu predviđenog prostora rješenja

### 3.3. Umjetna selekcija

Tijekom života neke generacije, određene jedinke moraju umrijeti. Kako želimo da prežive dobre jedinke (elitizam), svakoj jedinci moramo pridružiti neki broj.

Pridruživanje brojčane vrijednosti jedinci odražuje *funkcija dobrote* – važan faktor uspješnosti pronalaženja rješenja. Eksperimentima je pokazano da guranje samo najboljih jedinki u sljedeću generaciju ne daje nužno optimalan rezultat pa se uvode *operatori selekcije*- funkcije/algoritmi odabira jedinki za preživljavanje.

Najčešće koristimo jedan od tri navedena operatora selekcije: generacijski jednostavni odabir (eng. roulette-wheel selection), turnirski odabir (tournament selection), elistički odabir.

*Generacijski jednostavni odabir (roulette-wheel selection)* slaže dobrote jedinki na pravac tako da najveća dobrota zauzima najveći dio pravca. Odabir se vrši generiranjem slučajnog broja od 0 do kraja duljine. Treba primijetiti da najveća dobrota ima najveću vjerojatnost preživljavanja, ali isto tako da su nam za ovaj operator prihvatljive samo pozitivne vrijednosti dobrote.

*Turnirski odabir* se obavlja tako da se uzme  $n$  slučajno odabranih jedinki iz populacije, a u sljedeću generaciju kopira najboljih  $m$  ( $m < n$ ).

*Elitistički odabir* se vrši tako da se kopira  $n$  najboljih u sljedeću generaciju.

Nakon generiranja baze sljedeće generacije, kreće se u križanje preživjelih (Golub, 1997.)

### **3.4. Križanje**

Križanje je binarni operator. Kao parametre prima dvije jedinke (roditelje), a rezultat je novonstala jedinka (dijete) slična svojim roditeljima. Križanje se može dogoditi u jednoj ili više točaka prekida, a tijekom križanja je moguće pojavljivanje *mutacije*.

### **3.5. Mutacija**

Mutacija je unarni operator koji se primjenjuje pri procesu križanja s određenom vjerojatnošću  $p_m$ . Prilikom mutacije se slučajno odabrani gen zamjenjuje nekim novim. Iako je mutacija najčešće štetna, postoje slučajevi kada ona daje poboljšanja konačnog rješenja. Priroda operatora mutacije usporava konvergenciju.

### **3.6. Uvjet zaustavljanja**

Uvjet zaustavljanja je izraz koji određuje okončanje procesa evolucije. Može biti definiran postignutom dobrotom rješenja, brojem generacija, vremenski ograničenjem ili brojem iteracija bez značajnog poboljšanja.

Sada kada su objašnjeni svi potrebni parametri jednostavnog genetskog algoritma, pseudokod pojednostavljenog algoritma bi trebao biti nametnut sam od sebe.

```
Generiraj početnu populaciju
```

```
Izračunaj dobrotu svake jedinke u populaciji
```

```
Ponavljaj dok je uvjet zaustavljanja laž
```

```
    Odaberi najbolje jedinke operatorom selekcije
```

```
    Generiraj nove jedinke operatorima križanja i mutacije
```



## 4. Implementacija. Modeliranje jednostavnog vozača

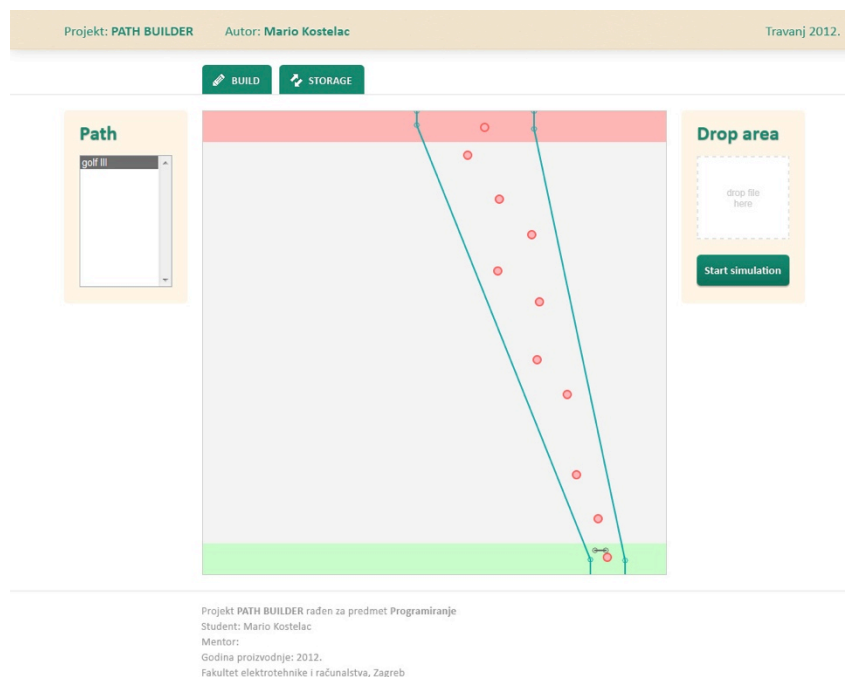
Cilj programskog dijela seminara je izgradnja sustava za modeliranje i simulaciju vožnje jednostavnog vozača. Kako model ne može voziti po nedefiniranoj stazi, potrebno je izgraditi i dio sustava za izgradnju staze – *PathBuilder*.

### 4.1. PathBuilder

PathBuilder je dio sustava za izgradnju staze i simulaciju već izračunate vožnje. Zbog poznavanja web tehnologija, njihove jednostavnosti i dolaska HTML5 (a time i canvas elementa), ovaj dio sustava je implementiran u web pregledniku (tehnologije: HTML + CSS + Javascript).

Izgradnja staza se obavlja definiranjem rubova ceste (s obje strane), kontrolnih točaka (eng. checkpoints) i početne linije putem jednostavno sučelja.

Simulacija se obavlja dovlačenjem generirane datoteke u prozor preglednika i pritiskom na tipku start.



Slika 1 Sučelje PathBuildera

## **4.1. Sustav za genetiku**

*Sustav za genetiku je zbog zahtijevnosti operacija implementiran u programskom jeziku Java.*

Sustav za genetiku obavlja najvažniji dio posla i programiran je generički, što povlači dobru posljedicu da je, uz male modifikacije, u stanju pružiti okolinu za različite modele i implemetacije vozača.

Kako je sustav u vrlo ranoj fazi razvoja, on neće biti analiziran, ali će se moći naslutiti što pruža.

## **4.2. Sustav za genetiku**

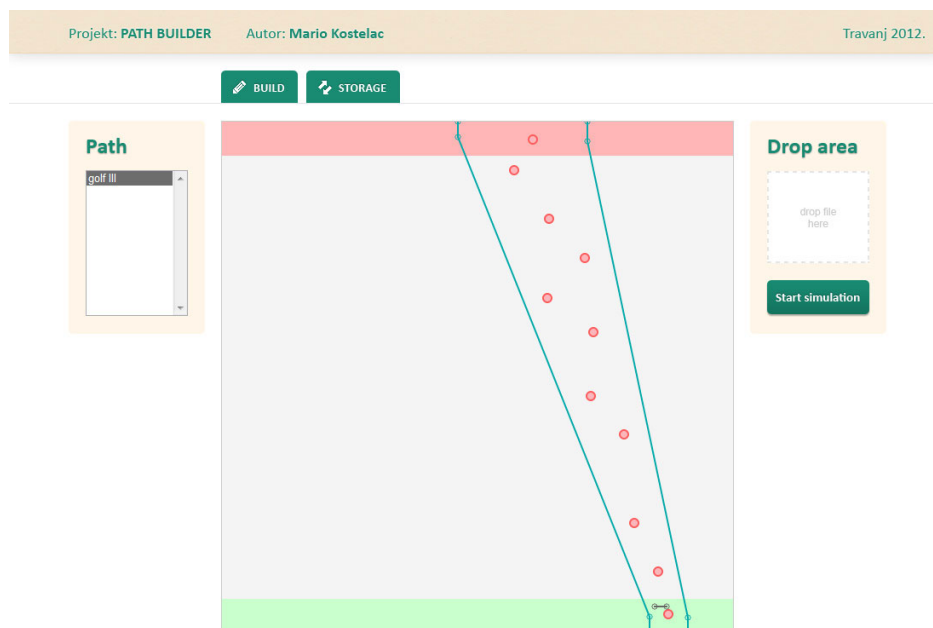
Komunikacija između PathBuildera i ovog sustava ostvarena je pomoću tekstualnih datoteka u kojima se ključnim riječima i koordinatama određenih točaka prenose sve informacije o stazama i izračunatim vožnjama. Kako bi live izvođenje genetskog algoritma u pregledniku zahtijevalo ponovno pisanje istog modela u JavaScriptu i stvorilo priličnu redundanciju, PathBuilder nakon procesa računanja dobiva gotova stanja koja treba animirati na prikladan način.

## **4.3. Model vozača**

Evolucijsko računarstvo nas ni u kojem pogledu ne ograničava u pogledu implementacije odlučivanja o sljedećoj akciji modela. Zbog jednostavnosti izvedbe, prikazani model je implementiran kao Mealyev automat – konačni automat koji odluku o sljedećoj akciji donosi na temelju trenutnog stanja i novopridošlih ulaza.

### **4.3.1. Ulazi**

Ulazi automata su numeričke vrijednosti koje vraćaju senzori – veza vozila i vanjskog svijeta. Analizirani model posjeduje samo jednu vrstu senzora – «lasere» koji mjere udaljenost između vozila i najbliže prepreke. Kako broj stanja automata (objašnjeno kasnije) raste eksponencijalno sa brojem senzora, a pokazano je da je manji broj senzora dovoljan za pravilno ponašanje u okolišu (Reynolds, 2000.), model posjeduje 3 fiksirana senzora raspoređena kao na slici. Senzori su idealizirani i sposobni su trenutačno izračunati udaljenosti.



Slika 2 Raspored senzora

### 4.3.2. Odlučivanje o sljedećoj akciji

Kako je cijeli model vrlo jednostavan, tako ni odlučivanje o sljedećoj akciji nije naročito komplicirano. Pomoću trenutnih vrijednosti koje vraćaju senzori i vrijednosti trenutka prije (vrijeme diskretizirano), model računa promjene udaljenosti na svakom od senzora. Dobivene razlike (derivacije) linearno skalira tako da najveća promjena poprima vrijednost 8 (nazovimo je nekom vrstom razlučivosti) i na temelju tih novoizračunanih vrijednosti odlučuje što će sljedeće učiniti.

Pseudokod:

```

ocitajSenzore
promjene = sadasnjeVrijednosti - prosleVrijednosti
zapamtiSadanjeVrijednosti
sljedecaAkcija = dohvatiAkciju(promjene)

```

Ovakav način odlučivanja spada u skupinu greedy rješenja jer na odluku modela u svakom trenutku najviše utječe prepreka prema kojoj se najbrže približava.

Kako je u stvarnosti automobil u nemogućnosti prečestih promjena smjera, tako je i ovaj model ograničen tako da svoju akciju možete donositi svakih  $n$  trenutaka.

### 4.3.3. Moguće akcije

Model je sposoban izvesti samo tri akcije, a one sve djeluju na nastavak simulacije tako da modificiraju vektor brzine.

- nastavi: akcija ne mijenja vektor brzine
- skreni lijevo: akcija zakreće vektor brzine za kut  $+\pi/8$
- skreni desno: akcija zakreće vektor brzine za kut  $-\pi/8$

### 4.3.4. Zapis genotipa

Zapis genotipa ostvaren je pomoću trodimenzionalnog polja koje svakoj mogućoj kombinaciji promjena udaljenosti prepreka dodjeljuje jednu od definiranih akcija. Kako najveća promjena dobiva vrijednost 8 (a moguće je i da bude negativna), imamo  $2*8+1$  različitih udaljenosti koje moramo pokriti. Trodimenzionalnost je posljedica broja senzora (3 senzora), što inducira da je veličan genotipa i broj kombinacija koje pokrivamo eksponencijalno ovisan o broju senzora. Za opisani model broj različitih kombinacija je  $3^{17*17*17}$ , što približno iznosi  $1.25*10^{2344}$ . Ovako velik predstavlja ogroman izazov za pravilnu konvergenciju, ali pokazat će se da je mali broj kombinacija ključan u pravilnom voženju staze.

## 4.4. Početna populacija

Kako bi dobili pitanje na odgovor «koliko je ovakav jednostavan model samostalno sposoban naučiti voziti?», početna populacija stvara se sasvim nasumično. Zbog velikog broja mogućih kombinacija veličina cijele populacije je velika – sadrži 1000 jedinki. Nakon što generiramo početnu populaciju, potrebno je ocijeniti njene jedinke *funkcijom dobrote*.

## 4.5. Funkcija dobrote

Funkcija dobrote za dodjelu konačne ocjene uzima nekoliko parametara u obzir: je li se vozilo slupalo, je li se vratilo u krivom smjeru, koliko je kontrolnih točaka ostavilo iza sebe te koliko je napredovalo u vertikalnom smjeru. Vrlo je lako prepoznati koja ostvarenja događaju donose nagradu, a koja kaznu. Dodatno, ako se uspoređuju

dvije jedinice, koje su obje uspješno odvozile stazu, bolja je ona koja je prešla manju udaljenost.

Vertikalno napredovanje po stazi inspirirano je *air efficiency* koeficijetom (Reynolds, 2000.).

```
private int calculateFitness() {  
    int fitness = 0;  
    fitness += (this.model.passedCheckpoints()*200);  
    if (this.model.isCrashed())  
        fitness -= 1000;  
    if (this.getSimulationModel().getPosition().getY() < 1)  
        fitness -= 1000;  
    if (this.model.isFinished()) {  
        fitness += 2000;  
        fitness -= this.model.getDrivenDistance();  
    }  
    fitness += 5*(300 - new Double(this.model.getPosition().getY()).intValue());  
    return fitness;  
}
```

Slika 3 Implementacija funkcije dobrote

## 4.6. Selekcija

Selekcija jediniki vrši se primjenom dvorazinskog turnira. Četiri nasumično odabrane jedinice se sučeljavaju u dvije borbe, a pobjednici odlaze u finali. Pobjednik finala nagrađen je preživljavanjem u sljedeću generaciju. Turniri se održavaju dok god ne bude izgrađeno pola sljedeće generacije. Ovakav način selekcije osigurava rastuću (ili barem konstantu) srednju ocjenu generacije.

```

private int calculateFitness() {
    int fitness = 0;

    fitness += (this.model.passedCheckpoints())*200;

    if (this.model.isCrashed())
        fitness -= 1000;

    if (this.getSimulationModel().getPosition().getY() < 1)
        fitness -= 1000;

    if (this.model.isFinished()) {
        fitness += 2000;
        fitness -= this.model.getDrivenDistance();
    }

    fitness += 5*(300 - new Double(this.model.getPosition().getY()).intValue());

    return fitness;
}

```

Slika 4 Shema turnirskog odabira

## 4.7. Križanje

Nakon selekcije se obavlja križanje parova preživjelih jedinki i «rađanje» novih kako bi se dobila potencijalno bolja rješenja. Križanje se odvija puknućem na jednom mjestu. Nakon što se slučajnim odabirom određuje mjesto puknuća, do tog mjesta dijete nasljeđuje gene jednog roditelja, a od tog mjesta nadalje gene drugog.

													točka prekida					
1	1	0	1	0	1	1	1	0	0	0	1	0	1	1	0	1	0	RODITELJI
0	1	1	0	0	1	1	0	1	1	0	0	1	1	1	0	0	1	
1	1	0	1	0	1	1	1	0	0	0	1	0	1	1	0	0	1	
0	1	1	0	0	1	1	0	1	1	0	0	1	1	1	0	1	0	DJECA

Slika 5 Križanje s jednom točkom prekida. Preuzeto iz (Golub, 1997.)

## 4.8. Mutacija

Mutacija se u zadanom slučaju događa sa vjerojatnošću  $p_k = 10^{-6}$ , što nam omogućava jednostavan izračun da pri nekom križanju jedinka ostane nepromijenjena. Vrijednost te vjerojatnosti  $P(\{\text{jedinka ostaje nepromijenjena}\}) = (1 - p_k)^{17 \cdot 17 \cdot 17}$ . U sljedećoj tablici prikazana je vjerojatnost  $p_k$  i pripadajuća vjerojatnost  $P$ .

Tablica 1 Tablica vjerojatnosti mutacije

$p_k$	P
1.00E-03	0.73%
1.00E-04	61.18%
5.00E-05	78.22%
1.00E-05	95.21%
5.00E-06	97.57%
1.00E-06	99.51%

#### 4.9. Podesivi parametri

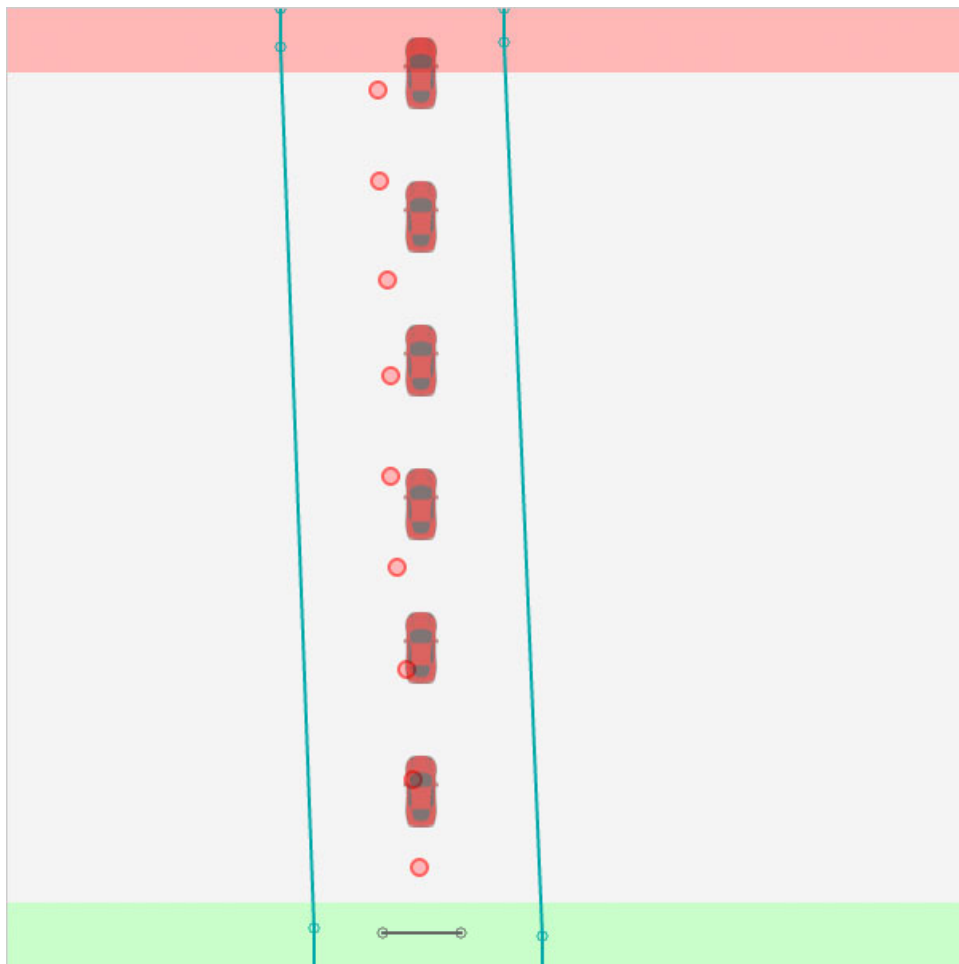
Iako je sustav u ranoj fazi razvoja, glavni parametri jednostavnog genetskog algoritma su podesivi promjenom pripadajućih konstanti. Tako je moguće mijenjati veličinu generacije, maksimalni broj generacija, vjerojatnost mutacije, broj senzora automobila, kut zakreta i kut koji automobil pokriva sensorima. Mijenjanje ponašanja automobila zahtijeva poznavanje Java. Novi modeli vozača mogu koristiti gotove senzore za udaljenost koji se ponašaju na već opisan način.

## 5. Analiza uspješnosti

U ovom poglavlju analiziraju se slučajevi učenja modela na nekoliko različitih staza uz prikaz konvergencije i izgleda staze. Parametri pri evoluiranju ponašanja vozača su:

- veličina generacije: 1000
- maksimalni broj generacija: 20
- vjerojatnost mutacije:  $10^{-6}$

### 5.1. Staza 1



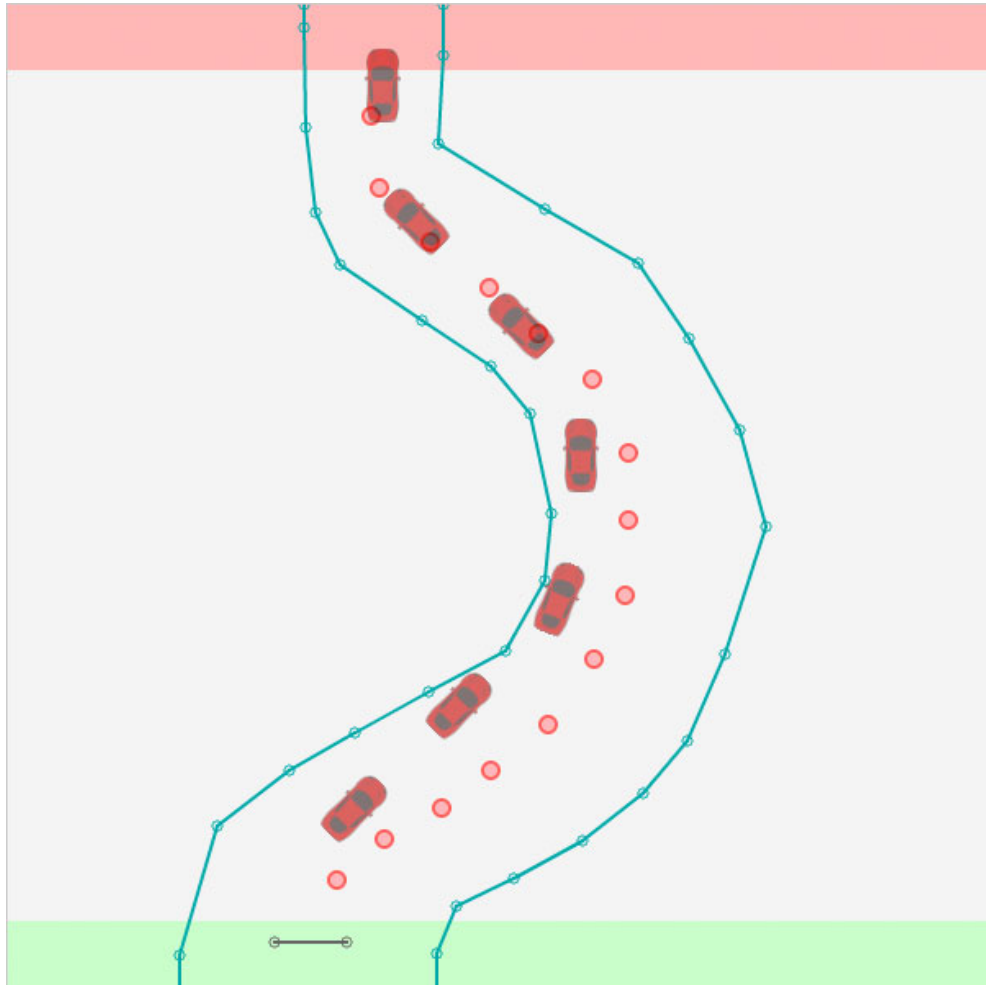
*Slika 6 Simulacija jedinke prve generacije*

Kako je vidljivo s priložene slike, prva staza nije bila nikakav izazov za opisani model vozača. Već u prvoj generaciji je pronađen model koji je odvezio kao na slici.



## 5.2. Staza 2

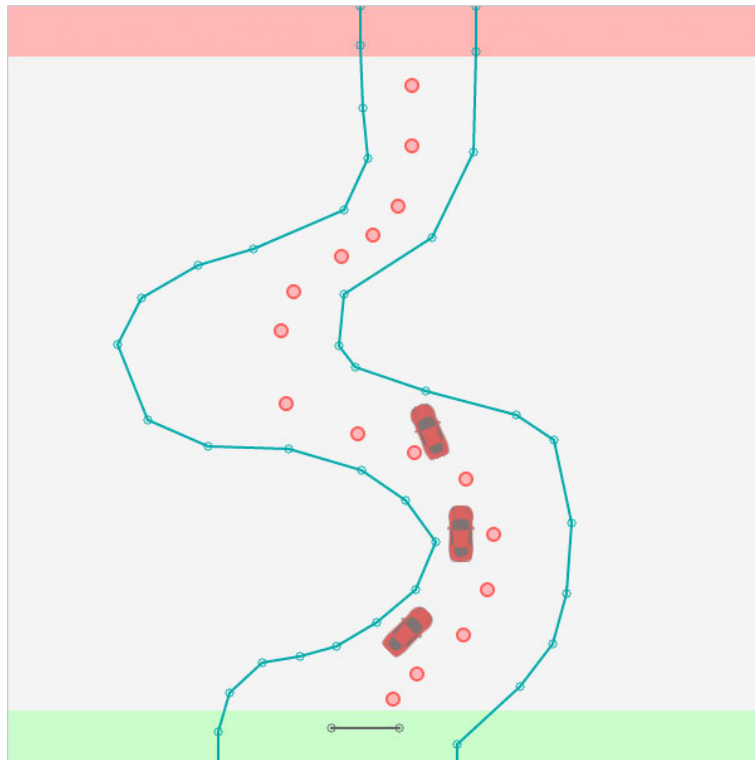
Kako potpuno ravna staza nije nikakav problem za modeliranog vozača, sljedeća je modelirana s blagim zavojem. Kao i na slici, već prva generacija je polučila jedinkom koja je sposobna stazu odvoziti do kraja. Simulacija najbolje jedinke prikazana je na slici. Graf konvergencije izostaje zbog lakog savladavanja izgrađene staze.



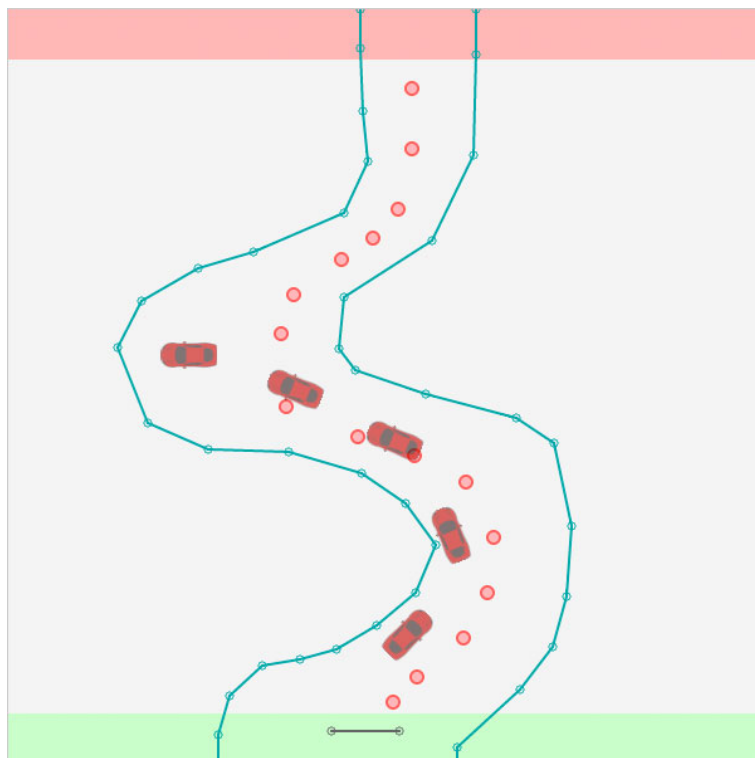
*Slika 7 Simulacija najbolje jedinke prve generacije*

## 5.3. Staza 3

Treća staza je kompliciranija nego druga. Dodan je jedan oštiji sa širokom stranom kao mamcem. Slijede slike simulacije.



*Slika 8 Simulacija najbolje jedinke prve generacije*

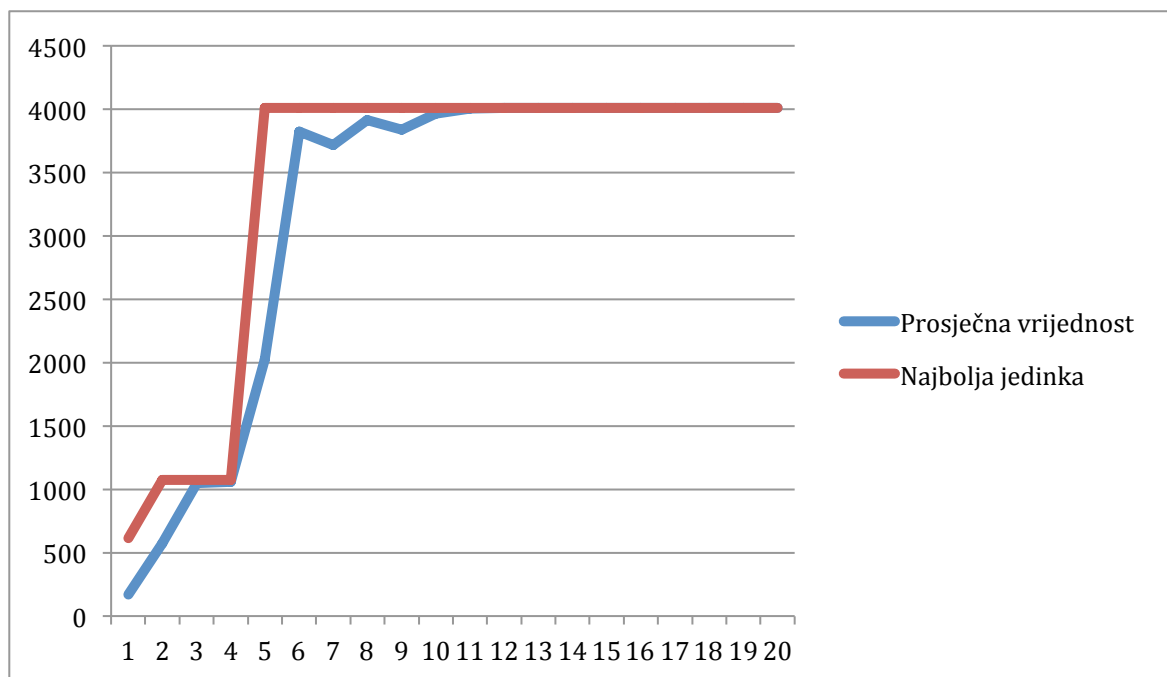


*Slika 9 Simulacija najbolje jedinke 4. generacije*



*Slika 10 Simulacija najbolje jedinke 5. generacije*

Iako je predviđeno da bi model trebao biti sposoban naučiti u 20 generacija, model je savladao stazu već u 5. generaciji. Slijedi graf konvergencije funkcije dobrote.



*Graf 1 Funkcija dobrote*

#### 5.4. Staza 4

Četvrta staza koju model treba savladati je kompliciranija od prethodne i na određenim dijelovima ima poprilična suženja, čije bi savladavanje moglo predstavljati problem.



*Slika 11 Simulacija najbolje jedinke 1. generacije*

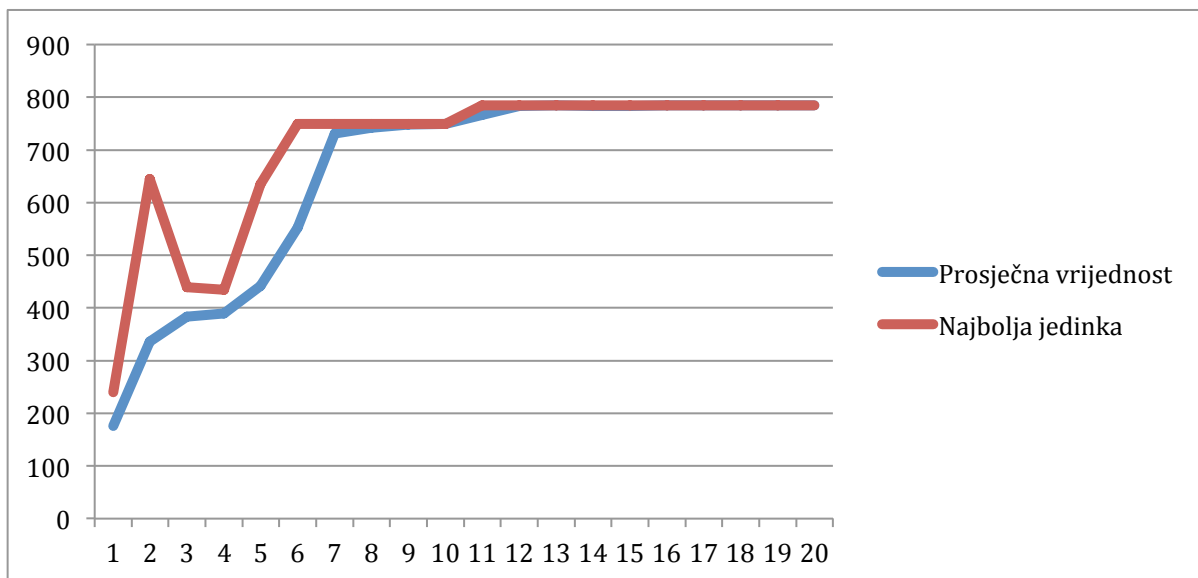


*Slika 12 Simulacija najbolje jedinke 6. generacije*



*Slika 13 Simulacija najbolje jedinke 20. generacije*

Sa slika je vidljivo da model nije uspio savladati stazu ni nakon 20 generacija. Isto tako, vidljivo je da nakon 6. generacije ne postoje značajni pomaci u uspješnosti savladavanja.



Graf 2 Funkcija dobrote

## **6. Zaključak**

Genetski algoritmi pružaju mogućnost rješavanja problema za koje ne znamo postoji li optimalno rješenje ili ne znamo kojim pristupom krenuti kako bi do njega došli. Modeliranjem opisanog vozača pokazano je kako vrlo jednostavan pristup donosi zadovoljavajuće rezultate, ali i ograničenja jednostavnog modela.

## 7. Literatura

- Mitchell, M. Complexity: A guided tour, New York: Oxford University Press, Inc., 2009
- Golub, M. Genetski algoritmi, 1997,  
[http://www.zemris.fer.hr/~golub/ga/ga\\_skripta1.pdf](http://www.zemris.fer.hr/~golub/ga/ga_skripta1.pdf)
- Reynolds, C. Evolution of Corridor Following Behavior in a Noisy World, 2000.,  
<http://www.red3d.com/cwr/papers/1994/sab94.html>
- Wikipedia, [http://en.wikipedia.org/wiki/Genetic\\_algorithm](http://en.wikipedia.org/wiki/Genetic_algorithm)
- Caron, F. Gaming expected to be a \$68 billion business by 2012., 9. Studenog 2010.
- Jakobović, D., Genetski algoritmi – predavanje, 24. travnja 2007.,  
<http://www.zemris.fer.hr/~yeti/studenti/GA%20predavanje.pdf>