UČENJE GENETSKIH ALGORITAMA POMOĆU NADGRAĐENIH PETRI-MREŽA IPROGRAMSKOG JEZIKA PYTHON

Perica Štrbac[1] Pavle Štrbac[2] Miloš Pejanović[3] Stefan Pejanović[4]

Rezime: Na strukovnom master studijskom programu Akademije tehničko-umetničkih strukovnih studija, Odseka Visoka škola elektrotehnike i računarstva na predmetu Programiranje u integrisanim tehnologijama uče se primena programskog jezika *Python* za programiranje heurističke metode optimizacije korišćenjem genetskog algoritma. U okviru jednog predavanja studenti se upoznaju sa osnovama genetskih algoritama i sa kreiranjem rešenja datog problema korišćenjem programskog jezika *Python*. Rešenje obuhvata programiranje korišćenjem osnovnog paketa (modula) jezika *Python*. Nakon toga uči se korišćenje *Python* biblioteke *GeneAl* na rešavanju drugog problema.Na vežbama studenti dobijaju novi problemski zadatak koji timski rešavaju primenom genetskog algoritma pri čemu koriste i *Python* biblioteku *matplotlib* za grafički prikaz dobijenih rešenja.U ovom radu prikazani su modeli rešenja navedenih zadataka pomoću genetskih algoritama u nadgrađenim Petri-mrežama. Nakon modelovanja, ovi modeli su konvertovani u konkretan *Python* kod. Izvršavanjem konkretnog koda proverena je korektnost modela. Studentima se kroz interaktivno modelovanje, simulaciju i analizu datih modela na dinamičan način opiše rad genetskog algoritma i način kodovanja datih rešenja u jeziku *Python*.

Ključne reči: Python, GeneAl, matplotlib, genetski algoritam, nadgrađene Petri-mreže

LEARNING GENETIC ALGORITHMS USING UPGRADED PETRI-NETS AND PYTHON PROGRAMMING LANGUAGE

Abstract: In the applied study program of the Academy of Technical and Artistic Applied Studies, Department of the High School of Electrical Engineering and Computer Science, in the subject Programming in integrated technologies, the application of the Python programming language for programming the heuristic optimization method using the genetic algorithm is taught. In one lecture, students introduced the basics of genetic algorithms and the creation of a solution to a given problem using the Python programming language. The solution includes programming using the basic package (modules) of the Python programming language. After that, we learn how to use the Python library GeneAl to solve another problem. During the exercises, students are given a new problem task that they solve as a team using the genetic algorithm. They also use the Python matplotlib library to display the obtained solutions graphically. The paper shows models of solutions to the above problems using genetic algorithms in Upgraded Petri nets. After modeling, these models were converted into concrete Python code. By executing the specific code, the correctness of the model was verified. Through interactive modeling, simulation, and analysis of the given models, the work of the genetic algorithm and the way of coding for the solutions in the Python language are described dynamically.

Key words: Python, GeneAl, matplotlib, genetic algorithm,Upgraded Petri-nets

1. UVOD

Genetski algoritmi pripadaju oblasti evolucionih algoritama za rešavanje problema optimizacije. Razvijeno je dosta tipova genetskih algoritama (jednostavni genetski algoritam, mravlje kolonije, kolonije pčela, itd.)koji su se pokazali podesnim za rešavanje određenih grupa problema [1][2]. Na Akademiji tehničko-umetničkih strukovnih studija na Odseku Visoka škola elektrotehnike i računarstva u Beogradu u okviru predmeta Programiranje u integrisanim tehnologijama sluša se tema o genetskim algoritmima.

Pre progamiranja studentima se pokazuje model genetskog algoritma u nadgrađenim Petri-mrežama gde se dinamičkim izvršavanjem mreže prikaže princip rada genetskog algoritma za dati optimizacioni problem. Nakon modela, studenti programiraju genetski algoritam pri čemu koriste programski jezik *Python,* aako je potrebno dobiti velike performanse onda koriste programski jezik

[1]Profesor, ATUSS – Odsek Visoka škola elektrotehnike i računarstva, Vojvode Stepe 283, Beograd, pericas@viser.edu.rs:
[2]Dipl.inž, Osterus d.o.o, Vase Stajića 24, Novi Sad, strbacpavlejob@gmail.com:
[3]Predavač, ATUSS – Odsek Visoka škola elektrotehnike i računarstva, Vojvode Stepe 283, Beograd, pejanovicm@viser.edu.rs:
[4]Dipl. inž, NewCo. – Naučno-tehnološki park, Veljka Dugoševića 54, Beograd., stefanpejanovic604@gmail.com:

C++. Progamiranje obuhvata pisanje koda bez korišćenja biblioteke *GeneAl* i pisanje koda koji koristi navedenu *Python* biblioteku.

Lekcija o genetskim algoritmima koristi znanja iz prethodnih lekcija koje obrađuju elemente jezika, liste, rečnike, *n*-torke, mape, lambde, redukcije, filtere, matematička izračunavanja i grafičke prikaze.

Studenti na predavanjima prolaze kroz praktičan primer jednostavnog genetskog algoritma gde se upoznaju sa elementima genetskog algoritma: populacijom, hromozomom, funkcijom dobrote, elitizmom, selekcijom, ukrštanjem i mutacijom. Nakon predavanja, studenti na vežbama dobijaju optimizacioni problem koji je potrebno rešiti korišćenjem genetskog algoritma.

U radu je korišćena originalna klasa Petri-mreža, nadgrađene Petri-mreže (*UpgradedPetri-Nets - UPN*)za grafovski prikaz modela [3].Za kreiranje UPNmodela korišćen je originalni programski paket *PeM* (*Petri-netManager*) koji omogućuje modelovanje, simulaciju i analizu nadgrađenih Petri-mreža.

2. *UPN* MODEL JEDNOSTAVNOG GENETSKOG ALGORITMA

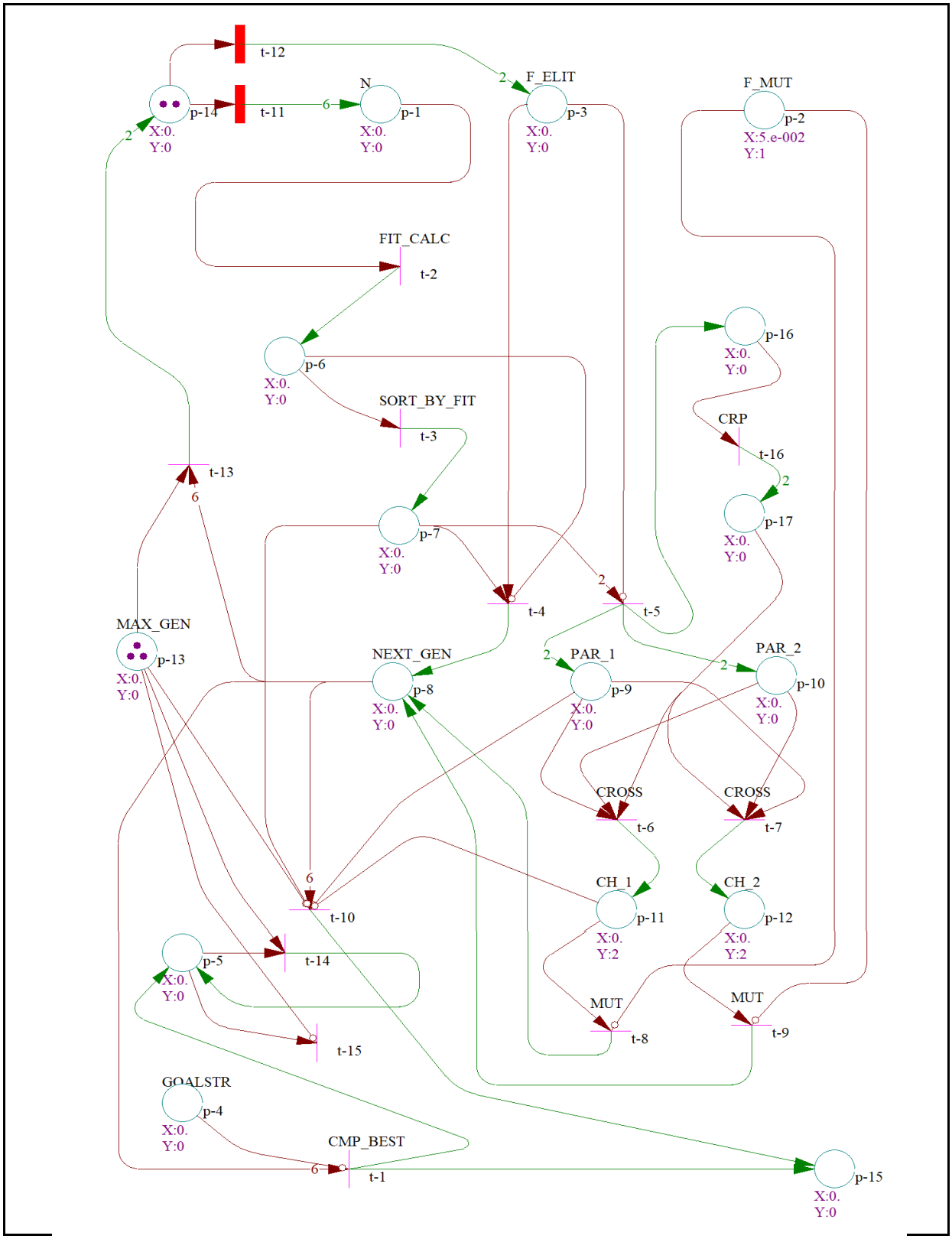
Pored *UPN*koriste se i obojene Petri-mreže za probleme koji se rešavaju genetskim algoritmima [4].Elementarne oznake*UPN*koje će se koristiti u radu su: *t* (prelaz), *p* (mesto),*μ*(označavanje), *B*(izlazna funkcija) i *F* (ulazna funkcija) [3].

Na slici 1 dat je *UPN*model jednostavnog genetskog algoritma koji rešava evoluciju slučajnog teksta u ciljni tekst. Početno označavanje modela je μ_0*(0,0,0,0,0,0,0,0,0,0,0,0,0,3,2,0,0).* Paljenjem skupa prelaza *{t-11, t-12}*dobija se da je označavanje mesta *μ(p-1)=6*zbog *B(t-11,p-1)=6* i da je označa-vanje *μ(p-3)=2*zbog *B(t-12,p-3)=2.* Označavanje mesta *p-1* modeluje broj jedinki u populaciji, dok označavanje mesta *p-3* modeluje broj jedinki za elitizam. Novo označavanje je*μ(6,0,2,0,0,0,0,0,0,0,0,0,3,0,0,0).* Sada je omogućen prelaz *t-2* čijim paljenjem se dobija *μ(5,0,2,0,0,1,0,0,0,0,0,0,0,3,0,0,0)*i zapaljiv je skup prelaza*{t-2, t-3}*.Prelaz *t-2* (*FIT_CALC*) modeluje izračunavanje funkcije dobrote, odnosno, kvaliteta date jedinke. Prelaz *t-3* (*SORT_BY_FIT*) modeluje sortiranje jedinke prema pripadnom kvalitetu.

Daljim iterativnim paljenjem navedenog skupa prelaza sled označavanja će biti:

- *μ(4,0,2,0,0,1,1,0,0,0,0,0,0,3,0,0,0),*
- *μ(3,0,2,0,0,1,2,0,0,0,0,0,0,3,0,0,0),*
- *μ(2,0,2,0,0,1,3,0,0,0,0,0,0,3,0,0,0),*
- *μ(1,0,2,0,0,1,4,0,0,0,0,0,0,3,0,0,0),*
- *μ(0,0,2,0,0,1,5,0,0,0,0,0,0,3,0,0,0*).

Sada je omogućen prelaz *t-3* čijim paljenjem nastaje označavanje*μ(0,0,2,0,0,0,6,0,0,0,0,0,0,3,0,0,0)* i koje omogućuje paljenje prelaza *t-4*.Paljenje prelaza *t-4*će se desiti onoliko puta koliko je označavanje prelaza *p-3* i ono modeluje odabiranje najbolje jedinke za elitizamkoje se smeštaju u sledeću generaciju (mesto *p-8*). Nakon 2 uzastopna paljenja prelaza *t-4* novo označavanje je *μ(0,0,0,0,0,0,4,2,0,0,0,0,0,3,0,0,0)* i mogućen je prelaz *t-5*. Paljenjem prelaza *t-5* modelovano je odabiranje 2 roditelja (*PAR_1*i *PAR_2*)i aktiviranje dela mreže za određivanje tačke ukrštanja (*p-16, t-16, p17*).Novo označavanje je *μ(0,0,0,0,0,0,2,2,2,2,0,0,0,3,0,1,0).* Uzastopnih paljenja prelaza *t-5*će biti *μ(p-7)/2.* Sada je omogućen skup prelaza *{t-5, t-16}.* Paljenje prelaza *t-16* modeluje određivanje tačke ukrštanja.



Slika 1 – UPN model genetskog algoritma za evoluciju slučajnog teksta ka ciljnom tekstu

Paljenjem skupa prelaza *{t-6, t-7}* dobijaju se dva potomka (*CH1*,*CH2*) koji daljim paljenjem skupa prelaza *{t-8, t-9}* mogu biti mutirani prema faktoru mutacije datom u *x* atributu mesta *p-2*. Ovakvi potomci (mutirani ili ne) se smeštaju u sledeću generaciju (*NEXT_GEN*). Nakon formiranja sledeće generacije novo označavanje je *μ(0,0,0,0,0,0,0,0,0,6,0,0,0,3,0,0,0)*.Sada su za modelovani problem moguće tri granekoje su rešene korišćenjem konflikta u *UPN*.

Jedna grana je određena paljenjem prelaza *t-13* što modeluje novi ciklus računanja dobrote, sortiranja, elitizma, ukrštanja i mutacije. Paljenjem prelaza *t-13*smanjuje se označavanje mesta *p-13* koje modeluje maksimalan broj generacija (*MAX_GEN*) i sve jednike iz nove generacije prebacuju se sledom paljenja prelaza *t-13, t-11* u novi iterativni ciklus genetskog algoritma.

Druga grana je određena paljenjem prelaza *t-1* koje modeluje da je najbolja jedinka jednaka ciljnom stringu (*p-4, GOALSTR*).Ovo se može desiti pre maksimalnog broja generacija ili u poslednjoj generaciji.Paljenjem prelaza *t-1* najbolja jedinka se smešta u mesto *p-15*.

Treća grana je određena paljenjem prelaza *t-10*koje modeluje da su završeni svi ciklusi (generacije) genetskog algoritma (*μ(p-13)=0*). U ovom slučaju se opet najbolja jedinka smešta u mesto *p-15*.

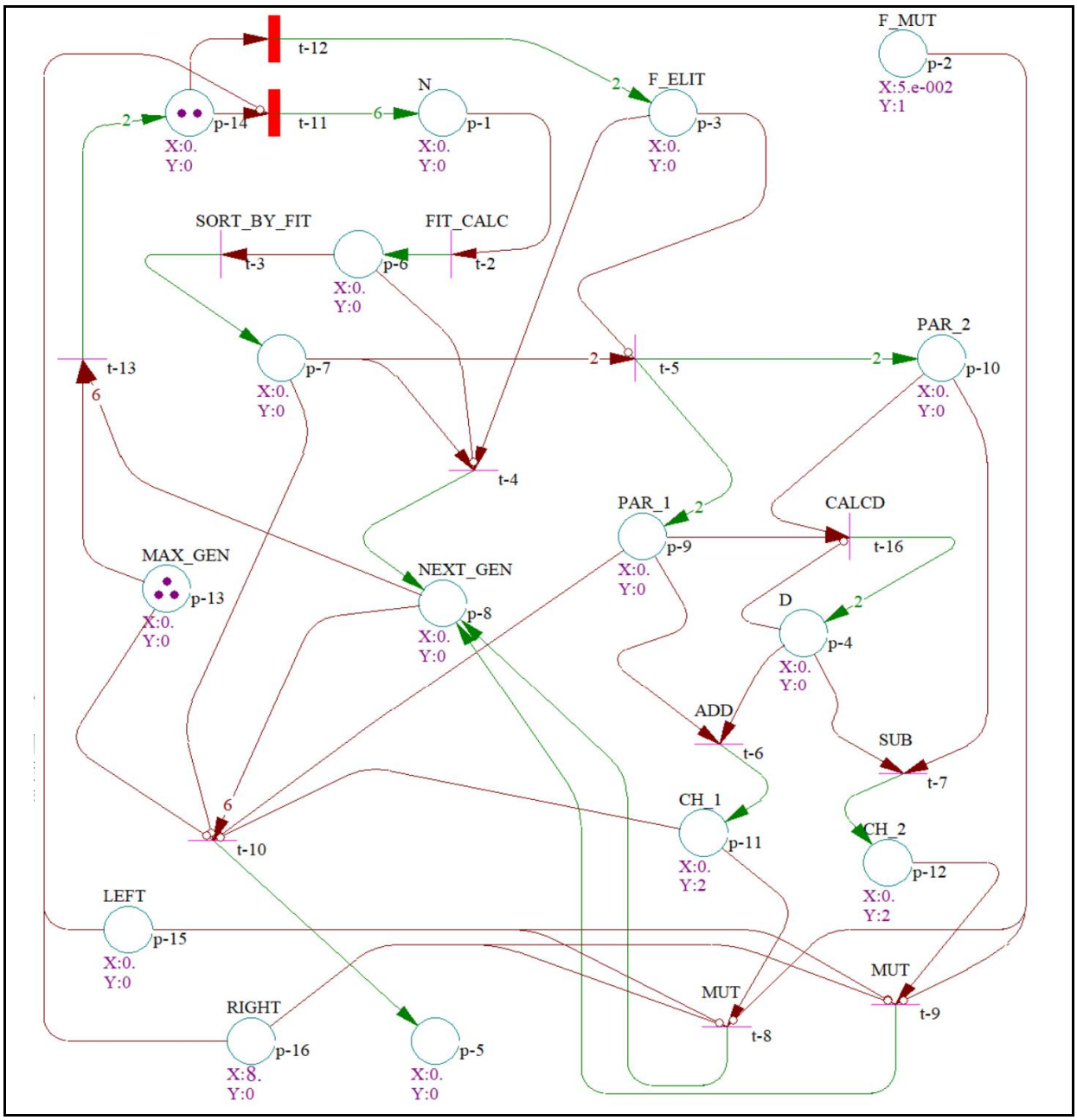
U slučaju da je najbolja jedinka nađena pre maksimalnog broja ciklusa onda će podmreža *p-13, p-5, t-14, t-15*potrošiti sva zaostala markiranja u p-13.

Mrtav čvor *UPN*modela datog na slici 1 je *μ(0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0)*.Mulitipliciteti su: *B(t-11, p-1)=6, B(t-12, p-3)=2, F(t-5, p-7)=2, B(t-5, p-9)=2, B(t-5, p-10)=6, F(t-13, p-8)=6, F(t-10, p-8)=6, F(t-1, p-8)=6, B(t-13, p-14)=2, B(t-16,p-17)=2.*

Nakon objašnjenja *UPN*modela, studentima se daju sledeće postavke zadatka: ciljni tekst je "Hello, world",populacija ima 100 hromozoma (tekstova dužine ciljnog teksta koji inicjalno sadrže ASCIIkaraktere čiji je kod u opsegu od 32 do 127), elitizam je dat kao apsolutna vrednost od 20 jedinki, mutacija je 5%, jednostavna je selekcija, ukrštanje je po jednoj tački (npr. neka su roditelji ABCDEGFHIJKLi MNOPQRSTUVWXi neka je tačka ukrštanja na indeksu 2 onda su deca ABOPQRSTUVWXi MNCDEGFHIJKL), maksimalan broj generacija je 500.Prema datom modelu i zahtevima studenti realizuju svoja programska rešenja genetskog algoritma za evoluciju slučajnog teksta u ciljni tekst.

3. UPN MODEL GENETSKOG ALGORITMA ZA TRAŽENJE MINIMUMA ILI MAKSIMUMA FUNKCIJE U DATOM OPSEGU

Na slici 2 dat je *UPN*model za genetski algoritam koji traži minimum ili maksimum funkcije u zadatom intervalu po *x*. Da li se traži minimum ili maksimum određuje funkcija dobrote (prelaz *t-2*). Početno označavanje je μ_0*(0,0,0,0,0,0,0,3,2,0,0,0,0,0)*. Deo mreže koji se odnosi na kreiranje populacije, računanje dobrote i elitizam je gotovo identičan modelu datiom na slici 1. Ovde još igraju ulogu mesta p-15 i p-16 koja modeluju kontrolu opsega po *x*. Selekcija roditelja (*PAR_1*i *PAR_2*) dalje učestvuje u kalkulaciji distance *D* koja ima vrednost od 0 do dužine razmaka po x (mesto *p-4*). Sada se jedan potomak dobija pomeranjem manjeg roditelja po xza distancu *D*, dok se drugi potomak dobija kada se veći roditelj po *x*pomeri za distancu *-D*. Mutacija je rešena pomeranjem *x*vrednosti potomka u skladu sa faktorom mutacije (mesto *p-2*), levom granicom opsega (mesto p-15) i desnom granicom opsega. Ovo je potrebno da se nedesi zarobljavanje u loklanom rešenju. Ovde je modelovano da se odigraju svi ciklusi genetskog algoritma dati označavanjem mesta *p-13*. Nakon svih ciklusa model dospeva u mrtav čvor (*μ(0,0,0,0,1,0,0,0,0,0,0,0,0,0)*) čime se najbolje rešenje smesti u mesto *p-5*. Dati *UPN* model je potpuno sinhronizovan inhibitorskim ulaznim lukovima.



Slika 2 – UPN model genetskog algoritma za traženje minimuma ili maksimuma funkcije u datom opsegu

4. PROGRAMSKO REŠENJE DATOG GENETSKOG ALGORITMA

Nakon modelovanja, simulacije i analize *UPN* model dat na slici 2 je transformisan u programsko rešenje koje je dato na slici 3. Dato programsko rešenje traži minimum ili maksimum funkcije $f(x)=3-3\sin(x/4)\cdot\cos(x)$ opsegu $x$od0.0 do 8.0 [5].Izlaz programa su tekstualni prikaz najbolje jedinke u svakoj generaciji (slika 4) te grafički prikaz vrednosti svih jedinki u poslednjoj generaciji (slika 5).

```
import math; import random
fun = lambda x: 3-3*math.sin(x/4)*math.cos(x)
POP_SIZE = 100; MUT_FACT = 0.1; ELI_FACT = 0.1; LEFT =   0.0; RIGHT =  8.0
GENERATIONS = 50; population = []
for _ in range(POP_SIZE):
population.append([LEFT+ random.random()*(RIGHT-LEFT),0])
for i in range(POP_SIZE):population[i][1]= fun(population[i][0])
```

```
g = 0
while(g<GENERATIONS):
population.sort(key = lambda x: x[1])#, reverse=True)
   print("GENERATION:{:02d}, x = {:7.12f} , y = {:7.12f})".format
(g,population[0][0],population[0][1]))
new_population=[]
   for i in range (int(POP_SIZE*ELI_FACT)):new_population.append(population[i])
   for i in range(int((POP_SIZE/2)*(1-ELI_FACT))):
      par1 = population[int(random.random()*POP_SIZE)]
      par2 = population[int(random.random()*POP_SIZE)]
      d = random.random()*math.fabs(par1[0]-par2[0])
      if(par1[0]<par2[0]):chi1 = par1[0]+d; chi2 = par2[0]-d
else:chi1 = par1[0]-d; chi2 = par2[0]+d
      if(random.random()<MUT_FACT):
         if(random.random()>0.5): chi1 = chi1 + random.random()*(RIGHT-chi1)
         else:                    chi1 = chi1 - random.random()*(chi1-LEFT)
      if(random.random()<MUT_FACT):
         if(random.random()>0.5): chi2 = chi2 + random.random()*(RIGHT-chi2)
         else:                    chi2 = chi2 - random.random()*(chi2-LEFT)
new_population.append([chi1,fun(chi1)]); new_population.append([chi2,fun(chi2)])
   population = new_population; g+=1
import matplotlib.pyplot as plt
import numpy as np
xvec = []; yvec = []
for i in range(POP_SIZE):
xvec.append(population[i][0]); yvec.append(population[i][1]);
plt.title('GRAFIČKI PRIKAZ POSLEDNJE GENERACIJE')
plt.xlabel('X - OSA'); plt.ylabel('Y - OSA'); plt.axis([0, 8, 0, 6]); plt.scatter(xvec,yvec)
xfun = np.linspace(0,10,1000); yfun = list(map(fun,xfun)); plt.plot(xfun,yfun,color="red")
plt.show()
```

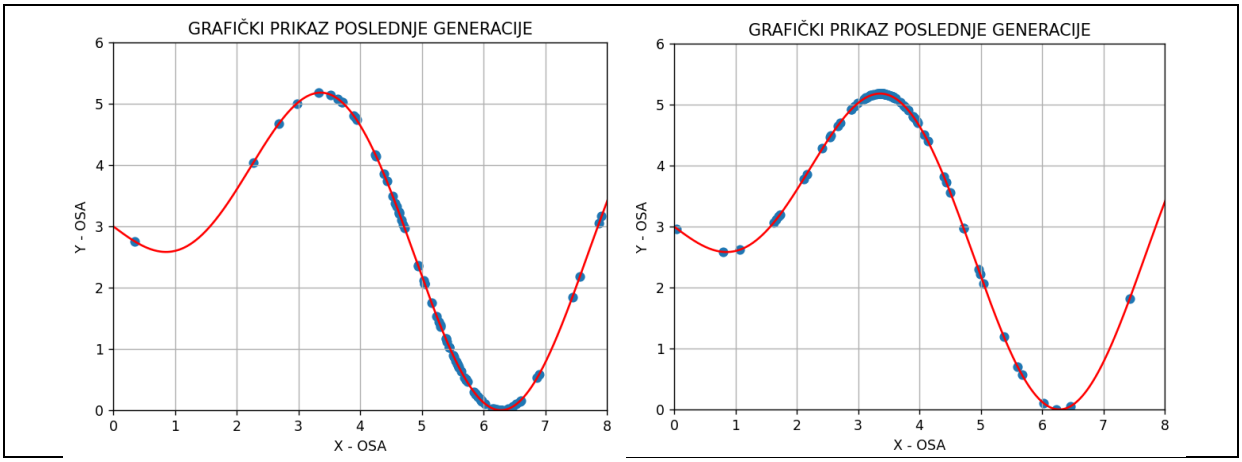
Slika3 – Programski kod dobijen transformacijom UPN modela datog na slici 2.

Na slici 4 dat je tekstualni izlaz programa gde se vide informacije o rednom broju generacije, *x*vrednosti najbolje jedinke i njena pripadna *y*vrednost u toj generacijikod traženja minimuma funkcije.

```
GENERATION:00, x = 6.262072777412 , y = 0.000710362148)
GENERATION:01, x = 6.278463147509 , y = 0.000035538614)
...
GENERATION:48, x = 6.283185827290 , y = 0.000000000000)
GENERATION:49, x = 6.283185827290 , y = 0.000000000000)
```

Slika4 – Najbolje jedinke po generacijmamaza traženje minimuma

Na slici 5 dat je grafički prikaz poslednje generacije za traženje minimuma i traženje maksimuma [6]. Tačkama su označene jedinke a krivom linijom lambda funkcija čiji se minimum traži za dati opseg. Vidi se da je korišćenjem mutacije izbegnuto zaglavljivanje u lokalnom rešenju i da je koncentracija jedinki pomerena ka stvarnom rešenju.



Slika5 – Grafički prikaz vrednosti jedinki u posledljoj generaciji (levo za minimum, desno za maksimum).

5. KORIŠĆENJE BIBLIOTEKE *GENEAL* ZA PROGRAMIRANJE GENETSKOG ALGORITMA

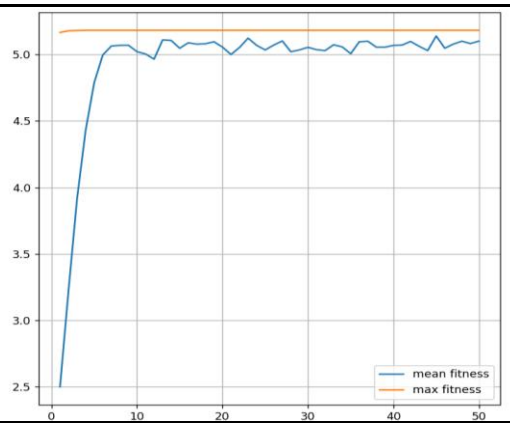
Biblioteka *GeneAl* omogućuje rešavanje genetskih algoritama gde je hromozom opisan binarnim stringom ili realnom vrednošću. Za naš slučaj koristimo drugi slučaj. U kodu datom na slici 6 definisana je funkcija *ffun* čiji maksimum tražimo. Hromozom je realan broj u opsegu od 0.0 do 8.0, populacija ima 100 jedinki, maksimalan broj generacija je 50, faktor mutacije je 0.05, faktor turnirske selekcije je 0.8.

```
import math
from geneal.genetic_algorithms import ContinuousGenAlgSolver
from geneal.applications.fitness_functions.continuous import fitness_functions_continuous
def ffun(x):    return 3-3*math.sin(x/4)*math.cos(x)
solverC = ContinuousGenAlgSolver(
   n_genes=1,
   fitness_function=ffun,
   pop_size=100,    max_gen=50,    mutation_rate=0.05,    selection_rate=0.8,
   selection_strategy="tournament", problem_type=float,    variables_limits=(0, 8)
)
solverC.solve()
```

Slika 6 – Programski kod za traženje maksimuma funkcije korišćenjem biblioteke GeneAl.

Na slici 7 dat je izlaz programa datog na slici 6 gde se prikazuju najveće vrednosti funkcije nađene u svakoj petoj generaciji (iteraciji) te grafički prikaz najvećih vrednosti po generacijama.

```
Iteration: 5     Best fitness: 5.180862777554572
Iteration: 10    Best fitness: 5.180863931350707
Iteration: 15    Best fitness: 5.180863931350823
Iteration: 20    Best fitness: 5.180863931350823
Iteration: 25    Best fitness: 5.180863931350823
Iteration: 30    Best fitness: 5.180863931350823
Iteration: 35    Best fitness: 5.180863931350823
Iteration: 40    Best fitness: 5.180863931350823
Iteration: 45    Best fitness: 5.180863931350823
Iteration: 50    Best fitness: 5.180863931350823
```



Slika7 – Izlaz izvršavanja programskog koda datog na slici 6.

6. ZAKLJUČAK

Rad daje prikaz *UPN* modelovanja problema koji se rešava genetskim algoritmima te transformaciju*UPN* modela u programski kod jezuka *Python.* U modelima su implementirani paralelizam i sinhronizacija tako da bi se programsko rešenje moglo dobiti i konkurentnim programiranjem. Studenti kroz faze modelovanja, simulacije i analize dolaze do podesnog modela koga onda transformišu u programski kod. Programski kod mogu relizovati bez korišćenja biblioteke *GeneAl* ili korišćenjem navedene biblioteke. U programskom kodu menjanjem veličine populacije, tipa selekcije, tipa ukrštanja, načina mutacije i elitizma studenti eksperimetalno dolaze do sve boljeg programskog rešenja za dati problem koji se rešava genetskim algoritmom. Korišćenjem grafičkog prikaza populacije studenti mogu da uoče migraciju jedinki ka pravom rešenju i provere da li je došlo do zarobljavanja u lokalnom rešenju i koliko je kvalitetna tekuća postavka ulaznih parametara za genetski algoritam.

7. LITERATURA

[1] Sourabh, K., Sumit, S. C., Vijay K. (2021). *A review on genetic algorithm: past, present, and future*.Multimedia Tools and Applications80:8091–8126.

[2] Kulida, E. L., Lebedev, V. G. (2017). *Genetic Algorithm for Generating Trajectories of Specified Length.* International Conference on Knowledge Based and Intelligent Information and Engineering Systems, September 2017, Marseille, France, 1015-1022.

[3] Štrbac, P., Milovanović, G.V. (2013). *Upgraded Petri net model and analysis of adaptive and static arithmetic coding*, Elsevier: Mathematical and Computer Modelling Vol. 58, 1548–1562.

[4] Si, Y., Chan, Chan, V., Marlon, D., Defu, Z. (2018). *A Petri Nets Based Generic Genetic Algorithm Framework for Resource Optimization in Business*. Simulation Modelling Practice and Theory, Volume 86, 2018, 72-101.

[5] Entisar, A. S., Gubara, M. A. (2015) *Strategy of finding the maximum and minimum values of the function of n-variables with and without constraint.*International Journal of Engineering Sciences & ResearchTechnology, 247-251.

[6] Farooq, H., F., Nordin, M. Z., Mohd, F. H., Sulaiman, S. (2011) *An Interactive Visualization of Genetic Algorithm on 2-D Graph*.International Journal of Software Science and Computational Intelligence, August 2011, 1-8.