Duboki evolucijski algoritmi za samopodešavanje evolucijskih hiperparametara

Homolak, Sandi

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Sandi Homolak

Zagreb, 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentori:

Izv. prof. dr. sc. Petar Ćurković, dipl. ing.

Student:

Sandi Homolak

Zagreb, 2022.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem mentoru, izv. prof. dr. sc. Petru Ćurkoviću na ustupljenim materijalima, stručnim savjetima, upoznavanju s tematikom ovog rada, ali ponajviše na susretljivosti i ugodnoj atmosferi koju je pružao tijekom čitavog studija.

Ovom se prilikom također želim zahvaliti doc. dr. sc. Tomislavu Stipančiću čiji je pristup radu i predanost studentima također ostavio jak dojam.

Konačno, želim se zahvaliti svojoj obitelji, djevojci i bliskim prijateljima koji su mi pružali podršku, a posebno kolegi Ivanu Uroiću na odličnom društvu tijekom diplomskog studija.

Sandi Homolak

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske radove studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment,
inženjerstvo materijala te mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum:	Prilog:
Klasa: 602-14/22-6/1	
Ur. broj: 15-1703-22-	

DIPLOMSKI ZADATAK

Student: **SANDI HOMOLAK** Mat. br.: 0035199603

Naslov rada na hrvatskom jeziku: **Duboki evolucijski algoritmi za samopodešavanje evolucijskih hiperparametara**

Naslov rada na engleskom jeziku: **Deep evolutionary algorithms for evolutionary parameters self-tuning**

Opis zadatka:

Evolucijski algoritmi pripadaju području prirodom inspiriranih algoritama. Koriste se za pronalazak kvalitetnih rješenja teških optimizacijskih problema. Osnovna prednost ovakvih algoritama spram drugih tehnika optimiranja leži u njihovoj robusnosti, odnosno otpornosti na zapinjanje u lokalnim ekstremima. Ipak, evolucijski algoritmi počivaju na nizu kompleksnih parametara, tzv. hiperparametara čiji ispravan izbor i kombinacija presudno utječe na uspješnost algoritma. Postići optimalan izbor ovih parametara kompliciran je postupak i najčešće se svodi na metodu pokušaja i pogreške što je vremenski neefikasno i podložno subjektivnom tumačenju.

U ovom radu predlaže se metoda dubokog evolucijskog algoritma temeljena na ugniježđivanju dvaju evolucijskih algoritama s ciljem pronalaska optimalnih hiperparametara algoritma za različite optimizacijske probleme.

Potrebno je napraviti sljedeće:

- proučiti postojeće pristupe autonomnog podešavanja hiperparametara
- predložiti metodu podešavanja hiperparametara temeljenu na ugniježđivanju dvaju evolucijskih algoritama
- testirati predloženu metodu na standardnim ispitnim (engl. benchmark) funkcijama
- kritički se osvrnuti na dobivene rezultate i napraviti usporedbu s baznim algoritmom kod kojega su hiperparametri podešeni metodom pokušaja i pogreške.

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan: 29. rujna 2022.

Rok predaje rada: 1. prosinca 2022.

Predviđeni datum obrane: 12. prosinca do 16. prosinca 2022.

Zadatak zadao:
izv. prof. dr. sc. Petar Ćurković

Predsjednica Povjerenstva:
prof. dr. sc. Biserka Runje

SADRŽAJ

POPIS SLIKA

POPIS OZNAKA

Oznaka	Mjerna jedinica	Opis oznake
a	-	Faktor skaliranja linijske rekombinacije
a_i	-	Faktor skaliranja *i*-te varijable kod intermedijarne rekombinacije
A	-	Duljina niza
B_{unut}	-	Veličina populacije unutarnjeg algoritma
B_{vanj}	-	Veličina populacije vanjskog algoritma
d	-	Parametar koji ukazuje na dimenzije hiperkocke kod intermedijarne rekombinacije
f_i	-	Vrijednost dobrote *i*-tog pojedinca iz populacije
F	-	Maksimizacijska/minimizacijska funkcija
F_p	-	Fenotipski prostor
g	-	Broj izvođenja fitnes funkcije unutarnjeg algoritma
G_p	-	Genotipski prostor
L	-	Broj ponavljanja unutarnjeg evolucijskog algoritma s istim hiperparametrima
m	-	Broj kvalitativnih parametara
M	-	Broj pojedinaca unutar populacije
n	-	Broj kvantitativnih parametara
N	-	Broj koji predstavlja koliko puta se nije pronašlo rješenje unutar određenog broja iteracija
N_g	-	Ukupni broj gradova
N_{var}	-	Ukupan broj varijabli
p_c	-	Vjerojatnost križanja
p_m	-	Vjerojatnost mutacije
P_i	-	Vjerojatnost odabira *i*-tog pojedinca iz populacije
Q_i	-	*i*-ti kvalitativni parametar
R_i	-	*i*-ti kvantitativni parametar
S	-	Prostor svih parametara evolucijskog algoritma

S_h	-	Prostor svih kvalitativnih parametara evolucijskog algoritma
S_s	-	Prostor svih kvantitativnih parametara evolucijskog algoritma
t	s	Vrijeme potrebno za sva izvođenja unutarnjeg algoritma s istim hiperparametrima
$\bar{\boldsymbol{v}}$	-	Aritmetička sredina svih članova vektora iteracija
$\tilde{\boldsymbol{v}}$	-	Medijan svih članova vektora iteracija
$Var_i^{\,potomak}$	-	Vrijednost *i*-te varijable potomka
$Var_i^{\,roditelj}$	-	Vrijednost *i*-te varijable roditelja
x_i	-	*i*-ti član potencijalnog rješenja
$\mathbf{X}$	-	Potencijalno rješenje u obliku niza
$\mathbf{X_g}$	-	Matrica koja sadrži Kartezijeve koordinate posjećenih gradova u redoslijedu njihovog posjećivanja
$\Phi_{k,z}$	-	*z*-ta fitnes funkcija za *k*-ti problem

POPIS KRATICA

Kratica	**Opis**
ANN	*Artificial neural networks* – Umjetne neuronske mreže
CI	*Computational Intelligence* – Računalna inteligencija
EA	*Evolutionary algorithms* – Evolucijski algoritmi
EC	*Evolutionary Computation* – Evolucijsko računarstvo
EP	*Evolutionary programming* – Evolucijsko programiranje
ES	*Evolution strategies* – Evolucijske strategije
FUZZY	*Fuzzy logic* – Neizrazita logika
GA	*Genetic algorithms* – Genetski algoritmi
GP	*Genetic programming* – Genetsko programiranje
TSP	*Travelling salesman problem* – Problem trgovačkog putnika

SAŽETAK

U ovom diplomskom radu predlaže se metoda dubokog evolucijskog algoritma temeljena na ugnježđivanju dvaju evolucijskih algoritama s ciljem pronalaska kvalitetne kombinacije hiperparametara algoritama za različite optimizacijske probleme. Dan je pregled područja evolucijskog računarstva s naglaskom na evolucijske algoritme. Opisana je struktura evolucijskih algoritama te raznih metoda koje se koriste za postavljanje njihovih hiperparametara. U nastavku rada slijedi strukturni prikaz metode koja se predlaže te konačno, njezina implementacija na nekoliko standardnih ispitnih zadataka.

Ključne riječi: evolucijsko računarstvo, evolucijski algoritam, postavljanje hiperparametara

SUMMARY

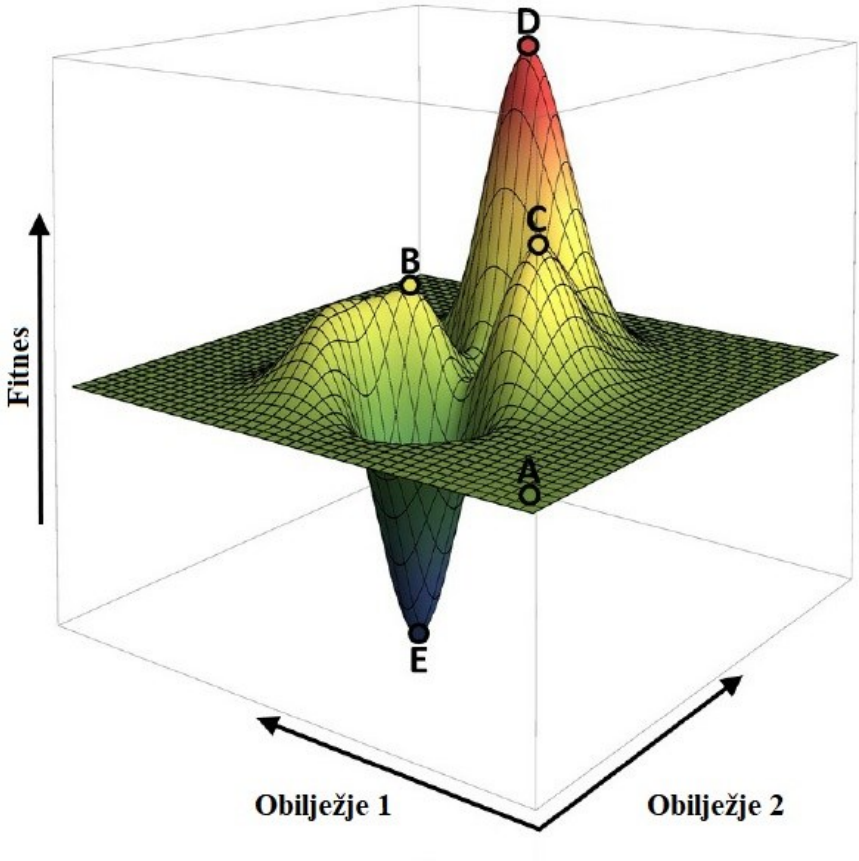
In this thesis, a deep evolutionary algorithm method based on the nesting of two evolutionary algorithms is proposed with the aim of finding a set of high-quality hyperparameters for different optimization problems. An overview of the field of evolutionary computation with an emphasis on evolution algorithms is given. The structure of evolutionary algorithms and various hyperparameter tuning methods are described. Furthermore, a structural presentation of the purposed method is shown, and finally, its implementation on several benchmark problems.

Key words: evolutionary computation, evolutionary algorithm, hyperparameter tuning

1. UVOD

Sredinom 19. stoljeća Charles Darwin je iznio Teoriju evolucije [1] koja nudi objašnjenja o podrijetlu biološke raznolikosti. Teorija evolucije počiva na procesu prirodne selekcije. Unutar ograničenog prostora i životnih resursa postoji određeni broj jedinki koji može opstati. Prema Hardy-Weinbergovom principu [2], koji se smatra temeljem populacijske genetike, populacija ne može evoluirati u standardnim uvjetima okoline ako se frekvencije gena i genotipova ne mijenjaju u nizu sukcesivnih generacija. Kako bi se održala genetička ravnoteža u populaciji, neophodno je odsustvo evolucijskih faktora koji je remete, a to su ponajprije prirodni odabir i mutacija. Darwin je svoju teoriju temeljio na opažanjima živog svijeta gdje u pravilu nastaje više potomaka nego što može opstati te gdje u svakoj novoj generaciji nalazimo više jedinki potomaka onih roditelja koji su bili bolje prilagođeni za život u danim uvjetima, što je rezultiralo i većom šansom za razmnožavanje. Obilježja koja na to utječu nazivamo fenotipskim obilježjima, a njihov uspjeh u konačnici ocjenjuje sama okolina. Neka fenotipska obilježja mogu biti nasljedna, međutim Darwin je uočio i nastanak nasumičnih varijacija u potomstvu koje mogu biti rezultat isključivo mutacije. Tako se novonastale kombinacije fenotipskih obilježja ponovno ocjenjuju unutar okoline i prenose na nove potomke. Ako je neka mutacija dovela do superiornijih obilježja određene populacije, jedinke koje posjeduju spomenutu novu kombinaciju obilježja imat će veću šansu za preživljavanje i razmnožavanje što će dovesti do danjeg prenošenja istih obilježja u nove generacije.

Kvaliteta kombinacije obilježja jedinke za danu okolinu izražava se kao njezin fitnes. Fitnes funkcija predstavlja funkciju koja uzima sva promatrana obilježja neke jedinke te nam daje kvantitativnu vrijednost koja predstavlja njihovu šansu za opstanak i daljnju reprodukciju. Ako promatramo n obilježja neke populacije, njihov adaptivni krajolik (*eng. adaptive landscape*) možemo zorno prikazati u n+1 dimenzija kao što se može vidjeti na slici 1.



Slika 1. Primjer adaptivnog krajolika [3]

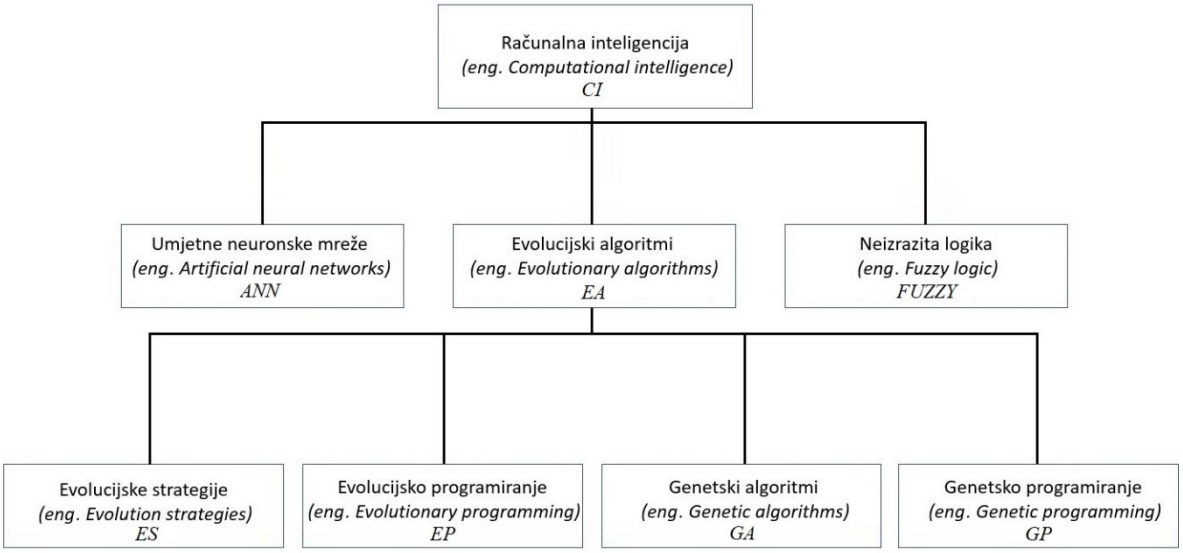
Na slici 1 može se uočiti da određene kombinacije obilježja, a koje su ovdje označene točkama B, C, D i E predstavljaju ekstremna odstupanja fitnesa u odnosu na prosječan iznos (točka A). Tako će za dani primjer jedinke koje posjeduju kombinacije obilježja koje odgovaraju okolini točaka B, C i D imati natprosječan fitnes i stoga veće šanse da se prenesu u nove generacije, dok će jedinke koje posjeduju kombinacije obilježja koje odgovaraju okolini točke E, odnosno globalnom minimumu, imati vrlo male šanse za preživljavanje i reprodukciju.

U slučaju kada se proces evolucije odvija s populacijom ograničene veličine bitno je spomenuti i fenomen poznat pod nazivom genetski drift. Naime, obzirom da se prilikom postupka selekcije i mutacije odvijaju nasumični odabiri, moguće je da se izgubi raznolikost obilježja u populaciji te da se bez obzira na fitnes funkciju članovi populacije udalje od lokalnih optimuma čime prelaze u područja nižeg fitnesa. Dakle, bez obzira na značaj fitnes

funkcije pri odabiru roditelja, moguće je da fitnes najboljeg pojedinca unutar nove generacije bude niži od najboljeg pojedinca prethodne.

1.1. Evolucijsko računarstvo

Koncept oponašanja evolucijskih procesa za rješavanje složenih problema potječe još iz vremena prije pojave modernog računala. Alan Turing je bio prvi koji se sugerirao da bi prirodna evolucija mogla predstavljati inspiraciju za pristup umjetnoj inteligenciji [4]. Evolucijsko računarstvo *(eng. Evolutionary Computation, EC)* ozbiljno je započelo 1950-ih. U to vrijeme bilo je nekoliko neovisnih pokušaja korištenja procesa evolucije u računarstvu, koji su se odvojeno razvijali narednih 15 godina. Tijekom ovog razdoblja pojavila su se tri pristupa: evolucijske strategije, Rechenberg i Schwefel [5]; evolucijsko programiranje, Fogel, Owens i Walsh [6]; genetski algoritmi, Holland [7], a četvrta grana, genetsko programiranje, Koza [8] pojavila se sredinom 1980-ih. Ti se pristupi razlikuju u načinu selekcije i mutacije te načinu kodiranja pojedinaca unutar populacije. Do 1990-ih, razlike između navedenih grana počele su se zamagljivati, a izraz 'evolucijsko računarstvo' nastao je 1991. da označi znanstveno područje koje obuhvaća sve navedene grane.



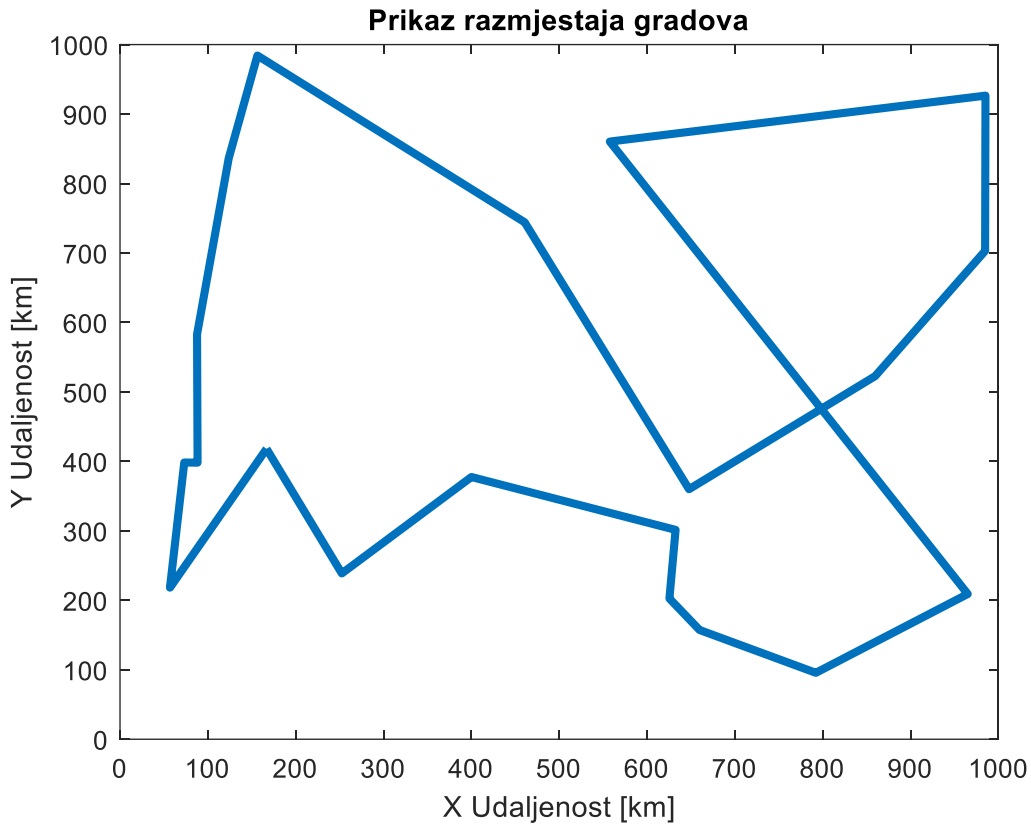
Slika 2. Podjela računalne inteligencije

1.2. Evolucijski algoritam

U računalnoj inteligenciji *(eng. Computational intelligence, CI)*, evolucijski algoritam (EA) najpoznatiji je predstavnik skupine biološki inspiriranih algoritama. Evolucijski algoritam koristi mehanizme koji proizlaze iz bioloških procesa evolucije poput reprodukcije, rekombinacije, mutacije i selekcije. Kandidati za moguće rješenje preuzimaju ulogu jedinki unutar populacije, a njihova sposobnost preživljavanja, definirana je tzv. fitnes funkcijom čija je zadaća kvantitativno prikazati kvalitetu svakog kandidata unutar pojedine generacije.

Evolucijski algoritmi imaju vrlo široko područje primjene. U radu [9] prikazana je uporaba evolucijskog algoritma za smanjenje energetske potrošnje tijekom procesa pročišćavanja prirodnog plina s visokim sadržajem sumpora. U navedenom radu demonstrirana je mogućnost smanjenja energetske potrošnje unutar postrojenja za čak 12.,7%. U radu [10] korišten je evolucijski algoritam kako bi se odredio investicijski plan za ojačanje kineskog željezničkog sustava od potresa i drugih elementarnih nepogoda. Dobiveni rezultati kasnije su provedeni u praksi te su se pokazali kvalitetnijim od rješenja dobivenih klasičnim metodama. U radu [11] prikazano je korištenje evolucijskog algoritma za segmentaciju medicinskih slika. Eksperimentalni rezultati pokazali su da predloženi pristup može proizvesti kvalitetnije rezultate od preostalih postojećih metoda. U radu [12] prikazana je mogućnost korištenja evolucijskih algoritama za upravljanje višeagentskim robotskim sustavom.

Kod evolucijskih algoritama, bitno je razlikovati fenotip i genotip, odnosno fenotipski prostor F_p i genotipski prostor G_p. Fenotip predstavlja stvarni prikaz nekog rješenja, dok je genotip kodirani prikaz tog rješenja unutar genotipskog prostora G_p koji je prikladan za provođenje operacija evolucijskog algoritma. Tako na primjer, kod rješavanja problema trgovačkog putnika koji će detaljnije biti opisan u nastavku rada, fenotip nekog rješenja predstavlja stvarni put koji je pronađen kao potencijalno rješenje (slika 3), dok bi genotip istog rješenja bio redom zapisan popis koordinata posjećenih gradova gledanog rješenja koji se koristi pri provođenju operacija evolucijskog algoritma.



Slika 3. Primjer fenotipa rješenja problema trgovačkog putnika

Potrebno je naravno definirati metodu preslikavanja iz fenotipskog u genotipski prostor kako bi za specifičan problem znali generirati inicijalnu populaciju, ali i metodu preslikavanja iz genotipskog prostora u fenotipski kako bi dobiveno rješenje mogli prikazati na kvalitetan i svojstven način.

Obično se provodi binarno kodiranje ili kodiranje realnim vrijednostima. Pojedini član populacije, odnosno pojedino rješenje koje kao i u biologiji ponekad nazivamo kromosom ili gen, može biti prikazano na različite načine kao što su niz znakova, polje ili stablo.

Tipičan evolucijski algoritam sastoji se od nekoliko koraka:

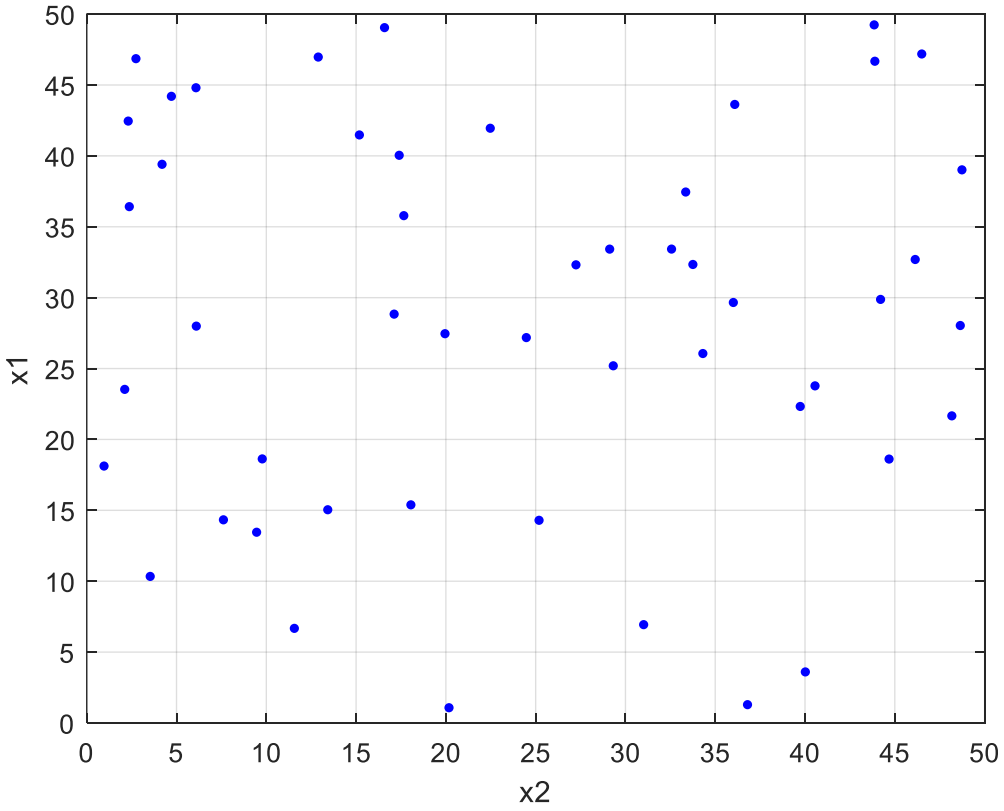
1. nasumično generiranje inicijalne populacije jedinki
2. evaluacija fitnesa svake jedinke iz populacije
3. selekcija roditelja iz trenutne populacije
4. postupak križanja i mutacije čime dobivamo potomstvo

5. provođenje elitizma
6. ponavljanje koraka 2. – 5. dok nije zadovoljen uvjet zaustavljanja.

U nastavku će svaki od koraka biti detaljnije razrađen.

1.2.1. Generiranje inicijalne populacije

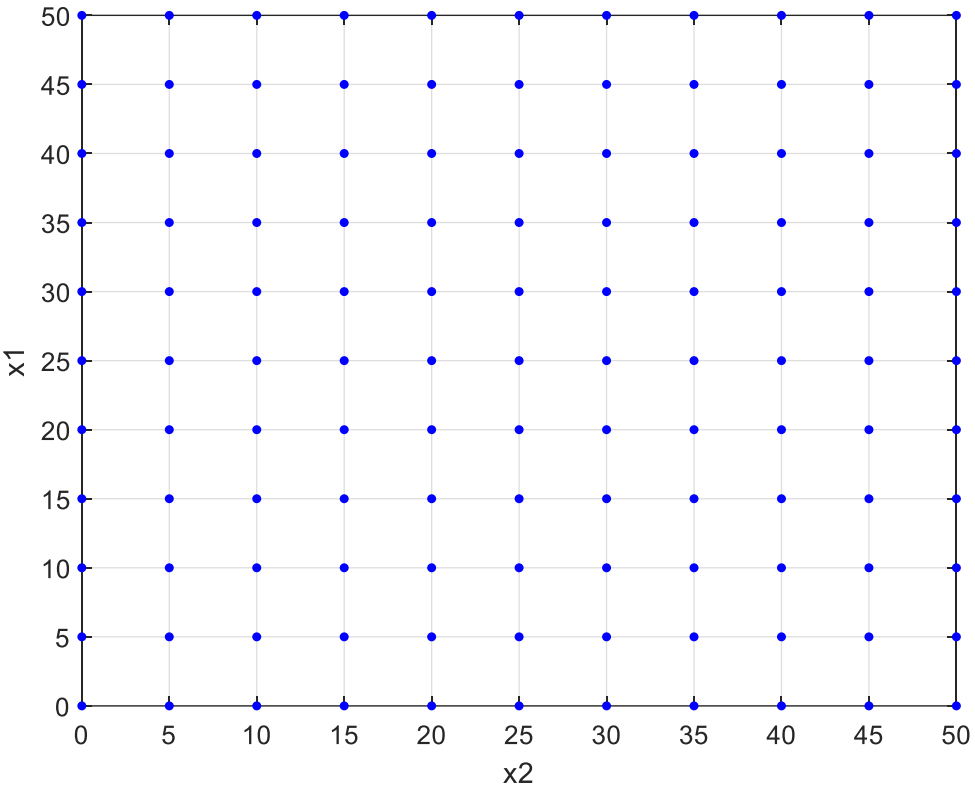
Generiranje inicijalne populacije jedinki moguće je provesti na različite načine. Uzmimo za primjer problem čije potencijalno rješenje zavisi o dvije varijable (x_1 i x_2), odnosno problem čije rješenje tražimo u dvodimenzionalnom prostoru. Recimo također da su vrijednosti x_1 i x_2 ograničene na interval [0, 50]. Inicijalnu populaciju možemo generirati potpuno nasumično čime ćemo dobiti distribuciju jedinki unutar inicijalne populacije koja nalikuje distribuciji prikazanoj na slici 4.



Slika 4. Primjer inicijalizacije po slučaju

Ova metoda inicijalizacija pogodna je za testiranje evolucijskih algoritama s obzirom na to da unutar postupka inicijalizacije nije iskorišteno nikakvo prethodno saznanje o potencijalnoj

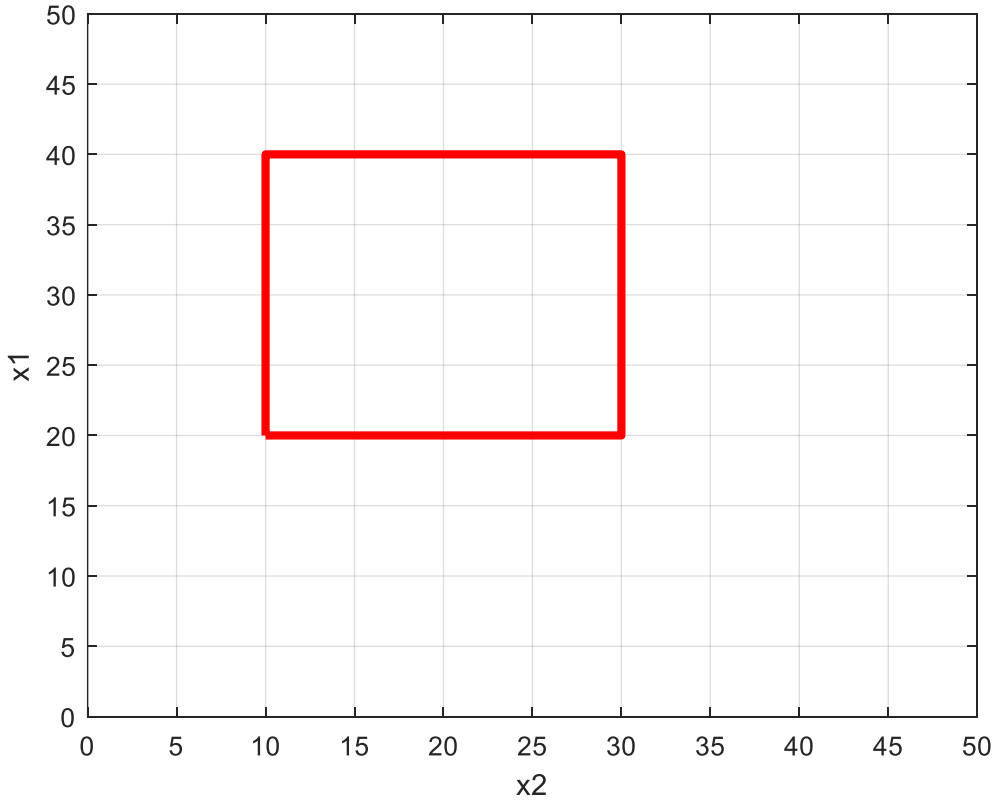
lokaciji optimuma. Međutim, kao što se može vidjeti na slici 4, ako koristimo inicijalizaciju po slučaju, vrlo je vjerojatno da će neki od dijelova prostora pretrage biti više zastupljeni unutar inicijalne populacije od drugih. To dovodi do mogućih problema prilikom potrage za globalnim optimumom, jer ako prostor oko njegove lokacije nije zastupljen u inicijalnoj populaciji, evolucijski algoritam može vrlo brzo konvergirati u nekom drugom lokalnom optimumu. Naravno, postupkom križanja i mutacije, algoritam još uvijek može ispitati dijelove prostora koji nisu zastupljeni u inicijalnoj populaciji, međutim to nije zagarantirano. Ako želimo biti sigurni da je svaki dio prostora jednako zastupljen u inicijalnoj populaciji, možemo provesti mrežnu inicijalizaciju kao što je prikazano na slici 5.



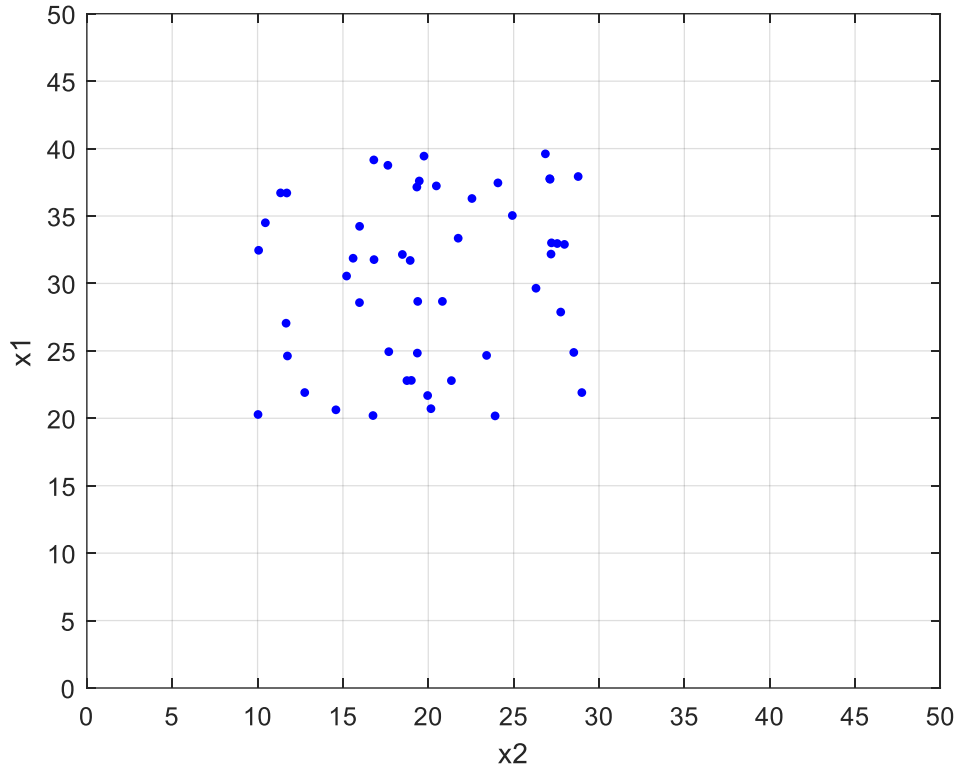
Slika 5. Primjer mrežne inicijalizacije

Mrežna inicijalizacija često dovodi do brzog pronalaska okolnog područja globalnog optimuma, stoga može biti vrlo efikasna. Međutim, također može usporiti sam proces evolucije s obzirom na to da klasična metoda križanja ne može rezultirati novim neispitanim rješenjima dok ne nastupi mutacija.

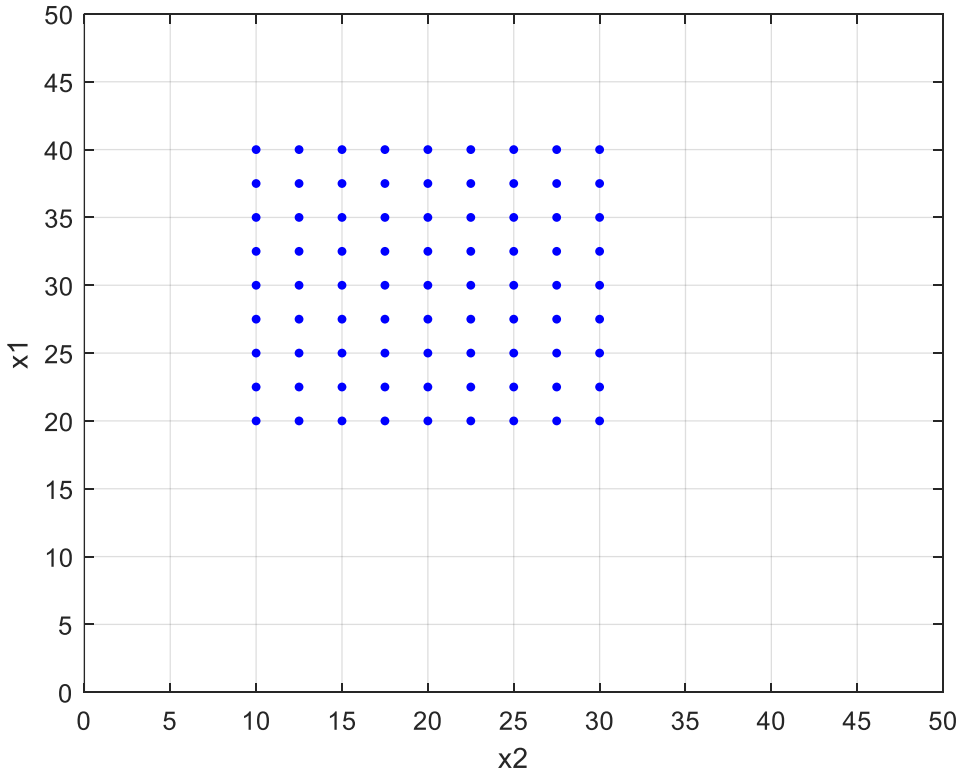
Ako postoji određeno znanje o domeni, npr. znamo da rješenja unutar područja označenog na slici 6 rezultiraju većom dobrotom, možemo suziti prostor pretrage kao što je prikazano na slici 7 i slici 8. Gdje je inicijalizacija po slučaju uz distribuciju temeljenu na znanju analogna inicijalizaciji po slučaju prikazanoj na slici 4 uz suženo područje pretrage, a mrežna inicijalizacija uz distribuciju temeljenu na znanju analogna mrežnoj inicijalizaciji prikazanoj na slici 5 također uz suženo područje pretrage.



Slika 6. Primjer područja izražene dobrote



Slika 7. Primjer inicijalizacije po slučaju uz distribuciju temeljenu na znanju



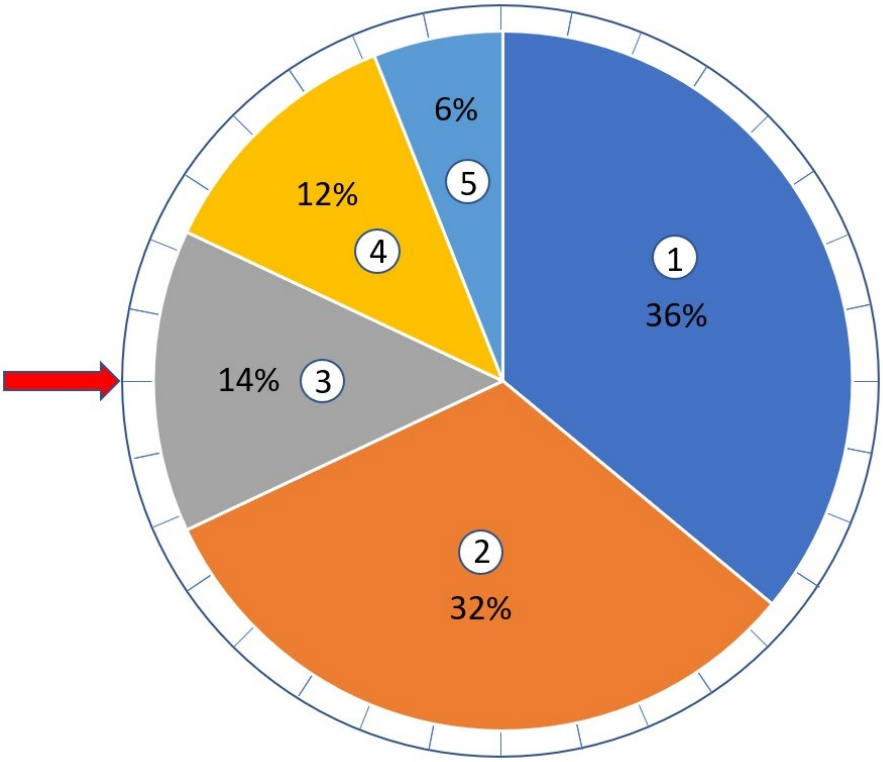
Slika 8. Primjer mrežne inicijalizacije uz distribuciju temeljenu na znanju

1.2.2. Funkcija dobrote

Nakon generiranja inicijalne populacije, svako pojedinačno rješenje unutar početne populacije je potrebno evaluirati korištenjem fitnes funkcije. Oblik koji poprima fitnes funkcija zavisi o željenom rezultatu evolucijskog algoritma za specifičan problem koji se rješava. Iz tog razloga, definiranje fitnes funkcije u konačnici predstavlja najveći utjecaj koji imamo pri vođenju evolucijskog procesa u željenom smjeru. Definiranje fitnes funkcije za zadani problem nije jednostavno kao što se na prvi pogled čini i često predstavlja najteži zadatak koji se nameće pri izradi evolucijskih algoritama. Bitno je također napomenuti da ne postoji način kako bi se dokazalo da je određen oblik fitnes funkcije optimalan za specifičan problem, stoga fitnes funkciju treba usavršavati dok nismo zadovoljni s dobivenim rezultatom. Postupak usavršavanja fitnes funkcije oduzima puno vremena jer kvalitetu fitnes funkcije ne možemo evaluirati dok ne vidimo kakav utjecaj je imala na vođenje cjelokupnog evolucijskog procesa. Iz tog se razloga za definiranje fitnes funkcije uglavnom koristi metoda pokušaja i pogreške uzimajući u obzir željena svojstva rezultata.

1.2.3. Selekcija

Nakon izračuna fitnesa svakog pojedinačnog rješenja unutar populacije, potrebno je izvršiti selekciju pojedinaca koji će predstavljati roditelje pri formiranju iduće generacije rješenja. Postoji više tipova selekcije, a standardno se koristi selekcija proporcionalna dobroti korištenjem ruletnog pravila (slika 9).



Slika 9. Ilustrativni prikaz korištenja ruletnog pravila

Vjerojatnost odabira P nekog pojedinca i iz populacije dobivamo normaliziranjem vrijednosti njegove dobrote f_i prema ukupnoj vrijednosti dobrote cjelokupne populacije koja se sastoji od M pojedinaca:

$$P_i = \frac{f_i}{\sum_{j=1}^{M} f_j}. \tag{1}$$

Tako bi za primjer koji je prikazan na slici 9 kod kojeg se populacija sastoji od 5 pojedinaca, vjerojatnost selekcije prvog rješenja iznosila 36%, za drugog 32% itd.

Bitno je spomenuti da postoje dva potencijalna problema koja se mogu pojaviti pri korištenju ove metode selekcije. Prvi problem je prevladavanje pojedinaca koji iskazuju visoke vrijednosti dobrote već u prvim iteracijama evolucijskog algoritma. Naime, bez obzira na to što je sama ideja algoritma pronalazak takvih rješenja s ciljem vođenja evolucijskog postupka u željenom smjeru, ako takva rješenja prevladaju već u prvim iteracijama, opseg pretraženog prostora neće biti dovoljno velik da bismo ta rješenja mogli prozvati globalnim optimumom.

Dakle, moguće je da algoritam prerano „zapne“ u potprostoru nekog lokalnog optimuma koji je prevladao jer je u tom trenutku predstavljao najbolje moguće rješenje. Kako bi se riješio navedeni problem, često se koristi metoda skaliranja dobrote, čime se umanjuje značajna razlika dobrote određenih pojedinaca. Drugi problem koji se može pojaviti je nastajanje sličnih vrijednosti dobrote kod pojedinaca čija se obilježja značajno razlikuju jer se u tom slučaju gubi evolucijski pritisak koji bi trebao voditi evoluciju u željenom smjeru.

Osim ruletnog pravila, moguće je još provoditi selekciju najboljeg postotka kod koje se prilikom selekcijskog postupka razmatraju samo ona rješenja koja ulaze u određen postotak najboljih. Postotak najboljih rješenja kod ove metode definira korisnik.

Turnirska selekcija se često koristi kod velikih populacija. Kod ovog tipa selekcije se odabire određen broj nasumičnih rješenja iz populacije te se potom rješenje s najvećom iskazanom dobrotom odabire kao pobjednik turnira. Postoji i tzv. deterministička turnirska selekcija kod koje se odabire samo jedno rješenje iz populacije te kao takva predstavlja ekvivalentnu metodu nasumičnoj selekciji.

Bitno je spomenuti i eliminacijske genetske algoritme (*eng. steady-state genetic algorithm*) kod kojih metoda selekcije poprima drugačiji oblik. Naime, eliminacijski genetski algoritmi predstavljaju alternativan pristup kod kojeg se prilikom selekcije, osim odabira rješenja s najvećom iskazanom dobrotom, odabiru i rješenja s najnižom vrijednošću iskazane dobrote. Kasnije, nakon provedenog postupka križanja i mutacije, novonastala rješenja zamjenjuju prethodno spomenuta rješenja nisko iskazane dobrote u novoj populaciji, a sva ostala rješenja unutar nove populacije prepisuju se iz prethodne populacije. Ovaj oblik genetskog algoritma ima svoje prednosti i nedostatke u odnosu na klasičan pristup. Male promijene koje se odvijaju unutar populacije tijekom više generacija osigurat će konvergenciju, ali će također smanjiti prostor pretrage i usporiti cjelokupan evolucijski proces s obzirom na to da će svaka iteracija dovesti do minimalnih promjena unutar populacije.

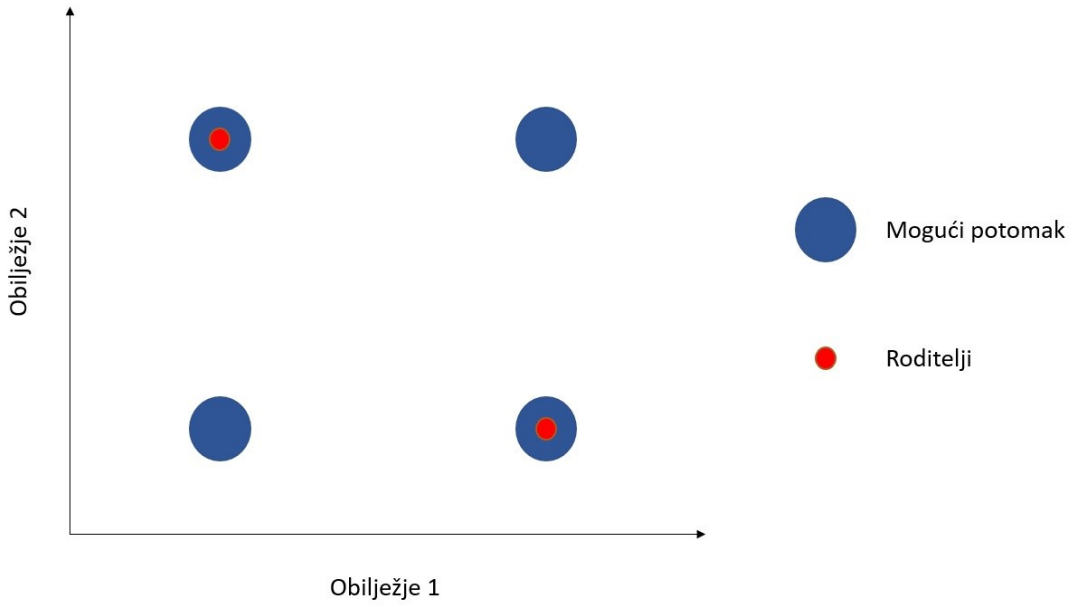
1.2.4. Križanje

Nakon obavljenog postupka selekcije, sljedeći korak evolucijskog algoritma je križanje odabranih jedinki. Smisao križanja počiva na ideji da novonastalo rješenje, odnosno

kromosom, može pokazati bolje karakteristike od oba roditelja ako pronađe bolju kombinaciju njihovih karakteristika. Do križanja između selektiranih roditeljskih kromosoma dolazi pod utjecajem parametra p_c kojeg nazivamo vjerojatnost križanja. U nastavku će biti opisane najčešće korištene metode križanja.

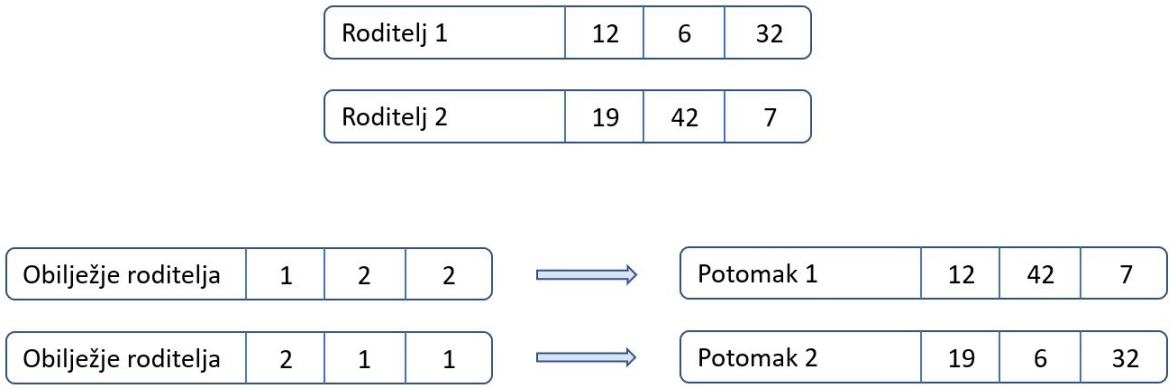
1.2.4.1. Diskretna rekombinacija

Diskretna rekombinacija (*eng. Discrete recombination*) [13] može se provoditi sa svakom populacijom bez obzira na odabranu metodu kodiranja, odnosno bez obzira na koji su način obilježja rješenja prikazana unutar genotipskog prostora. Kod diskretne rekombinacije se za svaku poziciju unutar kromosoma prepisuje se vrijednost jednog od roditelja s jednakom vjerojatnošću. Diskretna rekombinacija generira rubove hiperkocke kao potencijalna rješenja kao što se može vidjeti na slici 10.



Slika 10. Geometrijski učinak diskretne rekombinacije (2 obilježja)

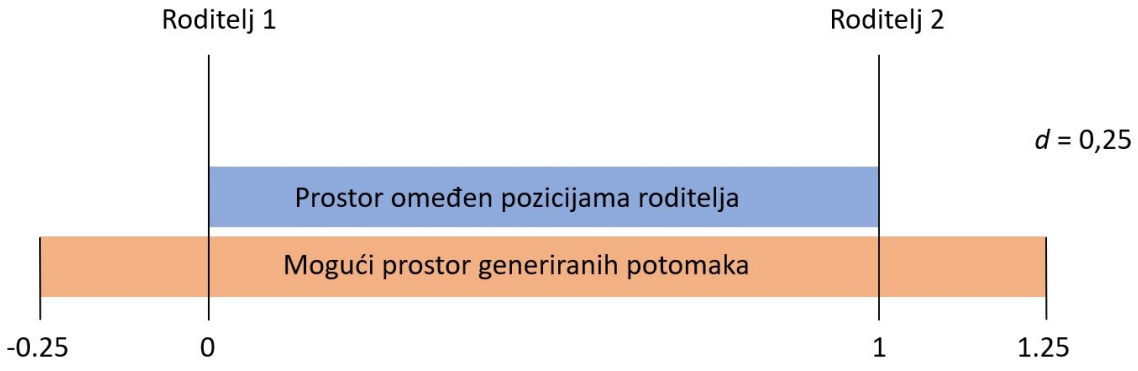
Primjer diskretne rekombinacije s 3 obilježja prikazan je na slici 11.



Slika 11. Primjer diskretne rekombinacije

1.2.4.2. Intermedijarna rekombinacija

Intermedijarna rekombinacija (*eng. Intermediate recombination*) [13] koristi se isključivo pri kodiranju s realnim vrijednostima. Kod ove metode se vrijednosti obilježja potomaka uzimaju iz okolnog prostora varijabli roditelja. Dimenzije hiperkocke unutar koje se potomci mogu pronaći definirana je parametrom *d.* Ako odaberemo $d = 0$, prostor unutar kojeg će se moći pronaći potomci bit će omeđen pozicijama roditelja. Takav tip intermedijarne rekombinacije nazivamo standardna intermedijarna rekombinacija. U svakom drugom slučaju, kada uzmemo $d > 0$, prostor unutar kojeg će se moći pronaći potomci prelazit će granice prostora kojeg omeđuju pozicije roditelja kao što se ilustrirano na slici 12.



Slika 12. Utjecaj parametra *d* na mogući prostor generiranih potomaka

Izračun vrijednosti svakog obilježja potomka korištenjem intermedijarne rekombinacije provodi se na sljedeći način:

$$Var_i^{potomak} = Var_i^{roditelj_1} \cdot a_i + Var_i^{roditelj_2} \cdot (1 - a_i),$$
$$i \in (1,2,\dots,N_{var}), a_i \in [-d, d+1]. \quad \textbf{(2)}$$

Gdje su:

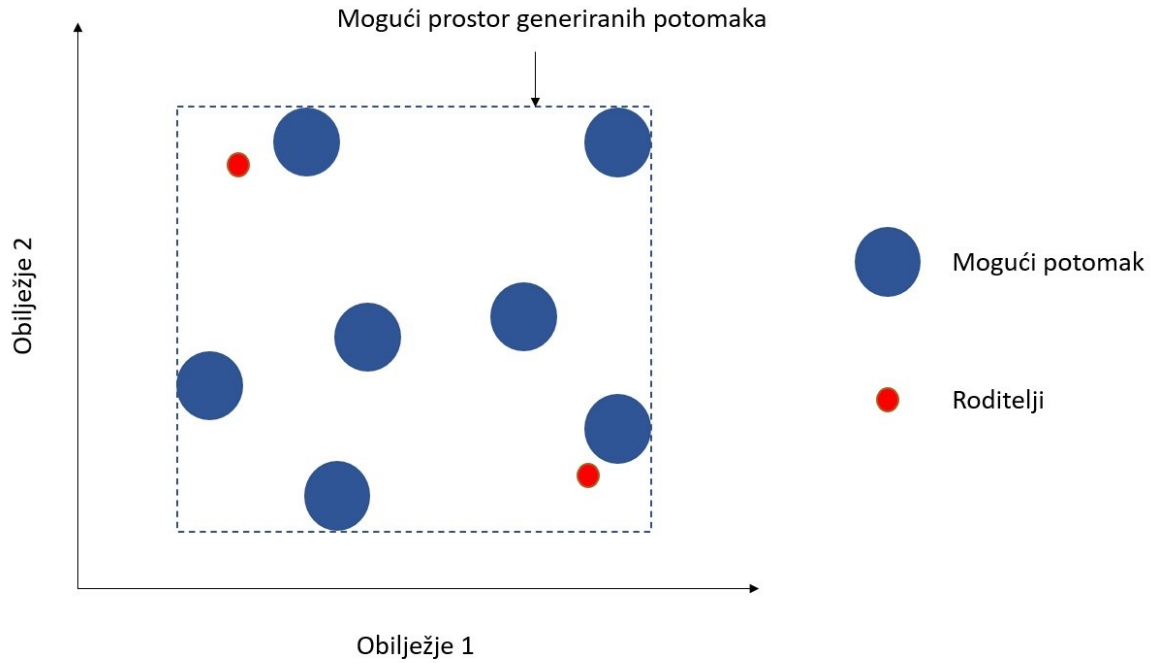
$Var_i^{potomak}$ vrijednost *i*-te varijable potomka

$Var_i^{roditelj_1}$ vrijednost *i*-te varijable prvog roditelja

a_i faktor skaliranja za *i*-tu varijablu

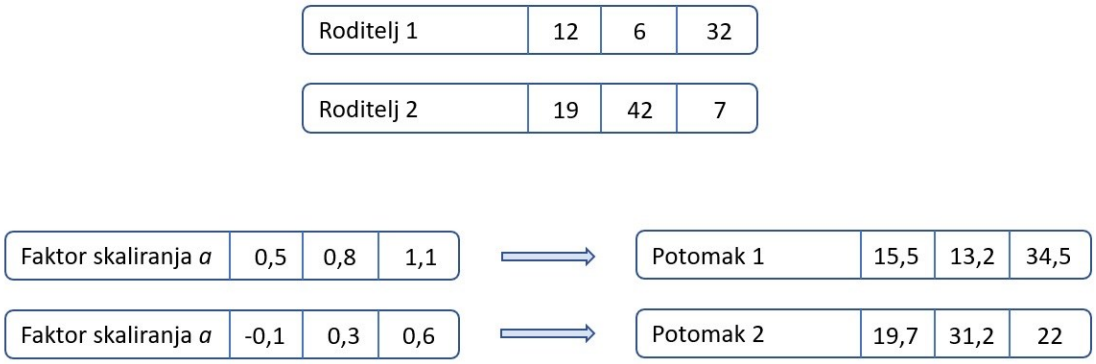
N_{var} ukupan broj varijabli

Geometrijski učinak na prostornu raspoređenost potomaka prikazan je na slici 13.



Slika 13. Geometrijski učinak intermedijarne rekombinacije (2 obilježja)

Primjer diskretne rekombinacije s 3 obilježja prikazan je na slici 14.



Slika 14. Primjer intermedijarne rekombinacije

1.2.4.3. Linijska rekombinacija

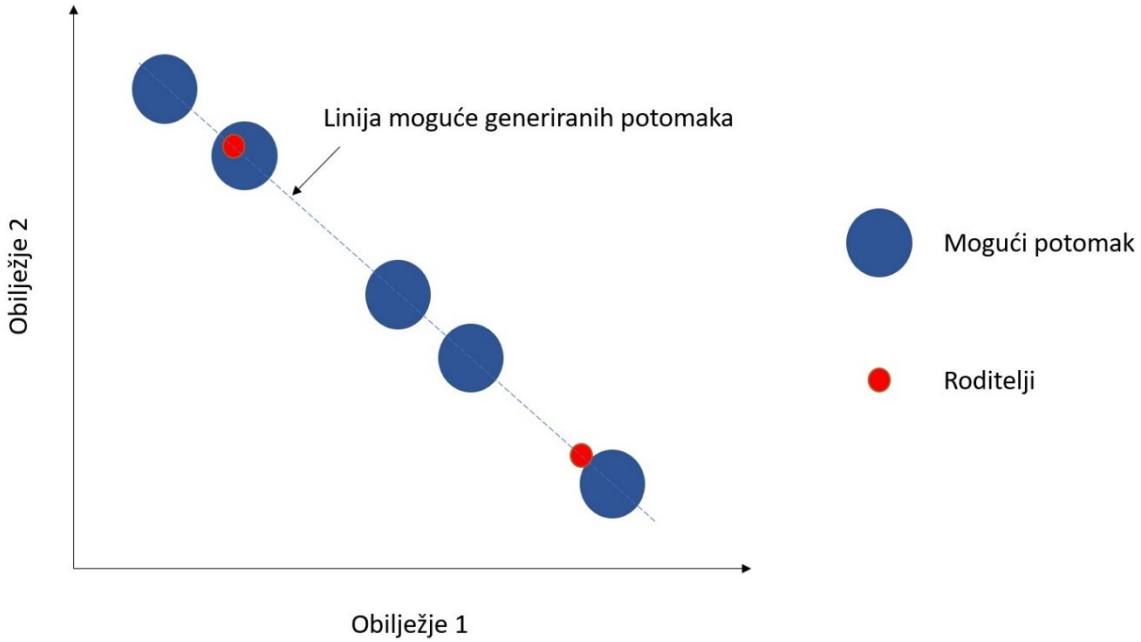
Razlika između linijske rekombinacije (*eng. Line recombination*) [13] i intermedijarne rekombinacije je što kod linijske rekombinacije faktor a poprima istu vrijednost za sve varijable roditelja. Tako će kod linijske rekombinacije izračun vrijednosti svakog obilježja potomka imati sljedeći oblik:

$$Var_i^{potomak} = Var_i^{roditelj_1} \cdot a + Var_i^{roditelj_2} \cdot (1 - a),$$
$$i \in (1,2, \dots, N_{var}), a \in [-d, d+1]. \quad \textbf{(3)}$$

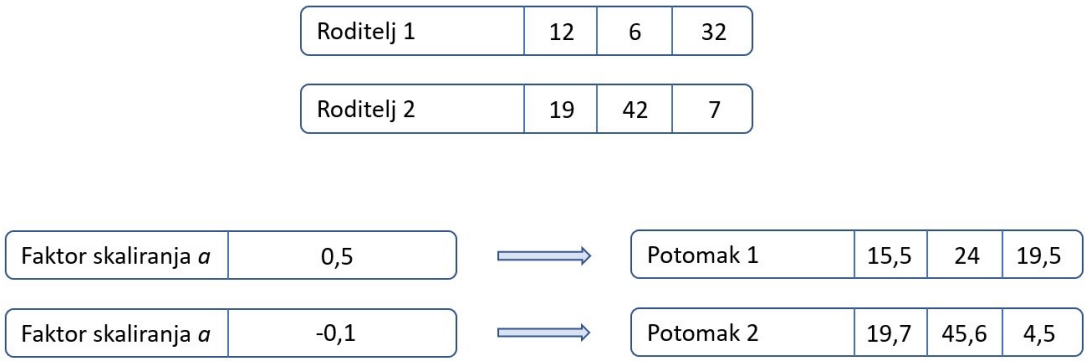
Gdje je:

a faktor skaliranja

Geometrijski učinak na prostornu raspoređenost potomaka za linijsku rekombinaciju prikazan je na slici 15. Ovdje se može vidjeti da će se svi nastali potomci biti pozicionirani na liniji koja spaja roditelje po čemu je ovaj tip križanja dobio naziv. Na slici 16 je prikazan primjer izvođenja linijske rekombinacije.



Slika 15. Geometrijski učinak linijske rekombinacije (2 obilježja)

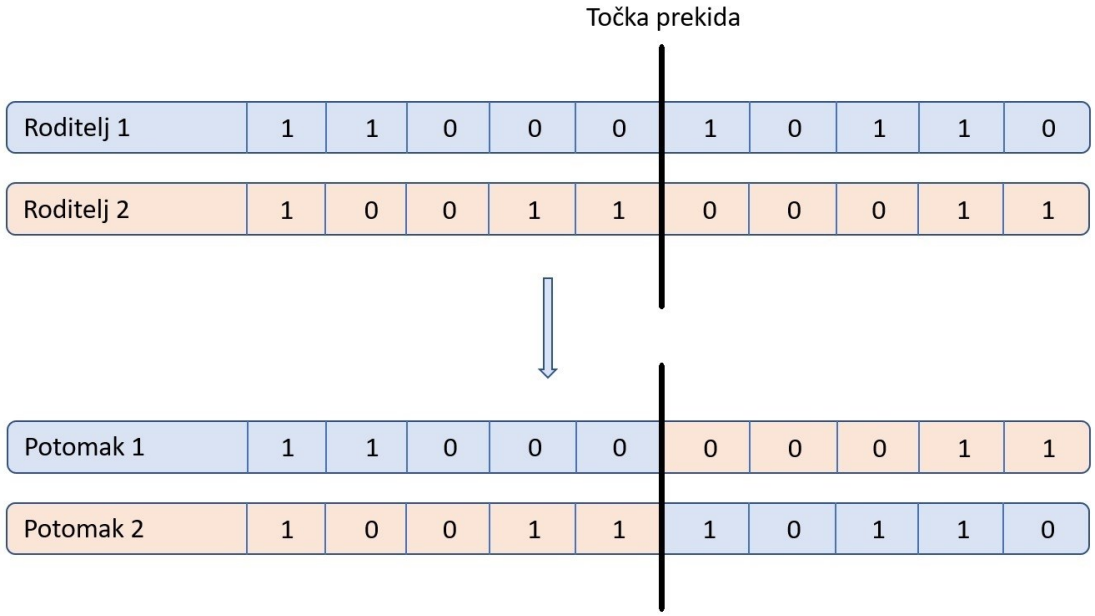


Slika 16. Primjer linijske rekombinacije

1.2.4.4. Križanje u jednoj točki/dvije točke/više točaka

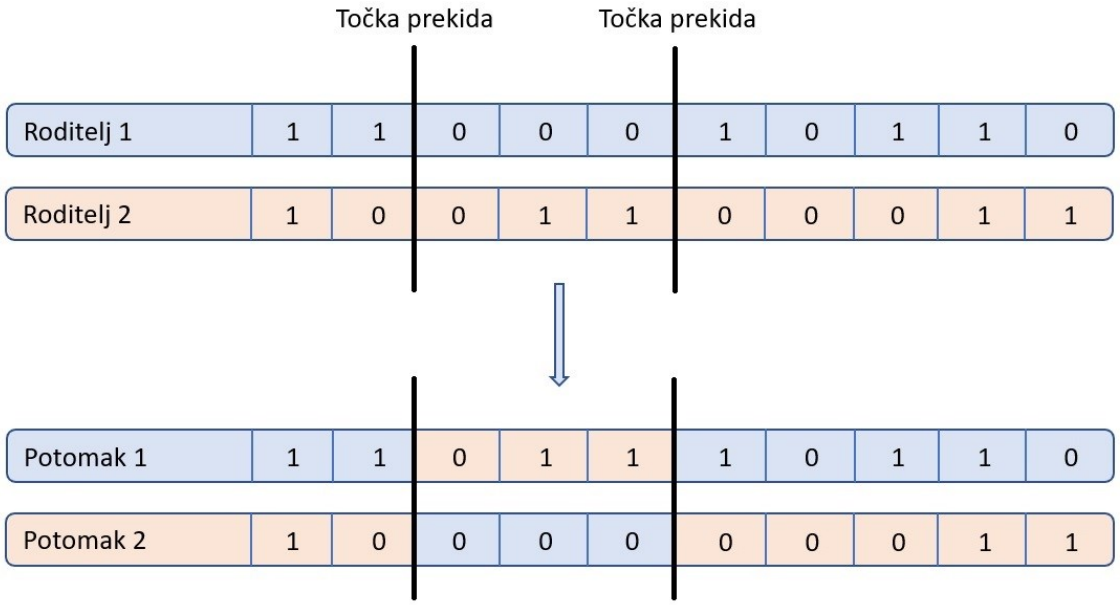
Križanje u točki (*eng. Point crossover*) [14] najčešće je korišteni i najjednostavniji oblik križanja. Ova metoda uglavnom se koristi s binarno kodiranim nizovima. Kod križanja u jednoj točki po slučaju se odabire točka prekida, te se zatim kombiniraju odgovarajući dijelovi roditelja tako da se prije točke prekida u kromosom potomka prepiše dio kromosoma jednog

roditelja, a nakon točke prekida dio kromosoma drugog roditelja. Primjer križanja u jednoj točki prikazan je na slici 17.

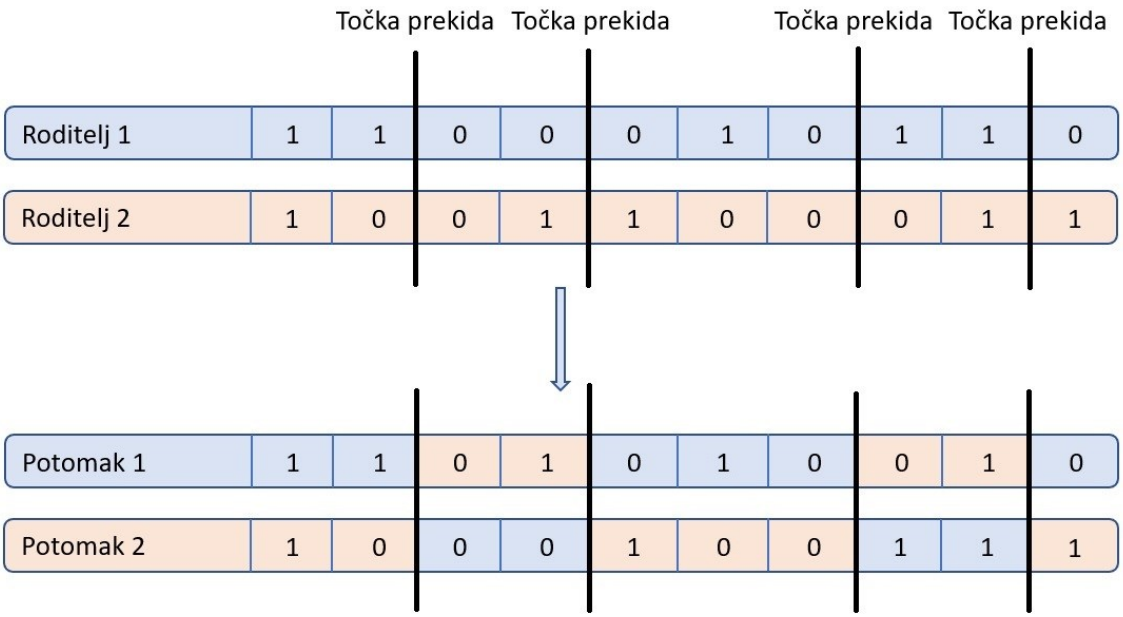


Slika 17. Primjer križanja u jednoj točki

U slučaju križanja u dvije ili više točaka, također se po slučaju odabiru točke prekida, a kromosomi potomaka sastoje se od naizmjenično prepisanih dijelova kromosoma roditelja kao što je prikazano na slikama 18 i 19.



Slika 18. Primjer križanja u dvije točke

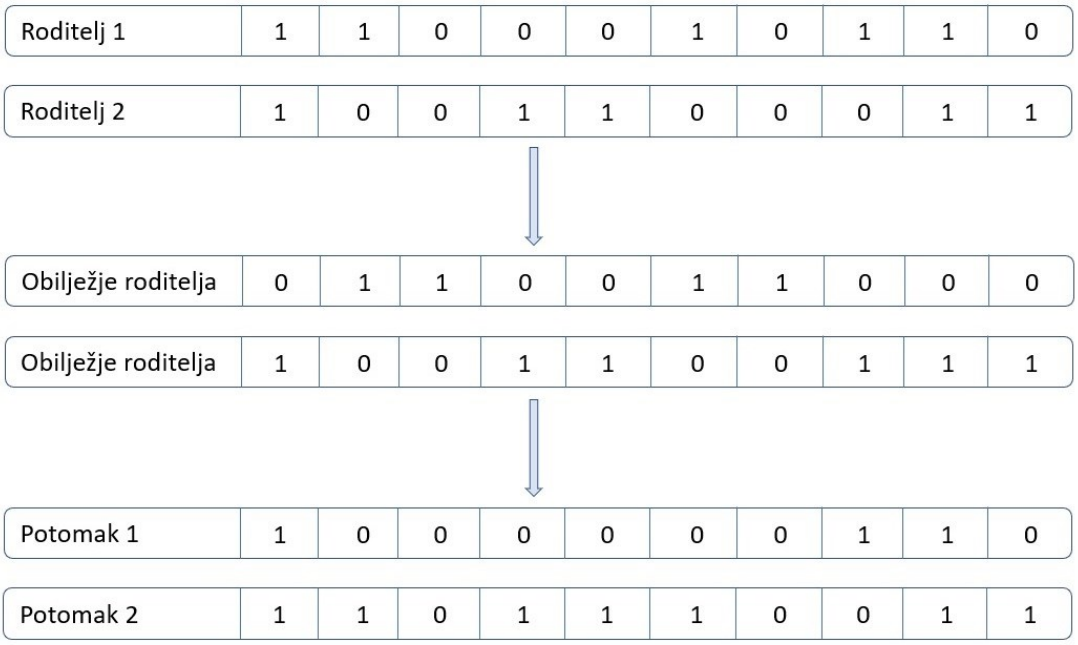


Slika 19. Primjer križanja u više točaka

Ideja križanja u više točaka zasnovana je na hipotezi da dijelovi kromosoma roditelja koji najviše doprinose dobroti ne moraju nužno biti povezani unutar njihovih kromosoma [15]. Nadalje, čini se da disruptivna priroda križanja u više točaka potiče širenje prostora pretrage, umjesto favoriziranja konvergencije oko pojedinaca izražene dobrote pronađenih unutar prvih nekoliko generacija što u konačnici doprinosi robusnosti algoritma [16].

1.2.4.5. Jednoliko križanje

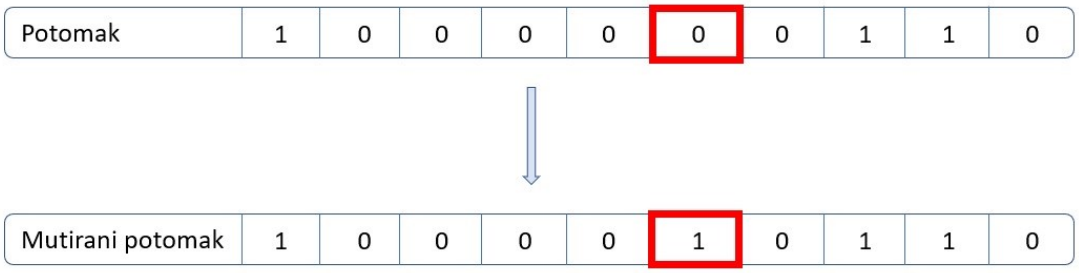
Jednoliko križanje (*eng. Uniform crossover*) [17] ekvivalentno je diskretnoj rekombinaciji, ali se koristi isključivo s binarno kodiranim nizovima. Za svako obilježje unutar kromosoma potomka, po slučaju se odabire roditelj čija će se vrijednost istog obilježja prepisati s jednakom vjerojatnošću. Istraživanja su pokazala da se s ovom metodom križanja uglavnom mogu postići bolji rezultati ako se vjerojatnost zamijene bitova kontrolira uvođenjem dodatnog parametra [18]. Primjer jednolikog križanja prikazan je na slici 20.



Slika 20. Primjer jednolikog križanja

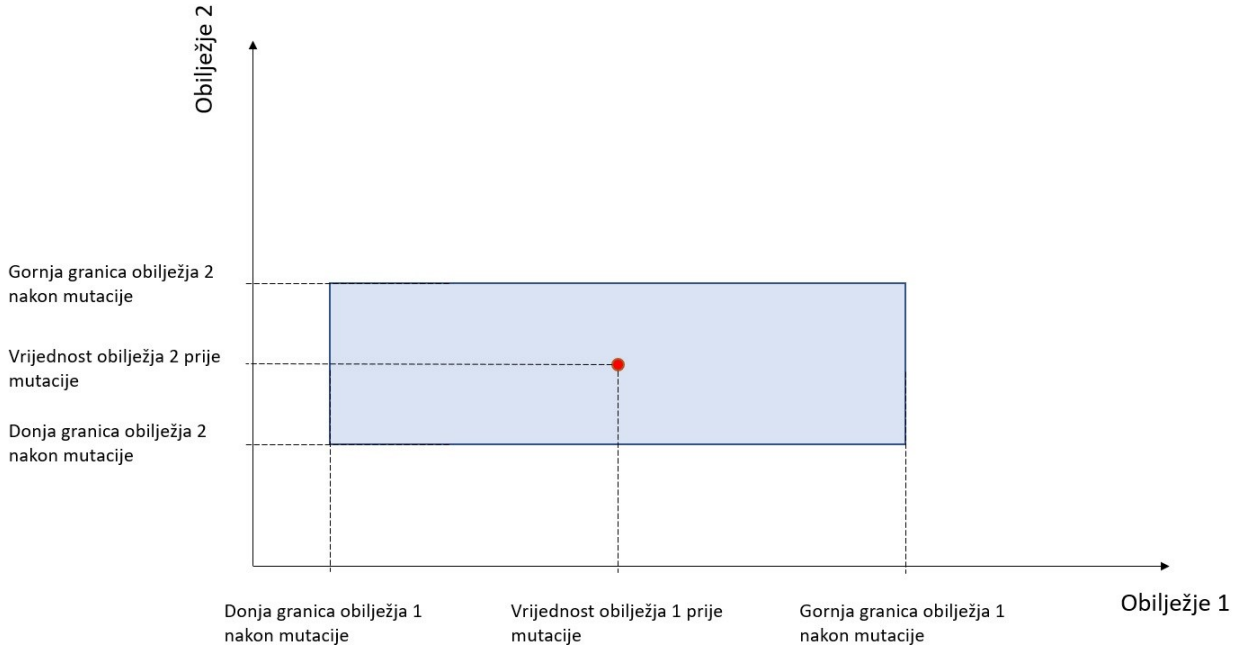
1.2.5. Mutacija

Mutacija se primjenjuje kako bi se održala genetska raznolikost između populacija. Uglavnom se provodi nakon križanja, međutim u specifičnim tipovima evolucijskih algoritama može se provoditi i prije križanja. Pojedinci iz populacije na kojima se provodi postupak mutacije odabiru se po slučaju, a vjerojatnost provođenja mutacije određena je parametrom p_m kojeg nazivamo vjerojatnost mutacije. Uvođenjem malih varijacija u kromosome potomaka može se lako proširiti prostor pretrage. Naime, ponekad se može dogoditi da prostor pretrage bude ograničen zbog odabrane metode križanja i/ili metode generiranja inicijalne populacije kao što je opisano u prethodnim poglavljima. Uz zadani parametar p_m, potrebno je definirati i sam operator mutacije uz zadani interval mogućih promjena na varijablama. U slučaju binarno kodiranih nizova, mutacija se može provoditi na određenom mjestu unutar kromosoma zamjenom bitova (obzirom da svaka varijabla binarno kodiranih kromosoma ima samo dva moguća stanja). Primjer mutacije binarno kodiranog niza zamjenom bitova prikazan je na slici 21.



Slika 21. Primjer mutacije binarnog niza zamjenom bitova

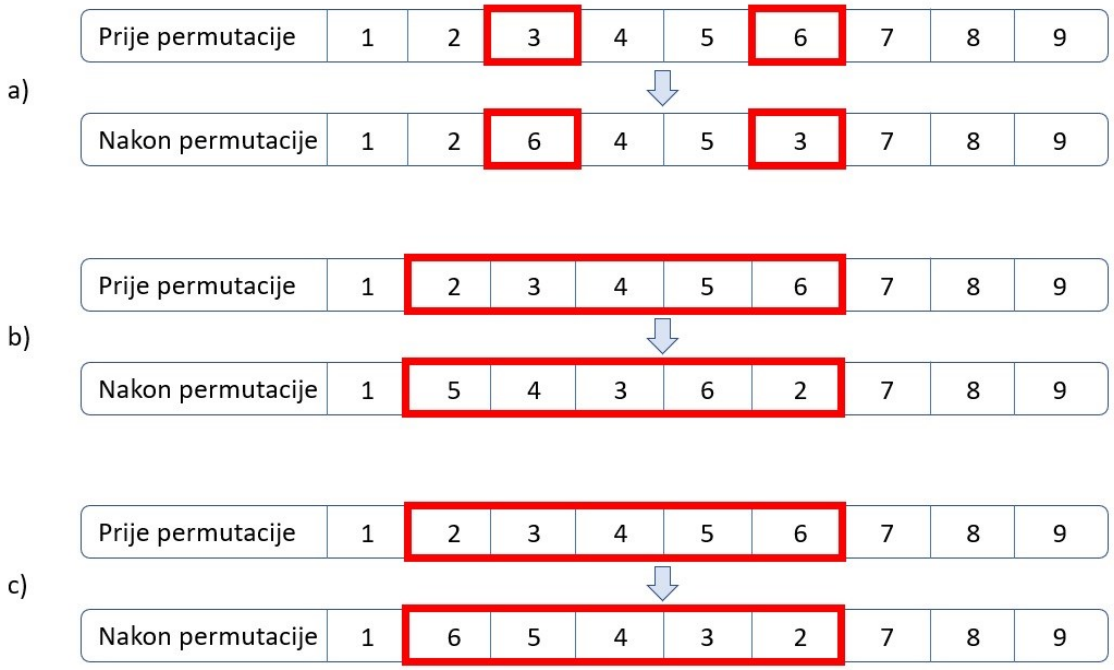
Kao što je prethodno spomenuto, prilikom izvođenja mutacije, potrebno je voditi računa o intervalu unutar kojeg mutacija može djelovati na inicijalnu vrijednost. Naime, ako interval mutacije nije pravilno definiran, moguće je dobiti rješenja koja kod određenih problema nisu valjana, stoga je prilikom definiranja operatora mutacije potrebno poznavati granice unutar kojih se dobivena rješenja smatraju validnima. Tako se prilikom provođenja mutacije na kromosomima kodiranim realnim vrijednostima treba definirati donja i gornja granica unutar kojih se očekuje dobivena vrijednost. Granice mogu biti jednako definirane za sve varijable unutar kromosoma, ali mogu biti i specifično definirane za svako zasebno obilježje kao što je prikazano na slici 22.



Slika 22. Postavljanje graničnih vrijednosti za specifična obilježja prilikom mutacije

Spomenute granice također mogu biti tako definirane da zavise o poziciji rješenja prije provođenja mutacije. Takav pristup bit će detaljnije objašnjen u nastavku rada.

Moguće je također provoditi permutaciju kod koje se zamjenjuju vrijednosti nasumično odabranih parametara. Primjeri najčešće korištenih oblika permutacije prikazani su na slici 23.



Slika 23. Primjeri permutacije: a) zamjena parametara, b) nasumično miješanje skupa parametara, c) inverzija skupa parametara

1.2.6. Elitizam

Kako bi se osigurala konvergencija evolucijskog algoritma, često se provodi elitizam. U pravilu elitizam označava postupak inkluzije pojedinaca najviše izražene dobrote u sljedeću generaciju bez provođenja operatora križanja i mutacije. Tako se nakon izračuna dobrote svakog pojedinca neke generacije, najbolja rješenja spremaju u memoriju kako bi se nakon križanja i mutacije ubacili u novonastalu populaciju. U praksi, elitizam omogućava daljnju pretragu okolnog prostora lokalnog ili globalnog optimuma.

S druge strane, implementacija elitizma ima i negativa učinak na proces evolucije. S obzirom na to da propagira već evaluirana rješenja u sljedeće generacije, smanjuje se vjerojatnost pronalaska drugih rješenja koja se ne nalaze u okolnom prostoru elitnih članova, a iskazuju

jednako visok ili još viši stupanj dobrote. Dakle, elitizam neizbježno smanjuje raznolikost populacije. Iz tog se razloga elitizam uglavnom provodi isključivo za jedno najbolje rješenje ili nekolicinu najboljih rješenja populacije. Bitno je za napomenuti da ne postoji univerzalno optimalan način provođenja elitizma već da on zavisi o samom problemu koji se rješava, ali i o strukturi evolucijskog algoritma koji se koristi. U konačnici, elitizam nije nužno implementirati u svaki evolucijski algoritam, ali se najčešće upotrebljava baš kako bi se osigurala konvergencija.

1.2.7. Uvjet zaustavljanja

Postupci koji su opisani u prethodnim poglavljima ponavljaju se za svaku generaciju rješenja tako što se na svakoj dobivenoj populaciji potomaka ponovno vrši izračun dobrote čime ulaze u postupak selekcije roditelja itd. sve dok se ne zadovolji određeni uvjet. Uvjet zaustavljanja definira sam korisnik prije pokretanja algoritma, a on može biti jedno od navedenog:

1. pronađeno je rješenje koje zadovoljava unaprijed postavljenje uvjete
2. izvršen je unaprijed zadani broj iteracija algoritma.

Uvjet zaustavljanja najčešće reflektira samu prirodu problema. Ako se rješavaju standardne ispitne (*eng. benchmark*) funkcije kod kojih je poznato optimalno rješenje s ciljem ocjenjivanja kvalitete samog algoritma, algoritam se zaustavlja u trenutku kada je pronađeno traženo rješenje s obzirom na to da daljnje izvođenje algoritma ne bi imalo nikakav doprinos. Isti uvjet zaustavljanja također se može postaviti ako nam je potrebno rješenje koje zadovoljava određene kriterije. S druge strane, u slučaju da globalni optimum nije poznat, a postoji vremensko ograničenje unutar kojeg se mora pronaći što bolje rješenje, postavlja se broj iteracija koje će algoritam izvršiti s nadom da će unutar istog broja iteracija zadovoljavajuće rješenje biti pronađeno.

2. POSTAVLJANJE HIPERPARAMETARA

Postavljanje odgovarajućih vrijednosti hiperparametara za evolucijske algoritme predstavlja jedan je od velikih izazova na području evolucijskog računarstva. Međutim, vrlo se malo truda ulaže u proučavanje učinka odabranih hiperparametara na samu izvedbu evolucijskih algoritama. U praksi se vrijednosti hiperparametara uglavnom odabiru konvencijama (npr. stopa mutacije treba biti niska) ili se odabiru na temelju eksperimenata koji ne nude dovoljno jasnu predodžbu o njihovom značaju. Dakle, postoji upečatljiv rascjep između široko priznate važnosti odabranih hiperparametara i pronalaženju istih u praksi.

Problem postavljanja hiperparametara evolucijskih algoritama dijelimo na:

1. upravljanje hiperparametara
2. podešavanje hiperparametara.

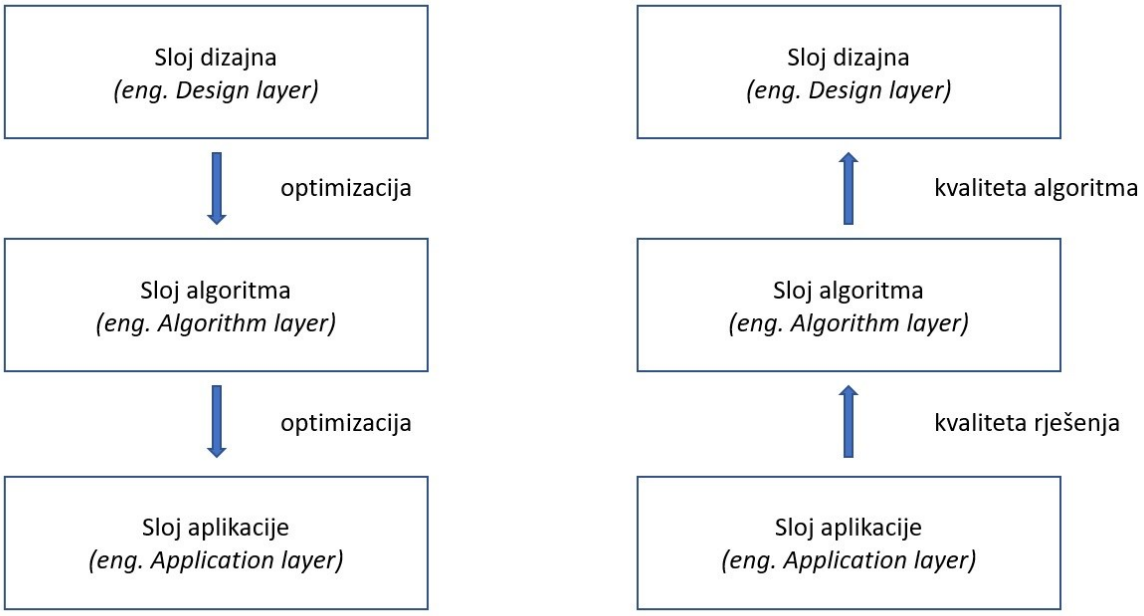
U slučaju upravljanja hiperparametrima, vrijednosti hiperparametara se mijenjaju tijekom samog izvođenja evolucijskog algoritma. U ovom je slučaju potrebno zadati početne vrijednosti hiperparametara i odgovarajuće strategije upravljanja koje dijelimo na:

1. determinističke
2. adaptivne
3. samoadaptivne.

Podešavanje hiperparametara predstavlja jednostavniji problem s obzirom na to da se njihove vrijednosti ne mijenjaju tijekom izvođenja cijelog procesa evolucijskog algoritma. Međutim, čak i samo podešavanje hiperparametara evolucijskog algoritma predstavlja težak zadatak zbog beskonačnog broja opcija i ograničenog znanja o učinku hiperparametara na korištenu strukturu evolucijskog algoritma pri rješavanju specifičnog zadatka.

2.1. Kontrolni i informacijski tok

A.E. Eiben and S.K. Smit daju detaljniji uvid u ovu problematiku promatrajući kontrolni i informacijski tok kroz 3 glavna sloja u prikazu hijerarhijske strukture kroz koju se provodi postavljanje hiperparametara [19].



Slika 24. Kontrolni i informacijski tok kroz 3 glavna sloja prilikom postavljanja hiperparametara

Kao što je ilustrirano na slici 24, cijela shema može biti podijeljena na dva optimizacijska problema. Donji sloj predstavlja sloj aplikacije (*eng. Application layer*), odnosno sam zadatak koji evolucijski algoritam rješava (npr. problem trgovačkog putnika). Evolucijski algoritam koji se koristi da bi pronašao rješenje zadanog problema predstavljen je srednjim slojem algoritma (*eng. Algorithm layer*). Gornji dio hijerarhijske strukture naziva se sloj dizajna (*eng. Design layer*) i unutar njega se nalazi metoda postavljanja hiperparametara. Unutar sloja dizajna, iterativno se pokušava pronaći najbolja kombinacija hiperparametara analogno sloju algoritma unutar kojeg se iterativno pokušava pronaći optimalno rješenje sloja aplikacije. Problem postavljanja hiperparametara ovdje je ilustriran prikazom informacijskog toka koji zauzima smjer suprotan kontrolnom toku. Naime, kvaliteta rješenja unutar sloja aplikacije zavisi o sloju algoritma, dok kvaliteta algoritma zavisi o sloju dizajna. Drugim riječima, da bismo došli do rješenja unutar sloja aplikacije moramo imati definirane slojeve iznad njega, a

da bismo navedene slojeve optimalno definirali potrebne su nam povratne informacije koje dobivamo iz aplikacijskog sloja tek nakon izvedbe već definiranog algoritma.

Bitno je napomenuti da sloj dizajna uz odabir hiperparametara evolucijskog algoritma, sadrži i odabir metode generiranja inicijalne populacije, metodu selekcije, metodu križanja i metodu mutacije koje su obrađene u prethodnom poglavlju. Za razliku od hiperparametara koje još nazivamo kvantitativnim parametrima evolucijskog algoritma, navedene metode nazivamo kvalitativnim parametrima s obzirom na to da ne poprimaju određenu vrijednost već diktiraju vođenje specifičnih postupaka unutar evolucijskog algoritma. Označimo li kvalitativne parametre s $Q_{1,\dots,}Q_{m,}$ a kvantitativne parametre s $R_{1,\dots,}R_{n,}$ problem optimizacije evolucijskog algoritma može biti prikazan u sljedećem obliku:

$$S = Q_1 \times \cdots \times Q_m \times R_1 \times \cdots \times R_n. \tag{4}$$

Gdje su:

S prostor svih parametara evolucijskog algoritma

m broj kvalitativnih parametara

n broj kvantitativnih parametara

Optimizacija evolucijskog algoritma obično prvo podrazumijeva strukturalnu optimizaciju koja predstavlja odabir kvalitativnih parametara:

$$S_s = Q_1 \times \cdots \times Q_m. \tag{5}$$

Nakon čega slijedi postavljanje hiperparametara za odabranu strukturu:

$$S_h = R_1 \times \cdots \times R_n. \tag{6}$$

Strukturalna optimizacija usko je vezana uz tip zadatka kojeg evolucijski algoritam rješava, a usporedba različitih struktura evolucijskih algoritama za rješavanje istog tipa zadatka uvijek podrazumijeva i postavljene hiperparametre navedenih struktura. Tako je moguće da jedna struktura evolucijskog algoritma pronađe bolje rješenje od druge strukture, ne kao rezultat

odabira kvalitativnih parametara, već kao rezultat bolje postavljenih hiperparametara. Isto tako, moguće je da evolucijski algoritam s dobro postavljenim hiperparametrima pronađe lošije rješenje od evolucijskog algoritma s loše postavljenim hiperparametrima, isključivo kao rezultat strukture drugog algoritma koja više odgovara danom zadatku. Iz tog je razloga bitno istaknuti vezu između kvalitativnih i kvantitativnih parametara. U nastavku poglavlja, pretpostavit ćemo da je određena struktura evolucijskog algoritma odabrana i koncentrirati se isključivo na postavljanje hiperparametara.

2.2. Evaluacija postavljenih hiperparametara

Općenito, postoje dvije mjere koje se koriste prilikom evaluacije postavljenih hiperparametara. Jedna mjera uzima u obzir kvalitetu pronađenog rješenja, dok druga mjera uzima u obzir brzinu izvođenja samog algoritma koju možemo iskazati na više načina, a najčešći su:

1. broj evaluacija dobrote
2. stvarno vrijeme izvođenja
3. procesorsko vrijeme izvođenja.

Kako bi se odradila evaluacija postavljenih hiperparametara, ove se dvije mjere koriste u različitim kombinacijama pa su tako najčešći slučajevi:

1. uz dano maksimalno vrijeme izvođenja algoritma, evaluacija postavljenih hiperparametara vrši se na temelju dobrote koju iskazuje najbolje pronađeno rješenje
2. uz minimalnu zadanu vrijednost dobrote, evaluacija postavljenih hiperparametara vrši se na temelju vremena koje je potrebno da se takvo rješenje pronađe
3. uz dano maksimalno vrijeme izvođenja algoritma i minimalnu zadanu vrijednost dobrote, evaluacija postavljenih hiperparametara vrši se na temelju Booleanove notacije uspjeha:

   algoritam je uspješno odradio zadatak ako je rješenje pronađeno, a u suprotnom se provođenje algoritma smatra neuspjehom.

Naravno, obzirom na stohastičku prirodu evolucijskih algoritama, za evaluaciju postavljenih hiperparametara je potrebno svaki tip ispitivanja provesti više puta. Nakon određenog broja

ponavljanja dolazimo do sljedećih vrijednosti koje daju uvid u kvalitetu postavljenih hiperparametara koje vežemo uz prethodno spomenute slučajeve:

1. prosjek najviše iskazane dobrote
2. prosječno vrijeme potrebno za pronalazak rješenja
3. stopa uspjeha.

Prilikom postavljanja hiperparametara evolucijskog algoritma, moguće je optimizaciju provesti s gledišta jedne od spomenutih mjera ili kombinacije više njih.

U najjednostavnijem slučaju, postavljanje hiperparametara se odvija na evolucijskom algoritmu čiji je zadatak riješiti jedan specifičan problem. U tom slučaju dolazimo do tzv. specijaliziranog evolucijskog algoritma (*eng. specialist evolutionary algorithm*), odnosno evolucijskog algoritma čija je zadaća konzistentno pružati dobra rješenja za jedan konkretan problem. U ovom slučaju nije garantirano da će postavljeni hiperparametri rezultirati dobrim rješenjem ako se istom evolucijskom algoritmu postavi problem koji se razlikuje od onoga putem kojeg su hiperparametri postavljeni. Stručnjaci su često u potrazi za hiperparametrima tzv. generaliziranih evolucijskih algoritama (*eng. generalist evolutionary algorithm*). Ovaj oblik evolucijskog algoritma mora pružati robusnost u smislu rješavanja različitih tipova zadataka. Međutim, dokazano je da kombinacija hiperparametara koja zadovoljava uvjet pronalaska dobrih rješenja za svaki mogući tip zadatka ne postoji [20], a pronalazak takvih hiperparametara za već male skupove različitih problema pokazao se vrlo kompleksnim, a ponekad također nedostižnim.

Postupke postavljanja hiperparametara također dijelimo u dvije kategorije:

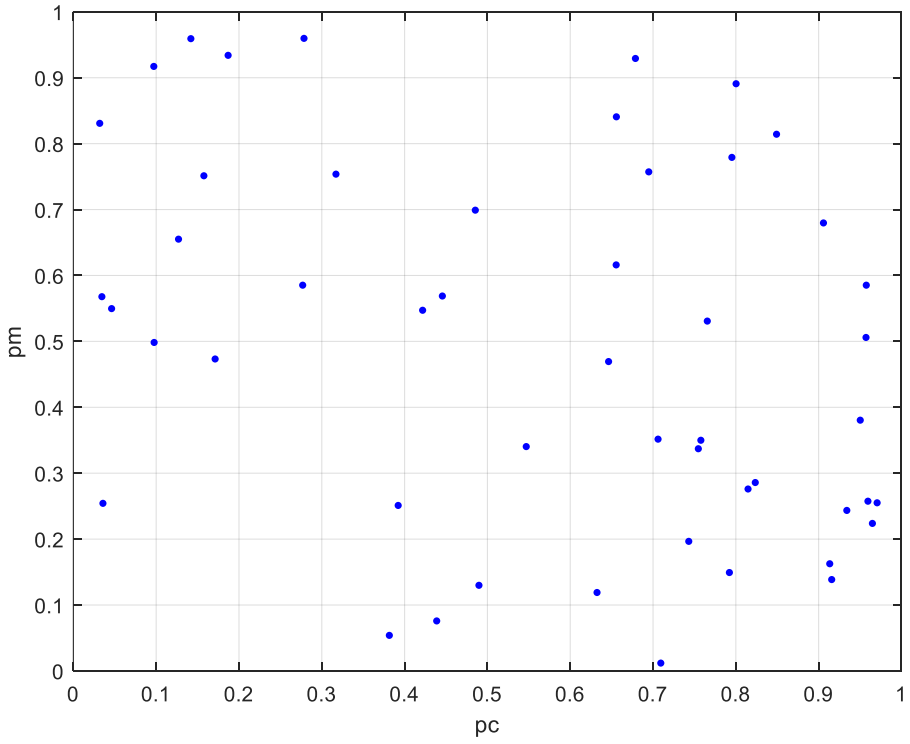
1. iterativni postupci
2. ne-iterativni postupci.

U slučaju ne-iterativnog postupka postavljanja hiperparametara, kombinacije hiperparametara koje se evaluiraju prilikom pretrage, generirane su na samom početku procesa. S druge strane, u slučaju iterativnog postupka, na početku pretrage se generira nekoliko kombinacija

vrijednosti hiperparametara, te se nakon njihove evaluacije generiraju nove kombinacije koje prolaze kroz isti postupak.

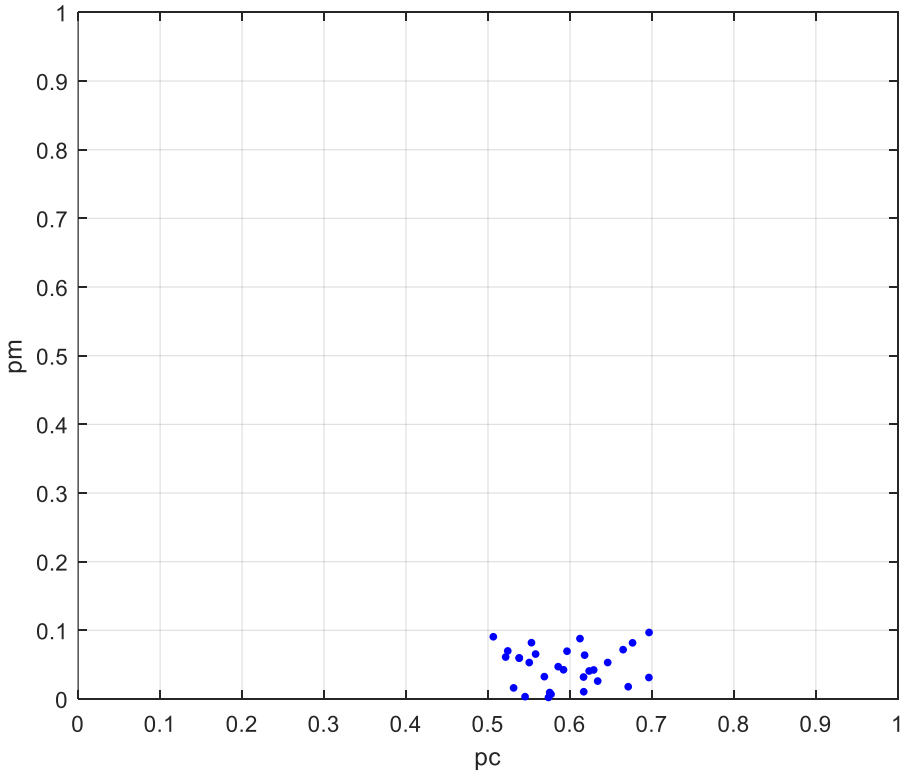
2.3. Najčešće korištene metode postavljanja hiperparametara

Kao što je prethodno spomenuto, postoje brojne metode koje se koriste prilikom postavljanja hiperparametara. Najjednostavnija od njih je nasumično generiranje kombinacija hiperparametara koja ulazi u skupinu ne-iterativnih postupaka. U početku se nasumično generira određen broj kombinacija traženih hiperparametara nakon čega se svaka od generiranih kombinacija primjenjuje na evolucijskom algoritmu. Svaka kombinacija se testira prethodno specificirani broj puta nakon čega slijedi evaluacija. Tip evaluacije zavisi o samoj prirodi problema koji evolucijski problem rješava kao što je već opisano u prethodnom poglavlju. Naposljetku se odabire najbolje evaluirana kombinacija hiperparametara. Ova metoda je ilustrirana na slici 25. Radi jednostavnije ilustracije svaka će metoda biti prikazana za postavljanje samo 2 hiperparametra (vjerojatnost mutacije i vjerojatnost križanja) premda je u praksi često potrebno pronaći veći broj hiperparametara.



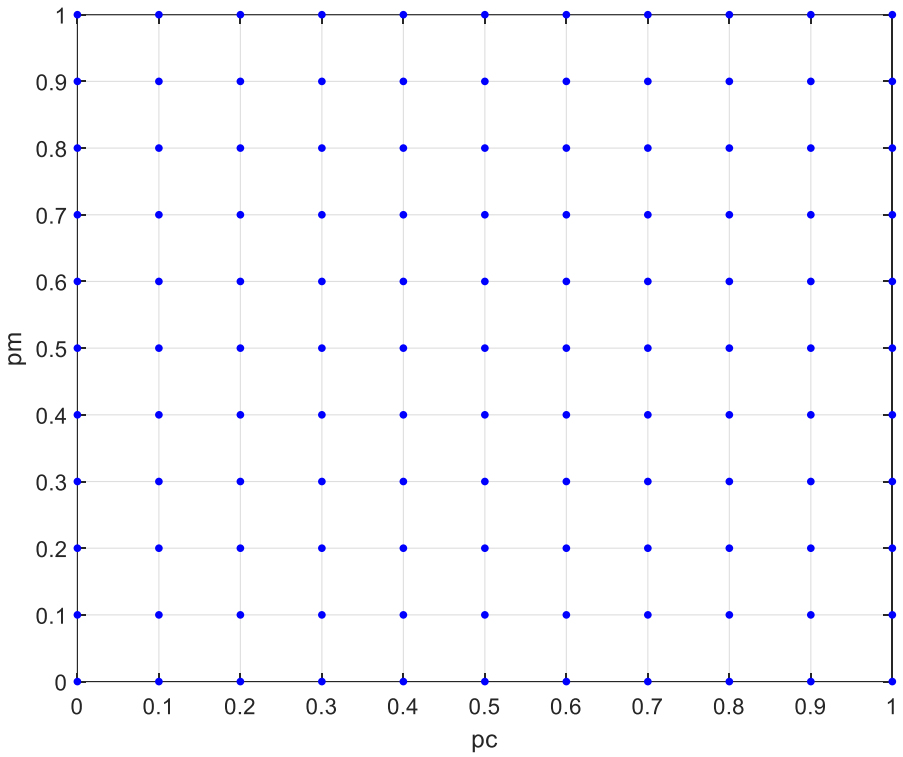
Slika 25. Primjer nasumične metode postavljanja hiperparametara

Sukladno generiranju inicijalne populacije evolucijskog algoritma, ako postoji određeno znanje o domeni, prostor pretrage vrijednosti hiperparametara također se može ograničiti. Primjerice, ako nasumičnim generiranjem pretražujemo prostor potencijalnih vrijednosti hiperparametara uz postavljene granične vrijednosti za svaki specifični hiperparametar, dobit ćemo pretragu koja nalikuje nasumično generiranim vrijednostima prikazanim na slici 26.

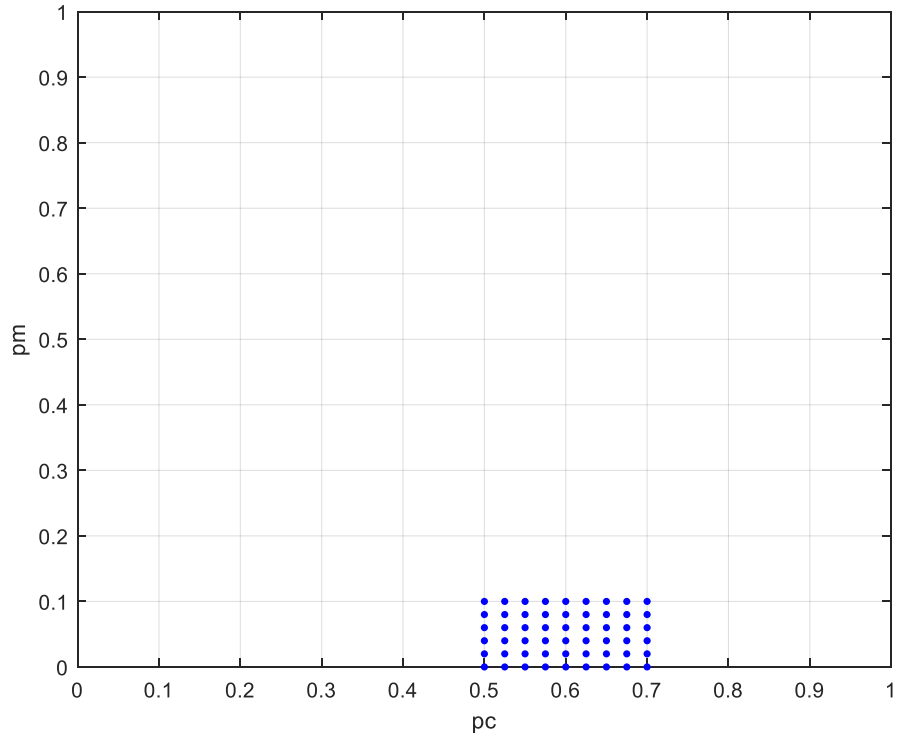


Slika 26. Primjer nasumične metode postavljanja hiperparametara uz distribuciju temeljenu na znanju

Druga metoda postavljanja hiperparametara analogna je mrežnoj inicijalizaciji populacije. Za svaki hiperparametar se odabire određeni korak s kojim se ispunjuje mreža kombinacija koje se testiraju i evaluiraju. Ovaj postupak također spada pod ne-iterativne metode pronalaska hiperparametara. Primjer primjene ove metode prikazan je na slikama 27 i 28.



Slika 27. Primjer mrežne metode postavljanja hiperparametara

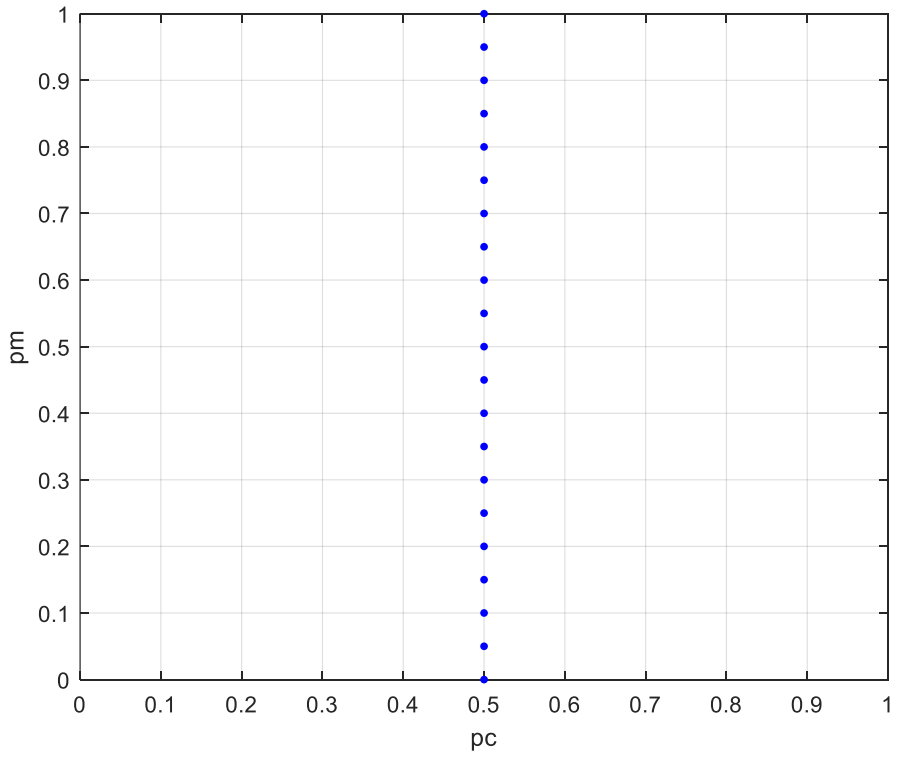


Slika 28. Primjer mrežne metode postavljanja hiperparametara uz distribuciju temeljenu na znanju

Problem sa spomenutim metodama je što njihovo izvođenje može dugo trajati. Kako bi se smanjilo vrijeme pretrage ponekad se koristi linijska metoda koja odrađuje pretragu parametar po parametar. Isprva se svi hiperparametri postavljaju na neku prethodno definiranu vrijednost nakon čega se jedan po jedan, uz korak koji se specifično zadaje za svakog od njih, istražuje njihovo djelovanje.

Tako bi se korištenje ove metode za pronalazak optimalnih vrijednosti vjerojatnosti križanja i mutacije moglo provesti na sljedeći način:

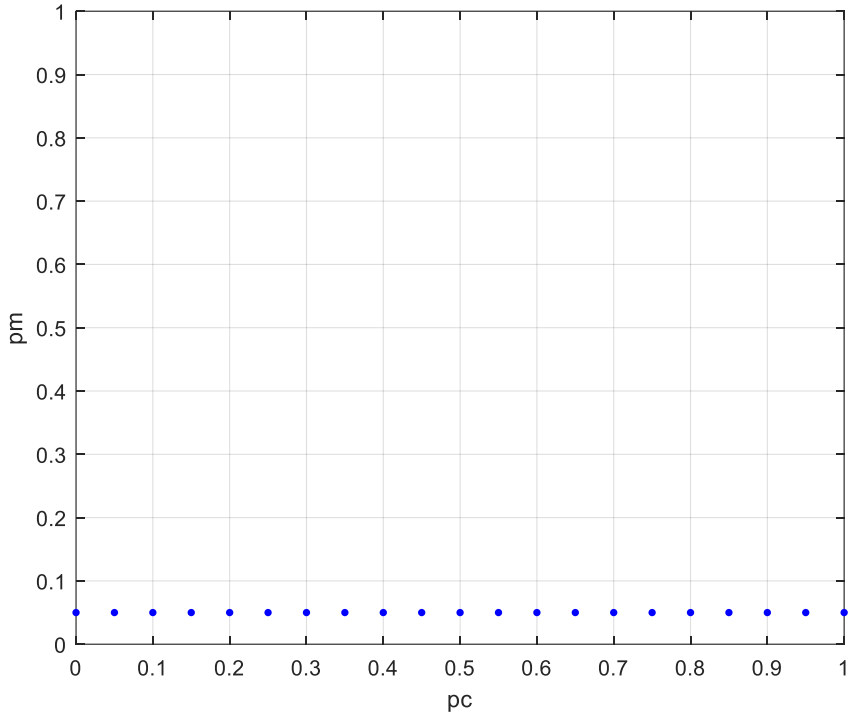
1. Za ispitivanje djelovanja vjerojatnosti mutacije, vjerojatnost križanja isprva postavljamo na vrijednost 0,5 te evaluiramo sve kombinacije prikazane na slici 29.



Slika 29. Primjer postavljanja prvog hiperparametra primjenom linijske metode postavljanja hiperparametara

2. Nakon evaluacije različitih vrijednosti vjerojatnosti mutacije uz zadanu vrijednost vjerojatnosti križanja 0,5 odabire se vjerojatnost mutacije koja je dala najbolje rezultate. Zatim se za tu vrijednost mutacije ponavlja isti postupak s različitim

vrijednostima vjerojatnosti križanja. U slučaju da je odabir vjerojatnosti mutacije od 0,05 u prethodnom koraku imao najbolji ishod, ispitivanje djelovanja vjerojatnosti križanja poprimilo bi oblik prikazan na slici 30.

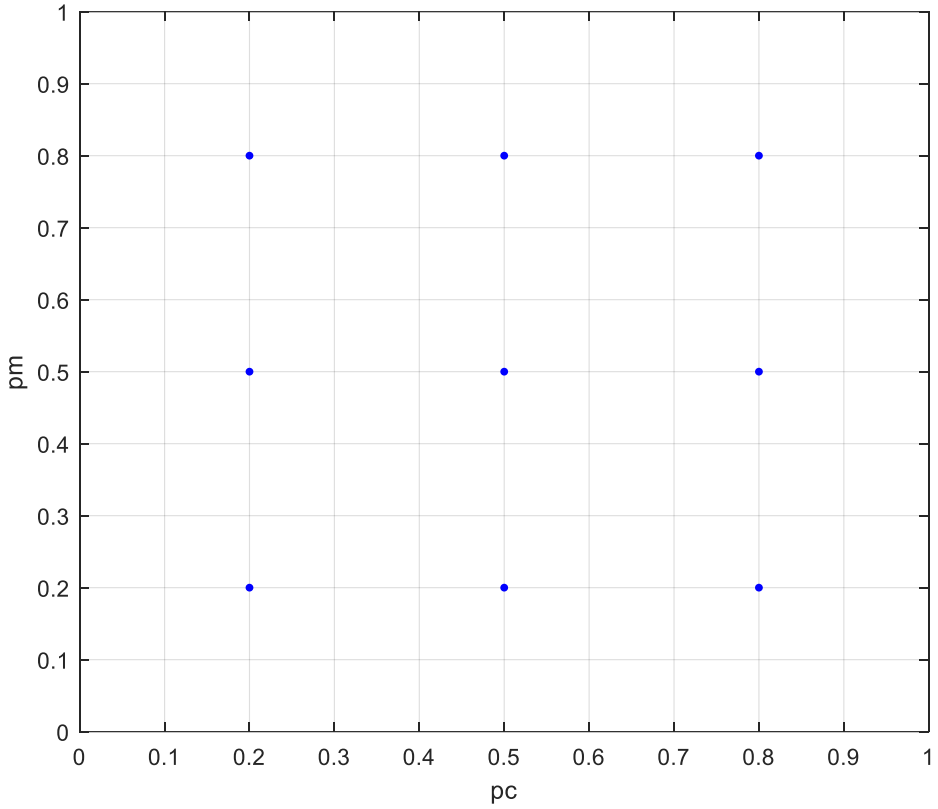


Slika 30. Primjer postavljanja drugog hiperparametra primjenom linijske metode postavljanja hiperparametara

3. Nakon evaluacije svih prikazanih kombinacija hiperparametara odabire se ona koja postiže najbolji rezultat. U slučaju postavljanja više od 2 hiperparametra, postupak bi se nastavio sve dok vrijednost svakog od njih nije postavljena.

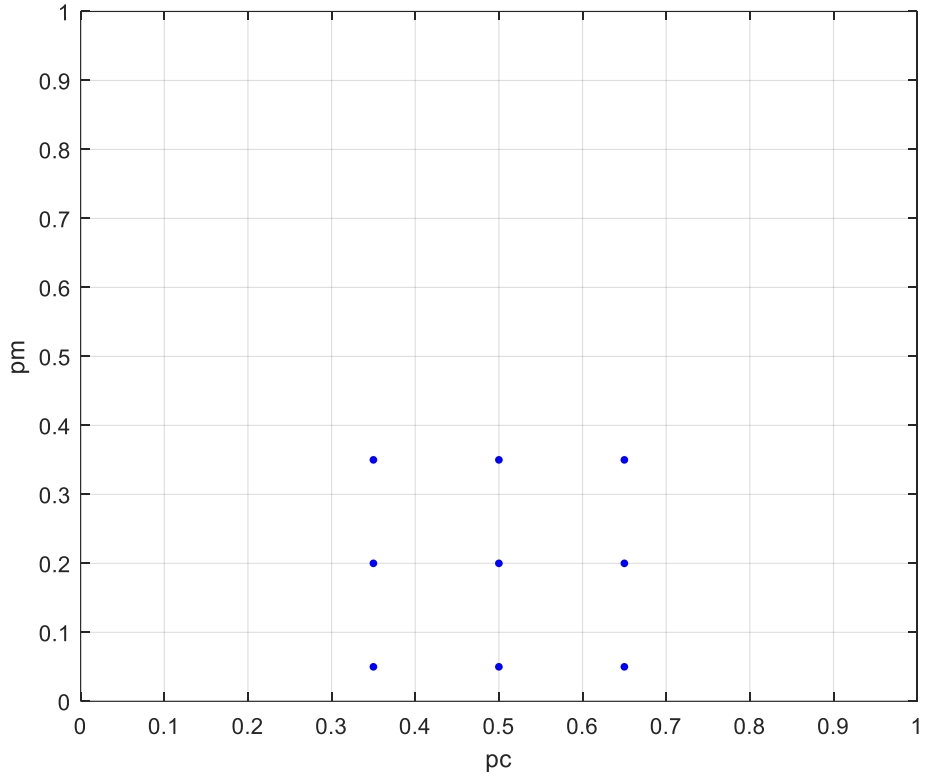
Ova metoda, kao i svaka druga, ne garantira pronalazak optimalnih vrijednosti hiperparametara. Veliki nedostatak ove metode je što najčešće konvergira području lokalnog optimuma, a razlog tome je zasebno ispitivanje djelovanja vrijednosti različitih hiperparametara. Za razliku od prethodne dvije metode, linijsku metodu postavljanja hiperparametara ubrajamo u iterativne postupke jer kombinacije hiperparametara koje se ispituju nisu poznate na samom početku procesa.

Kompas metoda slična je linijskoj te također predstavlja iterativnu metodu. Međutim, za razliku od linijske metode kod koje se ispituje djelovanje svakog hiperparametra zasebno, kod kompas metode se prostor pretrage dijeli na određen broj segmenata. Unutar svakog segmenta se uzima jedna kombinacija vrijednosti hiperparametara koja predstavlja taj segment kao što je prikazano na slici 31.



Slika 31. Primjer prvog koraka kompas metode postavljanja hiperparametara

Nakon evaluacije svih kombinacija hiperparametara, prostor pretrage se sužava na okolni prostor one kombinacije hiperparametara koja se pokazala najboljom u prethodnom koraku. Zatim se oko te točke ponavlja isti postupak kao što je prikazano na slici 32. Nakon svakog koraka se osim prostora pretrage smanjuje i udaljenost točaka koje se evaluiraju. Čitav postupak se može provoditi zadani broj puta ili dok se ne pronađe kombinacija hiperparametara koja zadovoljava prethodno definirane uvjete.



Slika 32. Primjer drugog koraka kompas metode postavljanja hiperparametara

Osim prikazanih metoda postoje još brojne druge metode koje se koriste kako bi se pronašle optimalne vrijednosti hiperparametara. Ponekad se u ovu svrhu koriste i metode koje potiču iz područja koja nisu striktno vezana uz evolucijsko računarstvo poput Taguchijeve metode ortogonalnih matrica [21] ili metode Latinskih kvadrata [22].

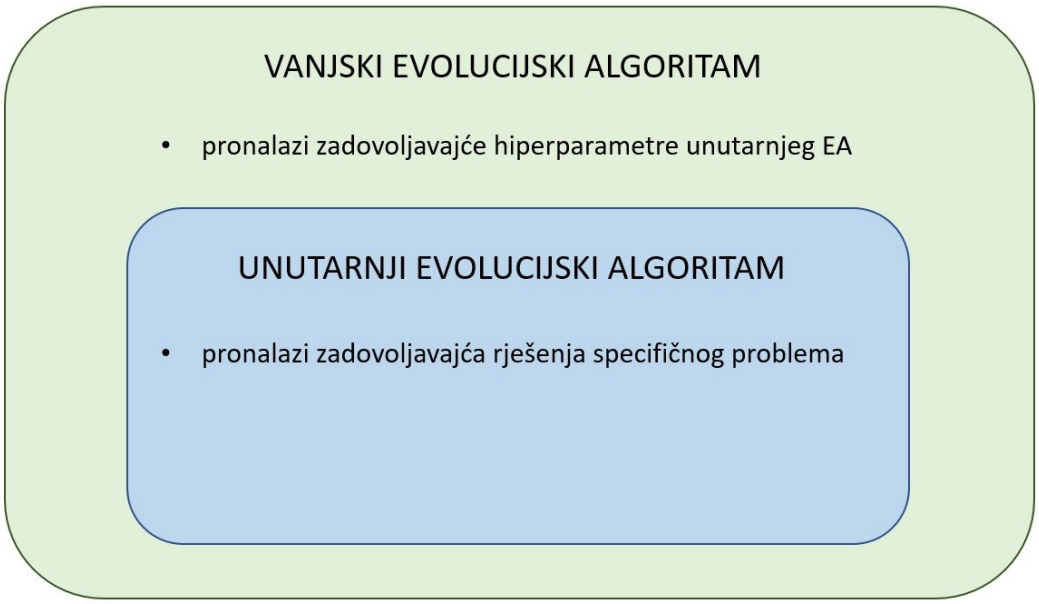
Prilikom ispitivanja velikog broja kombinacija hiperparametara, može se implementirati i tzv. metoda utrkivanja (*eng. racing method*) koja se koristi kako bi se ubrzao cjelokupni proces. Metoda utrkivanja se sastoji od nekoliko koraka:

1. svaka kombinacija hiperparametara se ispituje samo jedanput
2. određuje se kombinacija hiperparametara koja je ostvarila najbolji rezultat unutar jednog pokretanja algoritma
3. odbacuju se sve kombinacije hiperparametara koje su imale znatno lošiji rezultat od najbolje
4. nastavlja se evaluacija samo preostalih kombinacija hiperparametara.

3. DUBOKI EVOLUCIJSKI ALGORITMI

Razmotrimo li još jednom problematiku postavljanja hiperparametara evolucijskog algoritma, može se uočiti da upravo taj tip zadatka spada pod kategoriju teških optimizacijskih problema za koje koristimo evolucijske algoritme. U literaturi [23] se ovakav pristup postavljanju hiperparametara naziva meta pristupom (*eng. Meta evolutionary algorithms*) i može poprimiti razne oblike s obzirom na vrstu evolucijskog algoritma koji se koristi kako bi se postavile vrijednosti hiperparametara drugog evolucijskog algoritma.

U nastavku rada predlaže se pristup koji se temelji na ugnježđivanju dvaju evolucijskih algoritama uz fokus na robusnost metode u smislu pronalaženja hiperparametara različitih tipova evolucijskih algoritama. Kao što je već spomenuto, za implementaciju ove metode koristit će se 2 evolucijska algoritma. Jednom od njih pridružen je naziv unutarnji evolucijski algoritam te on ustvari predstavlja evolucijski algoritam čije hiperparametre postavljamo. Drugi evolucijski algoritam nazvan je vanjskim evolucijskim algoritmom te će njegov zadatak biti pronalaženje optimalnih vrijednosti unutarnjeg evolucijskog algoritma kao što je prikazano na slici 33.

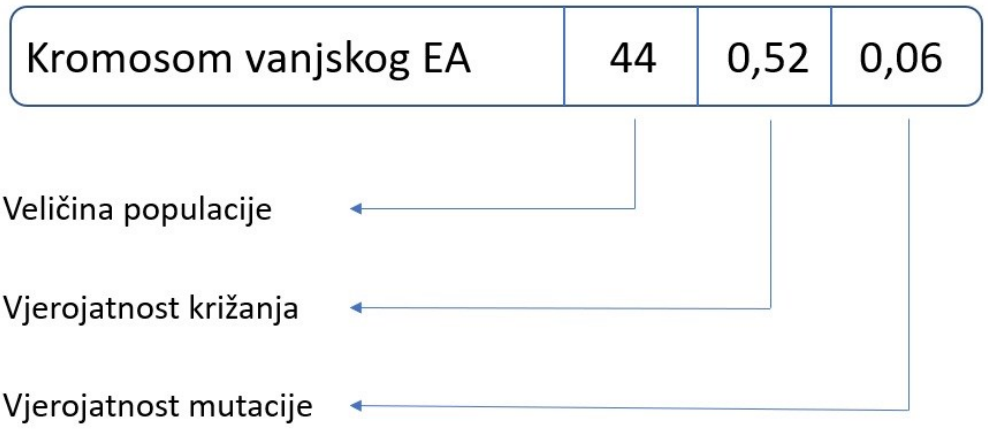


Slika 33. Struktura dubokih evolucijskih algoritama

3.1. Struktura vanjskog evolucijskog algoritma

Struktura vanjskog i unutarnjeg evolucijskog algoritma vrlo je slična uz nekoliko iznimki koje će detaljnije biti obrađene u nastavku rada.

Za početak je potrebno definirati populaciju vanjskog evolucijskog algoritma. Poput unutarnjeg evolucijskog algoritma čija se populacija sastoji od potencijalnih rješenja zadanog problema, populacija vanjskog evolucijskog algoritma sastojat će se od različitih kombinacija hiperparametara unutarnjeg evolucijskog algoritma. Duljina kromosoma zavisit će o broju hiperparametara unutarnjeg evolucijskog algoritma koji se postavljaju. Tako će se na primjer, prilikom postavljanja veličine populacije, vjerojatnosti križanja i vjerojatnosti mutacije unutarnjeg evolucijskog algoritma, kromosom vanjskog evolucijskog algoritma sastojati od 3 vrijednosti koje predstavljaju vrijednosti 3 navedena hiperparametra. Primjer takvog kromosoma prikazan je na slici 34.



Slika 34. Primjer kromosoma vanjskog evolucijskog algoritma

Generiranje inicijalne populacije vanjskog evolucijskog algoritma može se provesti na više načina kao i kod klasičnog evolucijskog algoritma. Ono što treba imati na umu prilikom generiranja inicijalne populacije su granične vrijednosti hiperparametara. Naime, vjerojatnosti križanja i mutacije mogu poprimiti vrijednosti između 0 i 1, dok veličinu populacije često definiramo kao paran broj. Iz tog je razloga potrebno pravilno definirati postupak generiranja inicijalne populacije čak i slučaju kada se provodi inicijalizacija po slučaju. Nakon ovog

postupka, populacija vanjskog evolucijskog algoritma poprimit će matrični oblik kao što je prikazano na slici 35.

```
vanjPopulacija =

  254.0000    0.5307    0.0869
  474.0000    0.5093    0.0518
  360.0000    0.6046    0.0227
  554.0000    0.5282    0.0607
  610.0000    0.5299    0.0207
  376.0000    0.6404    0.0476
  262.0000    0.6093    0.0650
  668.0000    0.6967    0.0183
  602.0000    0.6142    0.0821
  100.0000    0.6684    0.0295
  144.0000    0.5794    0.0653
  494.0000    0.6942    0.0895
  656.0000    0.6763    0.0960
  164.0000    0.5665    0.0784
  206.0000    0.5686    0.0741
  448.0000    0.6818    0.0186
  190.0000    0.5554    0.0829
  474.0000    0.5028    0.0790
  658.0000    0.6545    0.0638
  680.0000    0.5592    0.0485
```

Slika 35. Primjer populacije vanjskog evolucijskog algoritma

Generirane vrijednosti inicijalne populacije također mogu biti ograničene dodatnim uvjetima ako se posjeduje znanje o domeni ili se iz nekog razloga želi ispitati samo određeni dio prostora. Tako se na slici 34 može vidjeti da su vrijednosti veličine populacije ograničene na interval [100, 700], vrijednosti vjerojatnosti križanja na interval [0,5, 0,7] i vrijednosti vjerojatnosti mutacije na interval [0,01, 0,1].

Ovdje se također nameće pitanje veličine populacije vanjskog evolucijskog algoritma. U prikazanom primjeru je veličina populacije vanjskog evolucijskog algoritma postavljena na 20 jedinki kao pokazni primjer. Međutim, kao i kod unutarnjeg evolucijskog algoritma, vrijednosti hiperparametara vanjskog evolucijskog algoritma imat će značajan utjecaj na kvalitetu njihovog izvođenja, odnosno na kvalitetu pronađenih hiperparametara unutarnjeg evolucijskog algoritma. Utjecaj hiperparametara vanjskog evolucijskog algoritma detaljnije će biti razrađena u nastavku rada.

Nakon generiranja inicijalne populacije algoritam ulazi u glavnu iteracijsku petlju. Ovdje se definira uvjet zaustavljanja vanjskog algoritma. Kao što je već prikazano u prethodnom poglavlju, uvjeti zaustavljanja algoritma zavise o željenom ishodu. U slučaju vanjskog evolucijskog algoritma, uvjet zaustavljanja može biti jedno od navedenog:

1. odrađen je prethodno definiran broj iteracija glavne petlje
2. pronađena je kombinacija hiperparametara unutarnjeg evolucijskog algoritma koja zadovoljava prethodno definirane uvjete

Zbog računalne kompleksnosti ove metode koja rezultira dugim vremenskim izvođenjem vanjskog algoritma te uz činjenicu da nikada ne možemo biti sigurni da je pronađen globalni optimum, odnosno da su pronađene optimalne vrijednosti hiperparametara unutarnjeg evolucijskog algoritma, predlaže se korištenje prethodno definiranog broja iteracija glavne petlje kao uvjet zaustavljanja. U slučaju da nakon definiranog broja iteracija još uvijek nije pronađeno zadovoljavajuće rješenje, najbolje rješenje zadnje generacije može se zabilježiti i ubaciti u inicijalnu populaciju prilikom novog pokretanja vanjskog evolucijskog algoritma kako bi se detaljnije ispitalo okolno područje i potencijalno pronašlo još kvalitetnije rješenje.

Unutar glavne iteracijske petlje algoritam ulazi u populacijsku petlju gdje uzima pojedince iz populacije te svakog od njih određen broj puta provlači kroz unutarnji evolucijski algoritam. Naime, s obzirom na stohastičku prirodu evolucijskih algoritama, za svaku kombinaciju hiperparametara je potrebno više puta proći kroz evolucijski postupak kako bi se statistički utvrdila njihova kvaliteta. Naravno, prilikom svakog izvođenja unutarnjeg evolucijskog algoritma, potrebno je popunjavati prethodno alocirani vektor iteracija. Vektor iteracija može poprimiti različite oblike, a on ovisi o tipu problema koji unutarnji evolucijski algoritam rješava. U slučaju rješavanja standardnih ispitnih funkcija kod kojih je optimalno rješenje već poznato, u vektor iteracija se za svako novo izvođenje unutarnjeg algoritma sprema broj iteracija unutar kojeg je optimalno rješenje pronađeno. S druge strane, ako optimalno rješenje problema nije poznato, u vektor iteracija se spremaju vrijednosti najviše iskazane dobrote svakog izvođenja unutarnjeg evolucijskog algoritma.

Uz vektor iteracija, prilikom prolaska kroz populacijsku petlju je korisno spremati u memoriju vrijeme izvođenja algoritma ili optimalno, broj provedenih matematičkih operacija za danu kombinaciju hiperparametara.

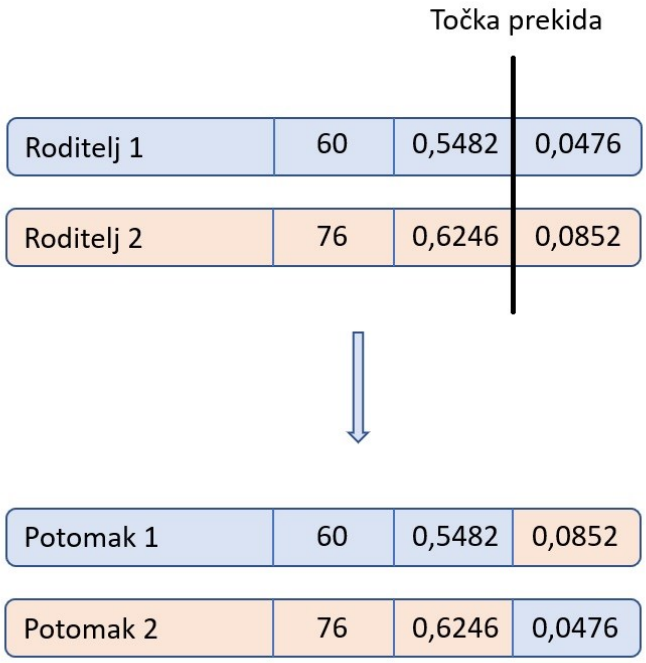
Nakon svih provedenih iteracija za jednu kombinaciju hiperparametara unutar populacijske petlje, vanjski evolucijski algoritam provodi evaluaciju istih hiperparametara korištenjem prethodno spomenutog vektora iteracija i vremena izvođenja ili broja provedenih matematičkih operacija. Ovaj postupak ustvari predstavlja izračun fitnesa jednog člana populacije vanjskog evolucijskog algoritma. Postavljanje ove fitnes funkcije najbitniji je i najteži zadatak prilikom korištenja dubokih evolucijskih algoritama. Naime, prilikom definiranja fitnes funkcije vanjskog evolucijskog algoritma, potrebno je pronaći onaj oblik funkcije koji za dani tip zadatka unutarnjeg evolucijskog algoritma vrednuje kvalitetu pronađenih rješenja u odnosu na vrijeme izvođenja za danu kombinaciju hiperparametara. Primjerice, jedna kombinacija hiperparametara može pronaći najkvalitetnija rješenja tijekom nekih izvođenja unutarnjeg evolucijskog algoritma, ali također imati i najveći broj neuspjelih pokušaja, dok druga kombinacija hiperparametara može konzistentno pružati kvalitetna rješenja. Iz tog razloga fitnes funkcija vanjskog evolucijskog algoritma u najvećoj mjeri ovisi o onim svojstvima unutarnjeg evolucijskog algoritma koje želimo da posjeduje. Ova tema će biti detaljnije obrađena kod primjera implementacije.

Navedeni postupak se provodi sve dok populacijska petlja ne odradi evaluaciju svake kombinacije hiperparametara iz inicijalne populacije. Nakon toga slijedi odabir kombinacije hiperparametara koje su iskazale najvišu dobrotu prilikom evaluacije kako bi se kasnije omogućilo provođenje elitizma. Kao i kod unutarnjeg evolucijskog algoritma, ovaj postupak bi se mogao provoditi i s većim brojem najboljih pojedinaca svake generacije. Prilikom implementacije primjera koji će biti navedeni u nastavku rada pokazalo se da je upravo ovaj oblik elitizma imao najbolji učinak na konačni rezultat. Razlog tome je što bi implementacija elitizma za veći broj pojedinaca rezultirala manjim brojem neispitanih kombinacija hiperparametara unutar svake generacije. To bi znatno usporilo evolucijski postupak jer je veličina populacije vanjskog evolucijskog algoritma već limitirana računalnom kompleksnošću dubokih evolucijskih algoritama.

Sljedeći korak vanjskog evolucijskog algoritma je selekcija roditelja. Selekcija je provedena korištenjem ruletnog pravila. Druge selekcijske metode također ulaze u obzir uz nekoliko iznimaka kao što je selekcija najboljeg postotka u slučaju male populacije vanjskog evolucijskog algoritma. Kao što je već spomenuto, zbog računalne kompleksnosti dubokih evolucijskih algoritama, vanjski evolucijski algoritam najčešće je ograničen na manje populacije, a odabir selekcije najboljeg postotka rezultirao bi danjim smanjenjem raznolikosti pojedinaca unutar već male populacije što bi u konačnici usporilo evolucijski proces.

Do križanja između selektiranih roditelja dolazi pod utjecajem vjerojatnosti križanja vanjskog evolucijskog algoritma. Kao i za veličinu populacije vanjskog algoritma, nameće se pitanje kako izabrati vrijednost ovog hiperparametra. Za implementacijske primjere ovog rada odabrane su tipične vrijednosti unutar intervala [0,5, 0,7]. Međutim, ako bismo htjeli pronaći optimalne vrijednosti morala bi se provesti analiza hiperparametarskog prostora vanjskog evolucijskog algoritma. Takva analiza zahtijevala bi nezamislivu količinu računalne snage, a pronađene vrijednosti hiperparametara vanjskog evolucijskog algoritma vjerojatno bi zavisile o vrsti problema unutarnjeg evolucijskog algoritma za kojeg vanjski algoritam pronalazi hiperparametre. Dakle, upitno je kolika bi bila isplativost istraživanja takvog razmjera samo za jedan specifičan tip zadatka unutarnjeg algoritma.

Za proces križanja je odabrana metoda križanja u jednoj točki. Naravno, moguće je odabrati i neku drugu metodu križanja od onih koje su navedene u prethodnom poglavlju. Tako bi se primjerice u slučaju većeg broja traženih hiperparametara mogla odabrati metoda križanja u više točaka. Kod drugih metoda kao što je intermedijarna rekombinacija, bitno je imati na umu granične vrijednosti svakog zasebnog hiperparametra. Primjer križanja na kombinaciji 3 hiperparametara prikazan je na slici 36.

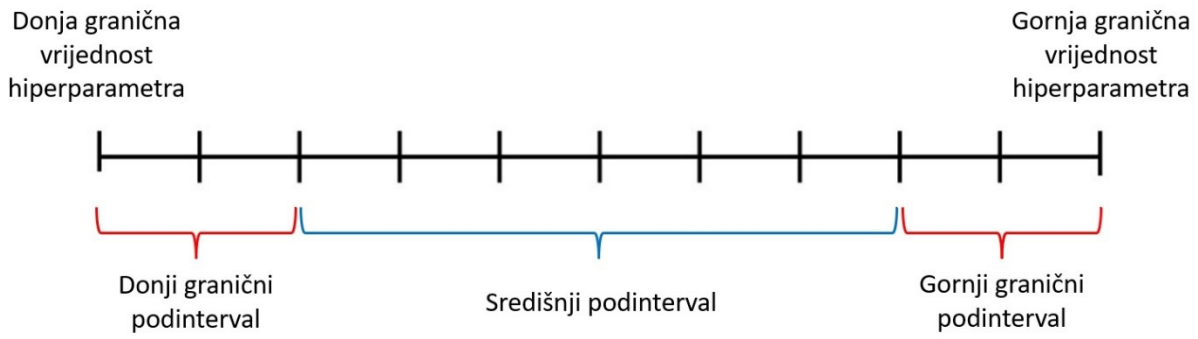


Slika 36. Primjer križanja u jednoj točki s kromosomima koji se sastoje od 3 hiperparametra

Nakon križanja slijedi postupak mutacije do koje dolazi pod utjecajem vjerojatnosti mutacije vanjskog evolucijskog algoritma. Za odabir vrijednosti vjerojatnosti mutacije vrijedi sve što je prethodno napisano o odabiru vrijednosti vjerojatnosti križanja. Međutim, za razliku od vjerojatnosti križanja koja je odabrana unutar tipičnog intervala, za vjerojatnost mutacije je odabrana vrijednost 0,2 koja premašuje tipični interval [0, 0,1]. Razlog tome je potreba za jednostavnijim i bržim ispitivanjem okolnog prostora trenutnog člana koji iskazuje najvišu vrijednost dobrote. Naime, u slučaju kromosoma koji se sastoje od malog broja hiperparametara, šansa da nastupi mutacija jednog od njih vrlo je mala ako se provjera nastupanja mutacije odvija za svaki hiperparametar zasebno. U slučaju provjere nastupanja mutacije na razini svakog kromosoma, duljina kromosoma ne bi imala nikakvu ulogu.

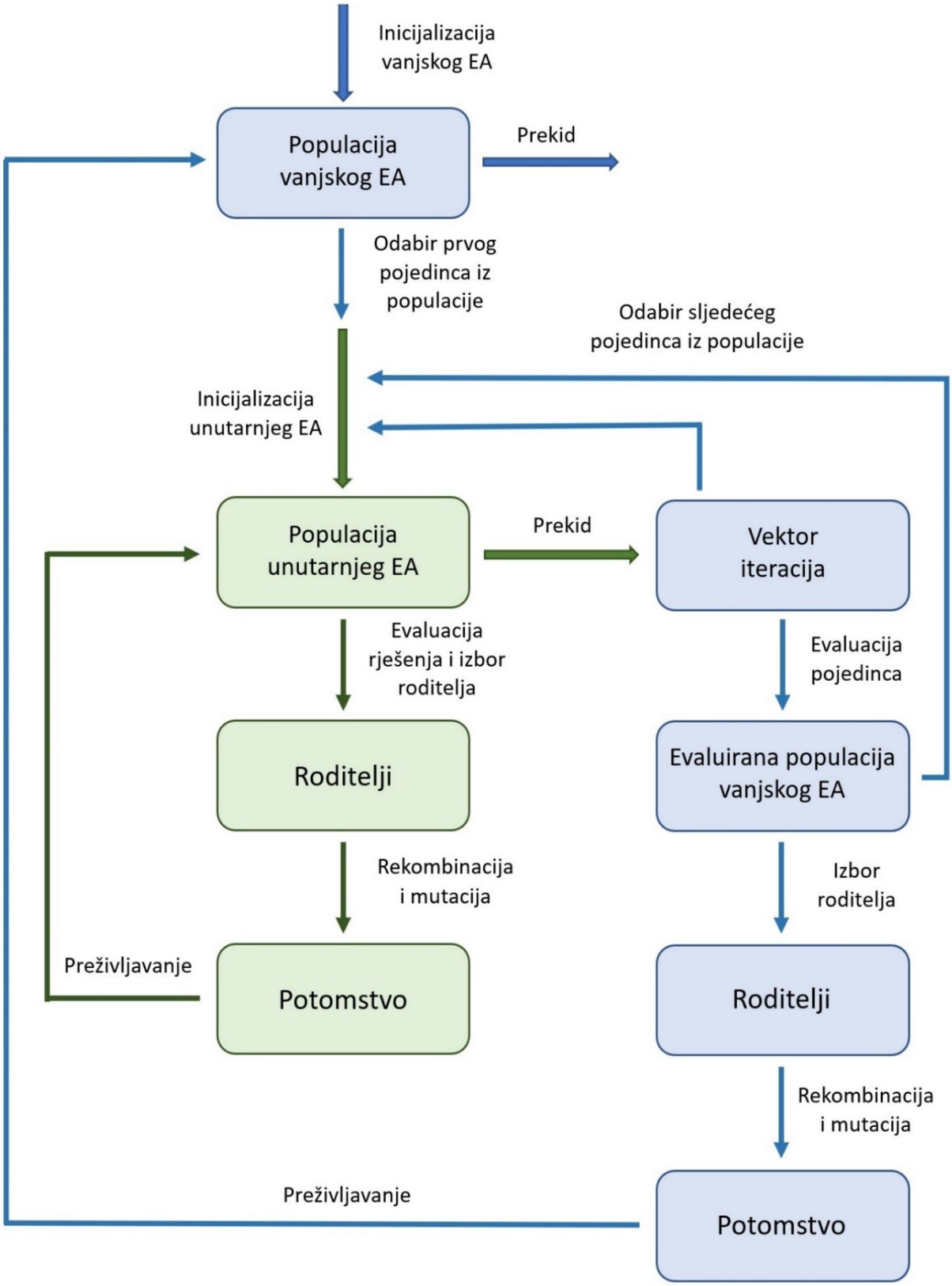
S obzirom na već spomenute granične vrijednosti traženih hiperparametara i zasebnih uvjeta kao što je parnost veličine populacije, postupak mutacije vanjskog evolucijskog algoritma mora se definirati zasebno za svaki tip hiperparametra koji se nalazi unutar kromosoma. Kako bi se dobio željeni efekt, prije provođenja mutacije vrijednosti nekog hiperparametra, prvo se provjerava unutar kojeg podintervala se njegova trenutna vrijednost nalazi. Interval unutar kojeg tražimo vrijednost nekog parametra podijeljen je na 3 podintervala. Središnji

podinterval predstavlja vrijednosti hiperparametra koje se ne nalaze u blizini graničnih vrijednosti te kao takav omogućava mutaciju koja može uvećati ili umanjiti trenutnu vrijednost hiperparametra. Bitno je napomenuti da mutacija unutar ovog podintervala mora biti tako definirana da u svakom mogućem slučaju konačna vrijednost hiperparametra ne prekorači zadane granične vrijednosti. Donji i gornji podintervali predstavljaju vrijednosti koje se nalaze u blizini graničnih vrijednosti promatranog hiperparametra te se unutar njih provodi mutacija koja uvijek uvećava ili uvijek umanjuje trenutnu vrijednost kako bi se osiguralo da će se konačna vrijednost hiperparametra nalaziti unutar prethodno definiranog intervala. Prikaz podjele na podintervale prikazan je na slici 37.



Slika 37. Podjela intervala dopuštenih vrijednosti hiperparametra na mutacijske podintervale

Nakon mutacije slijedi provođenje elitizma, odnosno ubacivanje prethodno evaluiranog pojedinca s najvišom iskazanom dobrotom čime konačno završava prvi korak glavne petlje vanjskog evolucijskog algoritma. Novonastala populacija ulazi u novi korak glavne petlje čime započinje novi ciklus svih prethodno opisanih postupaka. Kao što je prethodno spomenuto, vanjski algoritam se zaustavlja u trenutku kada je ostvaren uvjet zaustavljanja nakon čega slijedi prikaz dobivenih rezultata. Detaljni strukturni prikaz dubokog evolucijskog algoritma može se vidjeti na slici 38.



Slika 38. Detaljni strukturni prikaz dubokog evolucijskog algoritma

4. IMPLEMENTACIJA

U ovom poglavlju će biti prikazana implementacija dubokog evolucijskog algoritma za samopodešavanje hiperparametara na nekoliko različitih standardnih ispitnih funkcija. Implementacija je napravljena unutar programskog okruženja MATLAB. Također će biti prikazana usporedba kvalitete dobivenih hiperparametara s onima koje dobivamo korištenjem standardnih metoda podešavanja.

4.1. OneMin problem

OneMax i OneMin problemi predstavljaju jedne od najčešće korištenih standardnih optimizacijskih problema za evaluaciju različitih struktura evolucijskih algoritama ili postavljenih hiperparametara jedne specifične strukture. Razlog tome leži u jednostavnoj strukturi problema koja daje dobar uvid u samu kvalitetu korištenog evolucijskog algoritma.

Problem je definiran kao maksimizacijski ili minimizacijski problem (obzirom na to radi li se o OneMax ili OneMin problemu), a sama funkcija koju je potrebno maksimizirati ili minimizirati prikazuje se kao:

$$F(\mathbf{X}) = \sum_{i=1}^{i=u} x_i. \tag{7}$$

Gdje je:

$$\mathbf{X} = \{x_1, x_2, \dots, x_{u-1}, x_u\}, \text{ uz } x_i \in \{0,1\}. \tag{8}$$

Maksimalna teorijska vrijednost navedene funkcije iznosi *u* u slučaju kada svaki član poprima vrijednost 1 te kao takva predstavlja željeni izlaz evolucijskog algoritma koji rješava OneMax problem. S druge strane, minimalna teorijska vrijednost funkcije iznosi 0 kada svaki član poprima vrijednost 0 i kao takva predstavlja željeni izlaz evolucijskog algoritma koji rješava OneMin problem. U praksi su općenito česti optimizacijski problemi poput ovoga kod kojih je globalni optimum već poznat. Međutim, također je često kod takvih problema nepoznato gdje

se globalni optimum nalazi pa se stoga koriste evolucijski algoritma kako bi se on pronašao. U nastavku će se isključivo promatrati OneMin problem za koji je napravljena implementacija.

4.1.1. Implementacija dubokog evolucijskog algoritma za OneMin problem

Implementacijom dubokog evolucijskog algoritma pokušat će se pronaći kvalitetne vrijednosti hiperparametara unutarnjeg evolucijskog algoritma koji rješava OneMin problem. Specifično će se tražiti:

- Veličina populacije.
- Vrijednost vjerojatnosti križanja.
- Vrijednost vjerojatnosti mutacije.

Za duljinu niza je odabrano A=20. Ova vrijednost dovoljno je velika da smanji mogućnost slučajnog generiranja konačnog rješenja prilikom inicijalizacije što bi imalo negativan utjecaj na ispitivanje hiperparametara s obzirom na to da nam ne daje nikakav uvid u kvalitetu algoritma. Razlog zašto nije odabrana još veća duljina niza je računalna složenost cjelokupnog procesa. Naime, već s ovom duljinom niza je potrebno puno računalne snage kako bi se vanjski evolucijski algoritam proveo do uvjeta zaustavljanja što znači da samo izvođenje može trajati iznimno dugo uz ograničene resurse.

Za početak se generira inicijalna populacija vanjskog evolucijskog algoritma. Za veličinu populacije vanjskog evolucijskog algoritma je odabrano B_{vanj}=20. Kao što je već spomenuto u prethodnom poglavlju, optimalnu vrijednost veličine populacije vanjskog evolucijskog algoritma zasad ne možemo odrediti, a s obzirom na to da smo ograničeni zbog računalne kompleksnosti algoritma uzeta je dovoljno mala populacija koja neće dodatno usporiti postupak, a koja još uvijek nije premala da bi narušila sam evolucijski proces.

Veličina populacije unutarnjeg evolucijskog algoritma B_{unut} tražit će se unutar intervala [20, 200] i bit će tako definirana da se uvijek zaokruži na najbliži parni broj kako bi se kasnije pojednostavio postupak križanja unutar unutarnjeg algoritma. Vrijednost vjerojatnosti križanja unutarnjeg evolucijskog algoritma tražit će se unutar intervala [0,4, 0,8], a vrijednost

vjerojatnosti mutacije unutarnjeg evolucijskog algoritma unutar intervala [0,01, 0,2]. Ovime je namjerno proširen prostor pretrage unutar kojeg tipično možemo očekivati kvalitetne rezultate kako bi utvrdili hoće li duboki evolucijski algoritam samostalno suziti prostor pretrage na onaj unutar kojeg se uistinu nalaze kvalitetne kombinacije hiperparametara. Inicijalizacija populacije vanjskog evolucijskog algoritma provedena je metodom generiranja po slučaju. Na kraju ovog rada osvrnut ćemo se na izbor metode inicijalizacije vanjskog evolucijskog algoritma uz prijedlog korištenja i nekih drugih metoda.

Nakon inicijalizacije, populacija vanjskog evolucijskog algoritma poprimit će oblik sličan onom koji je prikazan na slici 39.

```
vanjPopulacija =

   48.0000    0.6601    0.1635
  176.0000    0.5976    0.1233
   96.0000    0.6342    0.1471
   92.0000    0.7962    0.0682
  188.0000    0.7522    0.0794
   34.0000    0.4886    0.0490
  170.0000    0.7371    0.0140
  114.0000    0.4257    0.1834
  194.0000    0.4165    0.0545
  186.0000    0.5658    0.1592
  196.0000    0.6216    0.1429
  148.0000    0.5874    0.1772
  134.0000    0.5406    0.0846
  156.0000    0.4696    0.0582
  176.0000    0.6348    0.0338
  144.0000    0.4102    0.1306
   70.0000    0.6883    0.0532
  192.0000    0.4195    0.1254
   92.0000    0.5141    0.1643
  162.0000    0.7298    0.0294
```

Slika 39. Primjer inicijalne populacije vanjskog evolucijskog algoritma (OneMin)

Sljedeći korak je ulazak u populacijsku petlju unutar koje se uzima prva kombinacija hiperparametara iz populacije vanjskog evolucijskog algoritma kao što je prikazano na slici 40.

Slika 40. Uzimanje prve kombinacije hiperparametara za provođenje unutarnjeg evolucijskog algoritma (OneMin)

Nakon toga se 1000 puta ponavlja izvršavanje unutarnjeg evolucijskog algoritma čiji se hiperparametri za svako od tih ponavljanja prepisuju iz uzete kombinacije hiperparametara. Definirana su dva uvjeta zaustavljanja unutarnjeg evolucijskog algoritma. Jedan od njih je pronalazak minimalne teorijske vrijednosti funkcije (8), odnosno kada jedno od rješenja unutar populacije unutarnjeg evolucijskog algoritma poprimi oblik niza koji se sastoji smo od nula. Drugi uvjet zaustavljanja je prolazak kroz 200 iteracija unutarnjeg algoritma bez pronalaska optimalnog rješenja. Ovime je ograničeno da u najgorem mogućem scenariju evaluacija neke kombinacije hiperparametara neće trajati dulje od vremena koje je potrebno da se izvede 200 000 iteracija unutarnjeg algoritma.

Nakon svakog zaustavljanja unutarnjeg algoritma, broj iteracija unutar kojih je pronađeno optimalno rješenje zapisuje se u vektor iteracija. U slučaju da se tijekom izvođenja unutarnjeg algoritma ostvario drugi uvjet zaustavljanja, odnosno da nije pronađeno rješenje unutar 200 iteracija, u vektor iteracija se upisuje 200 kao što bi bio slučaj da se unutar točno 200 iteracija rješenje pronašlo. Ovo neće stvarati problem s obzirom na to da nas zanimaju isključivo

kombinacije hiperparametara kod kojih se to gotovo nikada ne događa. Ovime smo onemogućili vanjskom evolucijskog algoritmu da pravilno evaluira nekvalitetne kombinacije hiperparametara već ih sve smatra jednako lošima, ali smo uvelike uštedjeli na cjelokupnom vremenu izvođenja. Primjer vektora iteracija prikazan je na slici 41.

```
vektoriteracija =

  Columns 1 through 10

    38    22    10    26    13    23    41    43    44    15

  Columns 10 through 20

    26    82    29    54    29    22    48    19    24    11

  . . .
```

Slika 41. Primjer vektora iteracija (OneMin)

Osim broja iteracija unutar kojih je pronađeno rješenje, tijekom ovog postupka se također sprema broj odrađenih evaluacija fitnesa koje se odvijaju u unutarnjem evolucijskom algoritmu.

Kada se izvrše sva izvođenja unutarnjeg evolucijskog algoritma s istom kombinacijom hiperparametara slijedi njihova evaluacija. Način evaluacije pojedine kombinacije hiperparametara vrlo je bitan jer određuje smjer konvergencije evolucijskog procesa. Naime, prilikom definiranja fitnes funkcije vanjskog evolucijskog algoritma je potrebno dobro poznavati problem koji rješava unutarnji algoritam kako bi se prepoznali i iskoristili oni parametri koji uistinu iskazuju njegovu kvalitetu. U nastavku će biti prikazan postupak pronalaženja fitnes funkcije za ovaj problem.

Za fitnes funkciju vanjskog evolucijskog algoritma isprva je korišten oblik:

$$\Phi_{1,1} = \frac{100}{\bar{\mathbf{v}}}. \qquad \textbf{(9)}$$

Gdje je:

$\bar{\mathbf{v}}$ aritmetička sredina svih članova vektora iteracija

Pokazalo se da je negativna strana korištenja ovakve fitnes funkcije to što se prevelik značaj daje onim izvođenjima unutarnjeg algoritma kod kojih se unutar 200 iteracija ne bi pronašlo rješenje. Naime, takav slučaj trebao bi imati negativan utjecaj na njihovu dobrotu, međutim taj utjecaj ne bi trebao biti od tolikog značaja.

Kako bi se riješio taj problem, umjesto korištenja aritmetičke sredine svih članova vektora iteracija, iskorišten je njihov medijan, a kako bi se onim izvođenjima unutarnjeg algoritma kod kojih se unutar 200 iteracija ne bi pronašlo rješenje ipak pridao neki značaj, u funkciju se ubacuje *N* koji predstavlja broj puta koliko se unutar 200 iteracija nije pronašlo rješenje. Fitnes funkcija sada poprima sljedeći oblik:

$$\Phi_{1,2} = \frac{10000}{\sqrt{1 + \frac{N}{15}\tilde{\mathbf{v}}^2}}. \qquad \textbf{(10)}$$

Gdje su:

$\tilde{\mathbf{v}}$ medijan svih članova vektora iteracija

N broj puta koliko se unutar 200 iteracija nije pronašlo rješenje

Novo dodani član definiran je tako da niži broj ponavljanja kod kojih se unutar 200 iteracija nije pronašlo rješenje neće imati velik utjecaj na dobrotu. Međutim, ako se to prilikom izvođenja ponovi veći broj puta, znatno će smanjiti evaluiranu dobrotu pojedinca. Budući da je medijan ipak znatno važniji prilikom evaluacije od dodanog člana, njegova vrijednost je kvadrirana, a da bi vrijednost dobrote održali unutar smislenih vrijednosti, povećana je konstanta u brojniku.

Korištenje takve fitnes funkcije svejedno je rezultiralo sljedećim problemima:

1. vanjski evolucijski algoritam favorizirao je one kombinacije hiperparametara koje su imale velike populacije bez obzira na njihovo vrijeme izvođenja
2. pokazalo se da medijan nema dobro svojstvo ponovljivosti što bi rezultiralo time da se ista kombinacija hiperparametara ne bi niti približno jednako vrednovala unutar različitih iteracija vanjskog evolucijskog algoritma.

Iz navedenih razloga je formiran sljedeći oblik fitnes funkcije:

$$\Phi_{1,3} = \frac{2500 \cdot L}{t \cdot \bar{\mathbf{v}}^3}. \quad \textbf{(11)}$$

Gdje su:

L broj ponavljanja unutarnjeg evolucijskog algoritma s istim hiperparametrima

t vrijeme potrebno za sva izvođenja unutarnjeg algoritma s istim hiperparametrima

Kako bismo riješili problem ponovljivosti, umjesto medijana vraća se izračun aritmetičke sredine svih članova vektora iteracija. Umjesto broja puta koliko se unutar 200 iteracija nije pronašlo rješenje, sada se koristi vrijeme potrebno za sva izvođenja unutarnjeg algoritma s istim hiperparametrima čime se penalizira korištenje velikih populacija koje usporavaju proces.

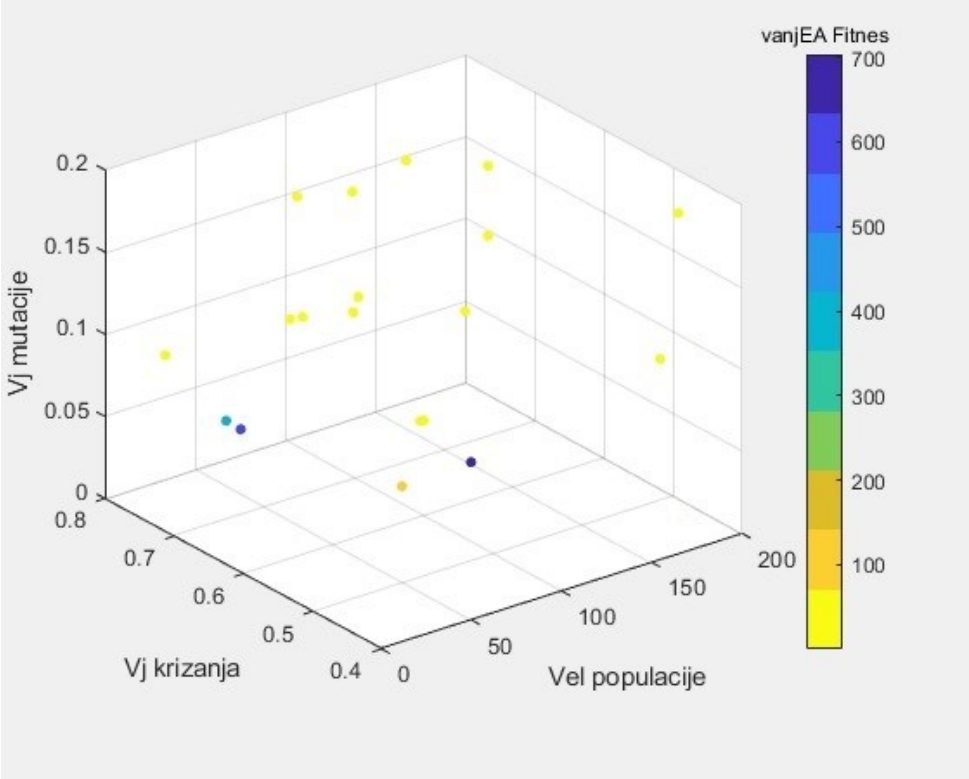
Ova fitnes funkcija pokazala se dobrom u smislu pronalaženja rješenja s višom iskazanom dobrotom u svakoj sljedećoj iteraciji glavne iteracijske petlje vanjskog evolucijskog algoritma. Jedina potrebna izmjena bila je korištenje broja izvođenja fitnes funkcije unutarnjeg algoritma umjesto vremena izvođenja cjelokupnog procesa s obzirom na to da vrijeme izvođenja zavisi o specifikacijama računala na kojemu se postupak provodi. Konačna fitnes funkcija poprima sljedeći oblik:

$$\Phi_{1,4} = \frac{1{,}25 \cdot 10^{10} \cdot L}{g \cdot \bar{\mathbf{v}}^3}. \quad \textbf{(12)}$$

Gdje je:

g broj izvođenja fitnes funkcije unutarnjeg algoritma

Nakon evaluacije prve kombinacije hiperparametara, uzima se sljedeća kombinacija hiperprametara iz populacije vanjskog algoritma te se provodi isti postupak sve dok svaka kombinacija hiperparametara nije evaluirana. Prije selekcije roditelja, na ekranu se prikazuje prostorni razmještaj svih evaluiranih kombinacija hiperparametara uz vrijednost njihove iskazane dobrote. Tako će nakon evaluacije inicijalne populacije vanjskog algoritma grafički prikaz izgled sličan onome koji je prikazan na slici 42.



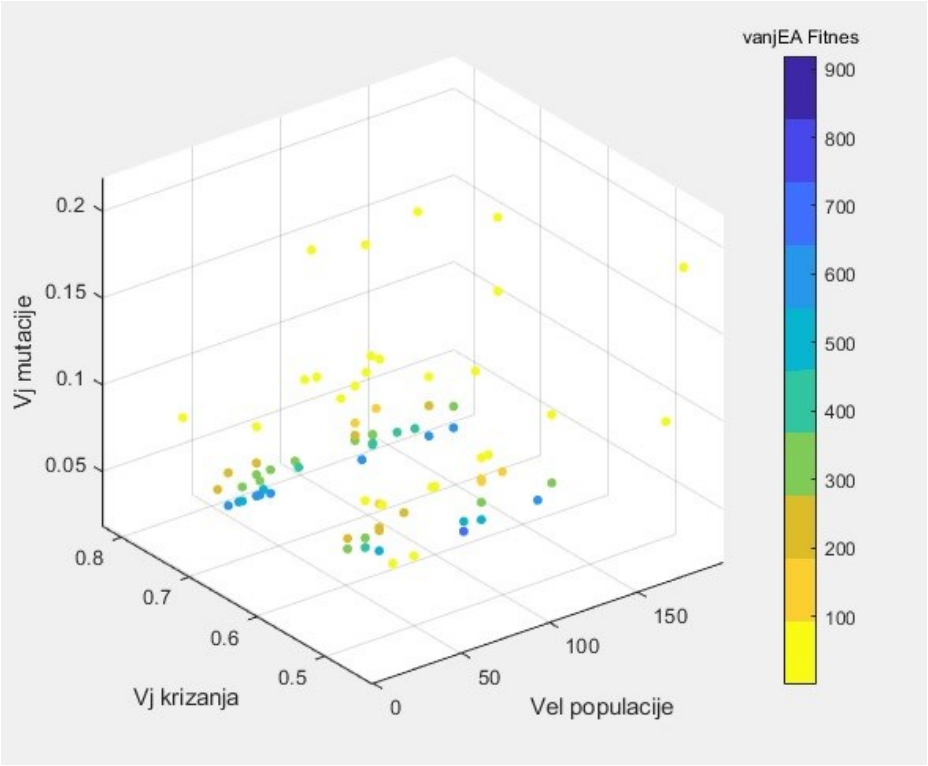
Slika 42. Primjer prostornog prikaza evaluiranih kombinacija hiperparametara inicijalne populacije vanjskog evolucijskog algoritma (OneMin)

Selekcija se provodi korištenjem ruletnog pravila nakon čega slijedi križanje. Križanje se odvija pod utjecajem hiperparametra vjerojatnosti križanja vanjskog evolucijskog algoritma koji je postavljen na vrijednost 0,7, a provodi se metodom križanja u jednoj točki.

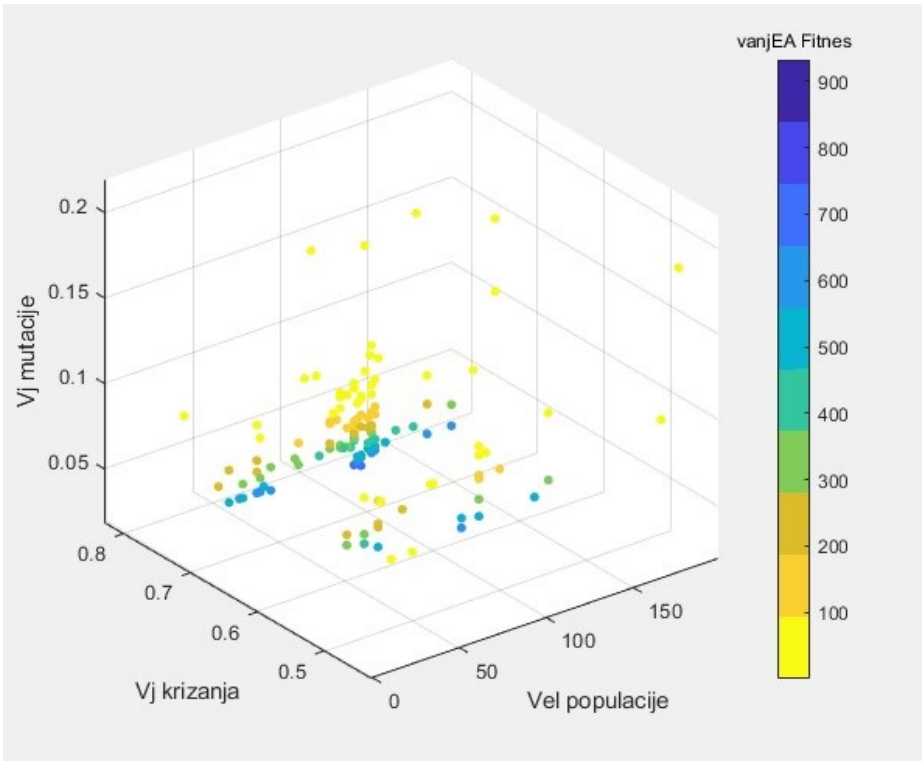
Nakon križanja slijedi postupak mutacije koji je detaljnije već razrađen u prethodnom poglavlju. S obzirom na to da se veličina populacije traži unutar intervala [20, 200], ukoliko se vrijednost veličine populacije na kojoj se odvija mutacija nalazi unutar donjeg podintervala koji je definiran kao [20, 40], odvija se mutacija koja može samo povećati veličinu populacije. Ako se veličina populacije na kojoj se odvija mutacija nalazi unutar srednjeg podintervala koji je definiran kao (40, 180), odvija se oblik mutacije koji može povećati ili umanjiti veličinu populacije uz ograničenje da ne može prekoračiti granične vrijednosti. Konačno, ako se vrijednost veličine populacije na kojoj se odvija mutacija nalazi unutar gornjeg podintervala koji je definiran kao [180, 200], odvija se mutacija koja može samo umanjiti veličinu populacije. Postupak analogan ovome odvija se i za preostala dva hiperparametra. Za vjerojatnost križanja su zadani podintervali: donji [0,4, 0,45], srednji (0,45, 0,75) i gornji [0,75, 0,8], dok su za vjerojatnost mutacije zadani podintervali: donji [0,01, 0,05], srednji (0,05, 0,15) i gornji [0,15, 0,2].

Konačno, nakon mutacije se provodi elitizam pojedinca s najvećom vrijednošću iskazane dobrote čime se završava prva iteracija glavne iteracijske petlje vanjskog evolucijskog algoritma. Ovaj postupak se ponavlja sve dok se ne zadovolji uvjet zaustavljanja vanjskog evolucijskog algoritma koji u ovom slučaju predstavlja 40 odrađenih iteracija glavne iteracijske petlje.

Nakon svake odrađene populacijske petlje, na prostorni prikaz se ucrtavaju nove evaluirane kombinacije hiperparametara kako bi se mogao pratiti tijek evolucijskog postupka vanjskog algoritma. Tako je nakon nekoliko iteracija glavne iteracijske petlje dobiven prikaz koji se može vidjeti na slici 43, a nakon svih odrađenih iteracija prikaz koji se može vidjeti na slici 44.



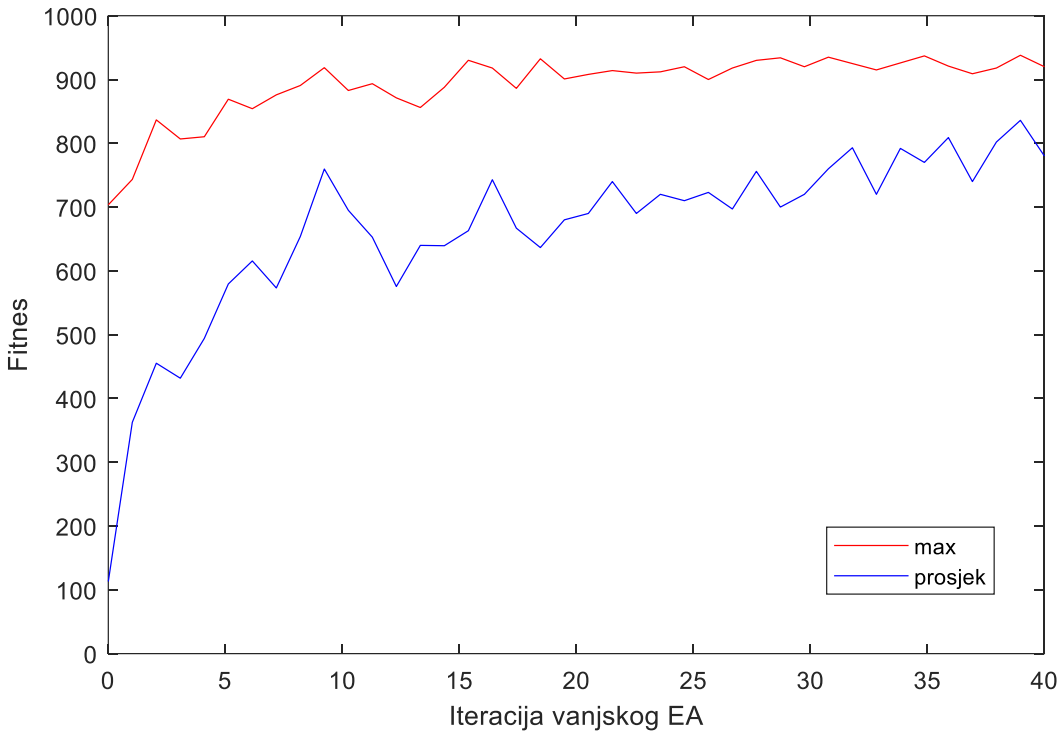
Slika 43. Primjer prostornog prikaza evaluiranih kombinacija hiperparametara nakon nekoliko iteracija glavne iteracijske petlje (OneMin)



Slika 44. Primjer prostornog prikaza evaluiranih kombinacija hiperparametara nakon svih odrađenih iteracija glavne iteracijske petlje (OneMin)

Na slikama 43 i 44 može se vidjeti da je najveći utjecaj na vrijednost iskazane dobrote za ovaj tip problema i korišteni oblik fitnes funkcije imao iznos vjerojatnosti mutacije. Također se može vidjeti da su već nakon prvih nekoliko obavljenih iteracija potpuno izbačene one kombinacije hiperparametara kod kojih je vrijednost vjerojatnosti križanja bila veća od 0,05 što znači da je vanjski evolucijski algoritam uistinu samostalno došao do istog zaključka.

Promatrajući grafički prikaz maksimalnih i prosječnih vrijednosti ostvarene dobrote za svaku iteraciju glavne iteracijske petlje, može se uočiti da prosječna vrijednost ostvarene dobrote populacije ima nagli porast prilikom prvih nekoliko iteracija nakon čega se polako približava maksimalnoj ostvarenoj vrijednosti unutar iste populacije kao što je i očekivano. Maksimalna ostvarena vrijednost dobrote također raste prilikom prvih nekoliko iteracija nakon čega stagnira. Razlog tome je pronalazak lokalnog optimuma nakon čega se sa svakom sljedećom iteracijom sve manje ispituju vrijednosti izvan okolnog područja. Grafički prikaz maksimalnih i prosječnih vrijednosti ostvarene dobrote za svaku iteraciju glavne iteracijske petlje može se vidjeti na slici 45.



Slika 45. Grafički prikaz maksimalnih i prosječnih vrijednosti ostvarene dobrote za svaku iteraciju glavne iteracijske petlje (OneMin)

Na slici 45 također se može uočiti da provođenje elitizma svejedno ne garantira monotoni rast maksimalno ostvarene dobrote. Razlog tome je stohastička priroda unutarnjeg evolucijskog algoritma koji prilikom više izvođenja s jednako postavljenim hiperparametrima neće uvijek ostvariti jednaki rezultat. U ovom slučaju se pokazalo da će taj rezultat varirati čak i ako uspoređujemo statističke podatke dobivene na temelju 1000 izvođenja iste kombinacije hiperparametara.

Prosječno ostvarena dobrota prilikom inicijalizacije iznosila je oko 100 što bi odgovaralo prosječno iskazanoj dobroti prilikom korištenja nasumične metode postavljanja hiperparametara. S obzirom na stohastičku prirodu dubokih evolucijskih algoritama, bez statističke analize rezultata velikog broja izvođenja vanjskog evolucijskog algoritma, teško je napraviti usporedbu s drugim metodama. Naime, prilikom pokretanja vanjskog evolucijskog algoritma, maksimalna vrijednost ostvarene dobrote unutar inicijalne populacije može varirati, a njezin iznos znatno će utjecati na brzinu pronalaska kvalitetne kombinacije hiperparametara.

4.2. Problem trgovačkog putnika

Problem trgovačkog putnika (*eng. Travelling salesman problem, TSP*) poznat je optimizacijski problem kod kojeg je za konačan broj gradova potrebno pronaći najkraću rutu posjetivši pri tome svaki grad jednom prije povratka u polazišnu točku. Ovaj problem prvi puta je formuliran 1930. godine i od tada se koristi kao standardni test prilikom ispitivanja raznih optimizacijskih metoda. Ovaj tip zadatka bitan je jer se često pojavljuje u praksi: određivanje rute vozila, određivanje redoslijeda bušenja rupa kod tiskanih pločica, alokacija opreme u skladištu, sekvenciranje DNK itd.

Problem je definiran kao minimizacijski problem kod kojeg funkcija koja se minimizira poprima sljedeći oblik:

$$F(\mathbf{X}_g) = \frac{1}{\sum_{i=1}^{i=N_g} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}}. \quad \textbf{(13)}$$

Gdje su:

$\mathbf{X}_{\boldsymbol{g}}$ matrica koja sadrži Kartezijeve koordinate posjećenih gradova (x_i, y_i) u redoslijedu njihovog posjećivanja

N_g ukupni broj gradova

4.2.1. Implementacija dubokog evolucijskog algoritma za TSP problem

Implementacijom dubokog evolucijskog algoritma pokušat će se pronaći kvalitetne vrijednosti hiperparametara unutarnjeg evolucijskog algoritma koji rješava problem trgovačkog putnika. Specifično će se tražiti:

- Vrijednost vjerojatnosti križanja.
- Vrijednost vjerojatnosti mutacije.

S obzirom na to da je ovaj problem računalno zahtjevniji od prethodno obrađenog OneMin problema (zbog čega zahtjeva i dulje vrijeme izvođenja) te kako bi se prikazao učinak dubokog evolucijskog algoritma prilikom postavljanja dvaju hiperparametara, veličina populacije je strogo definirana na 50 jedinki unutar svake populacije unutarnjeg evolucijskog algoritma. Promotrimo li prostorni prikaz kombinacije traženih hiperparametara koje je za implementaciju dubokog evolucijskog algoritma za OneMin problem prikazan na slici 41, uočit ćemo da prilikom postavljanja vrijednosti vjerojatnosti križanja i mutacije ustvari pretražujemo samo jednu ravninu. Ovaj oblik postavljanja hiperparametara vrlo je čest u praksi, a razlog tome je dodatan uvjet vremenskog ograničenja. Naime, od svih hiperparametara, veličina populacije najčešće ima najveći utjecaj na vrijeme izvođenja algoritma. Tako se postavljanjem strogo definirane veličine populacije u određenim granicama može pretpostaviti vrijeme izvođenja prije samog pokretanja algoritma.

Kao i kod prethodnog primjera implementacije, vjerojatnost križanja vanjskog evolucijskog algoritma postavljena je na 0,7, a vjerojatnost mutacije na 0,2. Veličina populacije vanjskog algoritma također iznosi 20 jedinki. Međutim, za razliku od OneMin problema, uvjet zaustavljanja definiran je kao 10 odrađenih iteracija vanjskog evolucijskog algoritma.

Vrijednost vjerojatnosti križanja unutarnjeg evolucijskog algoritma tražit će se unutar intervala [0,2, 0,9], a vrijednost vjerojatnosti mutacije unutarnjeg evolucijskog algoritma unutar intervala [0,01, 0,4].

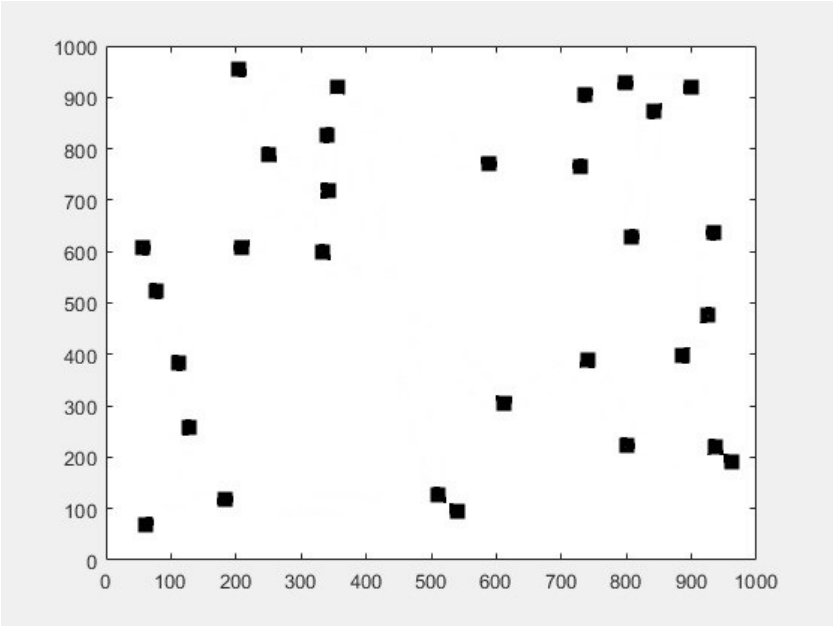
Primjer inicijalne populacije vanjskog evolucijskog algoritma prikazan je na slici 46.

```
vanjPopulacija =

   50.0000    0.7259    0.1471
   50.0000    0.3786    0.3340
   50.0000    0.5542    0.2383
   50.0000    0.6894    0.2244
   50.0000    0.8236    0.3677
   50.0000    0.8715    0.1215
   50.0000    0.5831    0.3053
   50.0000    0.2970    0.3040
   50.0000    0.3045    0.1584
   50.0000    0.3803    0.2315
   50.0000    0.7885    0.0396
   50.0000    0.3780    0.0310
   50.0000    0.7700    0.2170
   50.0000    0.3705    0.3139
   50.0000    0.8505    0.3743
   50.0000    0.4450    0.0607
   50.0000    0.3376    0.2318
   50.0000    0.3758    0.1931
   50.0000    0.6312    0.0146
   50.0000    0.5313    0.1415
```

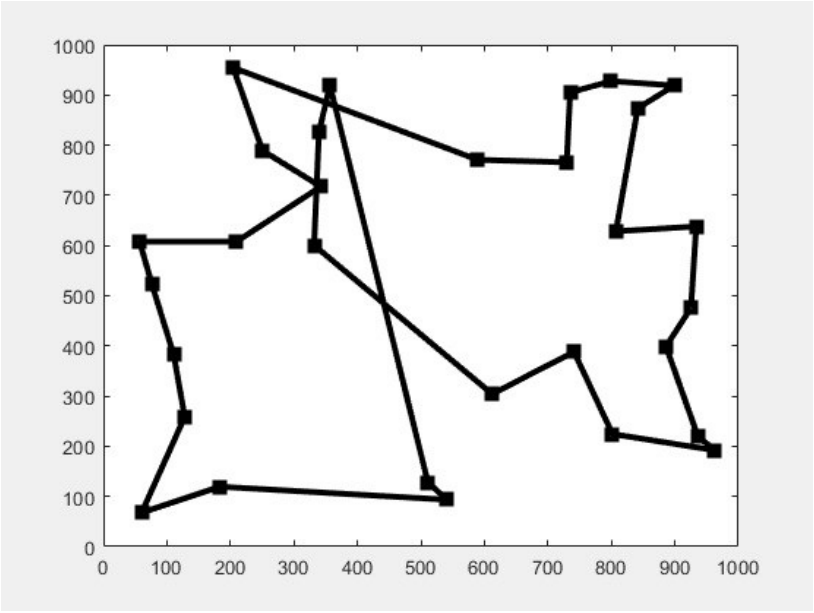
Slika 46. Primjer inicijalne populacije vanjskog evolucijskog algoritma (problem trgovačkog putnika)

Prije ulaska u populacijsku petlju, potrebno je definirati poziciju svakog grada koji se mora posjetiti. Za ovu implementaciju odabrano je 30 po slučaju generiranih točaka koji reprezentiraju navedene gradove. Raspored po slučaju generiranih točaka koji je korišten prilikom implementacije dubokog evolucijskog algoritma prikazan je na slici 47.



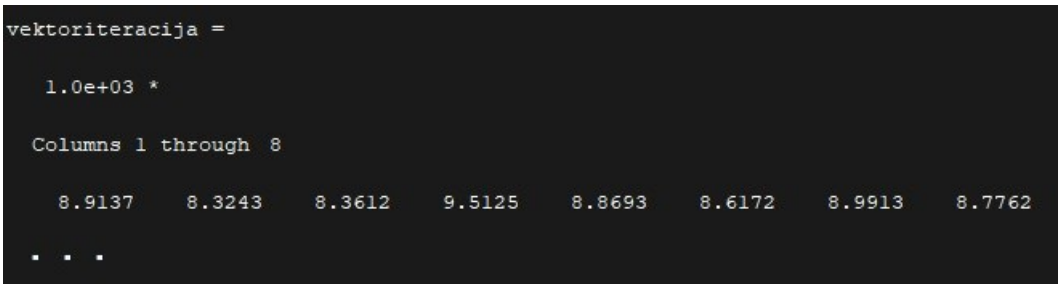
Slika 47. Raspored po slučaju generiranih točaka koji je korišten prilikom implementacije (problem trgovačkog putnika)

Unutar populacijske petlje svaka kombinacija hiperparametara provodi se 100 puta. Uvjet zaustavljanja unutarnjeg evolucijskog algoritma definiran je kao 3000 provedenih iteracija. Nakon svakog prekida unutarnjeg evolucijskog algoritma, grafički se prikazuje najkraća pronađena ruta kao što se može vidjeti na slici 48, te se u vektor iteracija sprema duljina pronađene rute.



Slika 48. Primjer najkraće pronađene rute (problem trgovačkog putnika)

Primjer vektora iteracija može se vidjeti na slici 49.



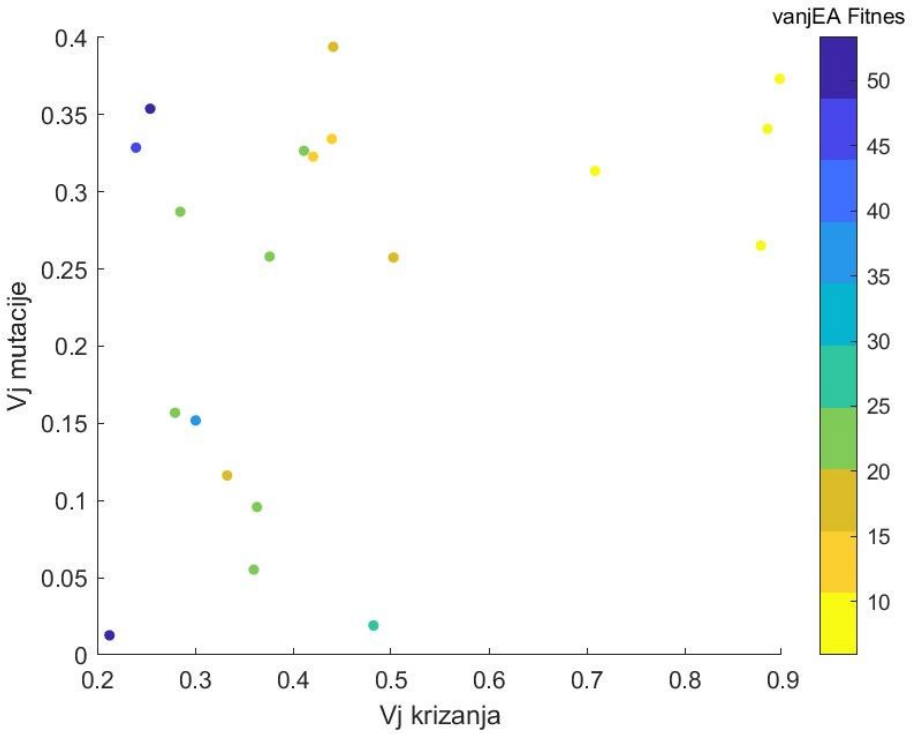
Slika 49. Primjer vektora iteracija (problem trgovačkog putnika)

Nakon svih izvođenja unutarnjeg evolucijskog algoritma s jednom kombinacijom hiperparametara, potrebno je navedenu kombinaciju evaluirati za što se koristi aritmetička sredina svih članova vektora iteracija. S obzirom na to da se za evaluaciju pojedinca koristi isključivo vrijednost aritmetičke sredine svih članova vektora iteracija, fitnes funkcija je pronađena korištenjem eksponencijalne regresije na temelju eksperimentalno dobivenih rezultata i ona poprima sljedeći oblik:

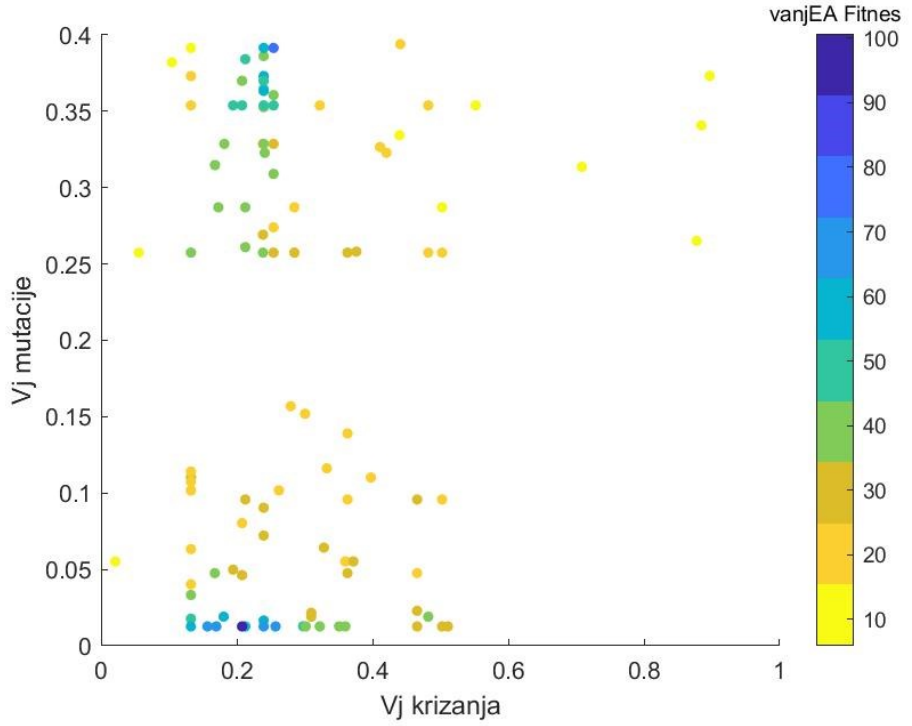
$$\Phi_2 = 8.3 \cdot 10^9 \cdot 0.8^{\frac{\bar{\mathbf{v}}}{100}}. \quad \textbf{(14)}$$

Nakon evaluacije svih članova populacije slijedi postupak selekcije i križanja koji se provodi analogno postupku prikazanom kod OneMin problema. Prilikom križanja doduše, rekombinacija može nastupiti isključivo na mjestu između vrijednosti vjerojatnosti križanja i mutacije s obzirom na to da veličinu populacije držimo konstantnom. Mutacija se također provodi analogno postupku prikazanom kod OneMin problema uz drugačije definirane podintervale. Za vjerojatnost križanja su zadani podintervali: donji [0,2, 0,3], srednji (0,3, 0,8) i gornji [0,8, 0,9], dok su za vjerojatnost mutacije zadani podintervali: donji [0,01, 0,05], srednji (0,05, 0,35) i gornji [0,35, 0,4].

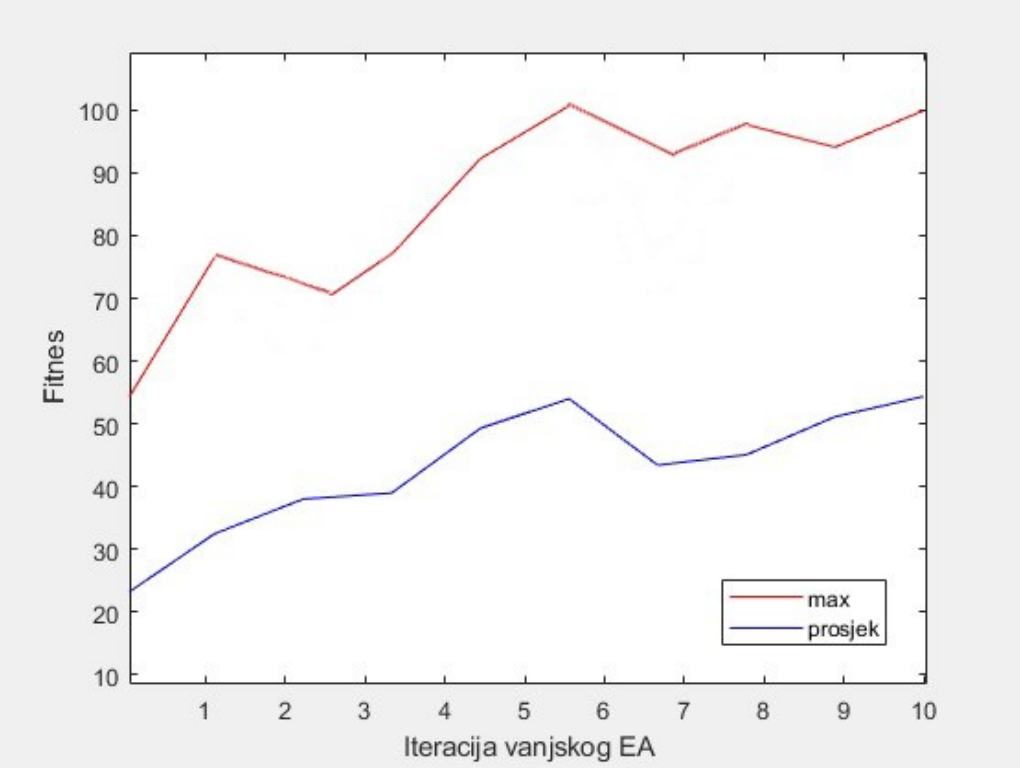
Nakon provođenja dubokog evolucijskog algoritma dobiveni su rezultati koji su prikazani na slikama 50, 51 i 52.



Slika 50. Prikaz evaluiranih kombinacija hiperparametara inicijalne populacije vanjskog evolucijskog algoritma (problem trgovačkog putnika)



Slika 51. Prikaz evaluiranih kombinacija hiperparametara nakon svih odrađenih iteracija glavne iteracijske petlje (problem trgovačkog putnika)



Slika 52. Grafički prikaz maksimalnih i prosječnih vrijednosti ostvarene dobrote za svaku iteraciju glavne iteracijske petlje (problem trgovačkog putnika)

Na slici 51 može se vidjeti da je vanjski evolucijski algoritam konvergirao dvama lokalnim optimumima koji su se oboje nalazili u području jednakog iznosa vjerojatnosti križanja. Na slici 52 može se vidjeti da prilikom prvih nekoliko iteracija vanjskog algoritma maksimalni iznos ostvarene dobrote raste nakon čega stagnira. Razlog tome je pronalazak lokalnog optimuma nakon čega preostali članovi populacije vanjskog algoritma poprimaju slične vrijednosti zbog čega se uočava jedino danji rast prosjeka ostvarene dobrote svih pojedinaca.

Prosječno ostvarena dobrota prilikom inicijalizacije iznosila je oko 25 što bi odgovaralo prosječno iskazanoj dobroti prilikom korištenja nasumične metode postavljanja hiperparametara.

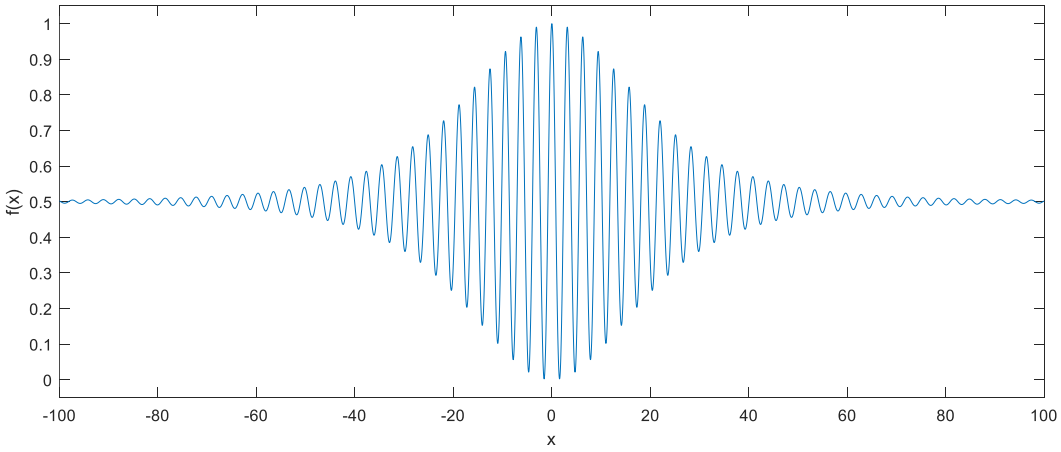
4.3. Višemodalna kontinuirana funkcija

U praksi su vrlo česti problemi kod kojih vrijednosti koje optimiramo dolaze iz višemodalne kontinuirane razdiobe. Pronalazak globalnog optimuma takvih problema može predstavljati težak zadatak za evolucijske algoritme. Naime, sam postupak evolucije usmjerava sljedeću

generaciju rješenja k trenutnom rješenju s najvećom izraženom dobrotom tako da sam postupak može vrlo lako završiti unutar lokalnog optimuma bez pronalaska globalnog optimuma. U ovom primjeru koristit će se sljedeća višemodalna funkcija:

$$f(x) = 0{,}5 - \frac{(\sin^2(x) - 0{,}5)}{(1+0{,}001x^2)^2}. \tag{15}$$

Navedena funkcija poprima oblik koji je prikazan na slici 53.



Slika 53. Prikaz korištene višemodalne kontinuirane funkcije

Globalni optimum zadane funkcije iznosi 1 te se ostvaruje za x=0. Kako bi se omogućilo korištenje operatora mutacije i križanja, ali i riješio problem neadekvatnog preslikavanja genotipskog prostora u fenotipski prostor, unutarnji evolucijski algoritam implementiran je korištenjem Grayevog koda.

4.3.1. Implementacija dubokog evolucijskog algoritma za višemodalne kontinuirane funkcije

Implementacijom dubokog evolucijskog algoritma pokušat će se pronaći kvalitetne vrijednosti hiperparametara unutarnjeg evolucijskog algoritma koji pronalazi globalni maksimum višemodalne kontinuirane funkcije. Specifično će se tražiti:

- Vrijednost vjerojatnosti križanja.
- Vrijednost vjerojatnosti mutacije.

Kao i kod implementacije dubokog evolucijskog algoritma za rješavanje problema trgovačkog putnika, veličina populacije unutarnjeg evolucijskog algoritma prethodno je definirana i iznosi 30 jedinki za svaku kombinaciju vrijednosti preostalih dvaju hiperparametara koji se postavljaju.

Vjerojatnost križanja vanjskog evolucijskog algoritma postavljena je na 0,7, a vjerojatnost mutacije na 0,2 kao što je bio slučaj i kod prethodnih primjera implementacije ove metode.

Vrijednost vjerojatnosti križanja unutarnjeg evolucijskog algoritma tražit će se unutar intervala [0,4, 0,8], a vrijednost vjerojatnosti mutacije unutarnjeg evolucijskog algoritma unutar intervala [0,01, 0,2] tako da će populacija vanjskog evolucijskog algoritma poprimiti oblik sličan onom koji je prikazan na slici 54.

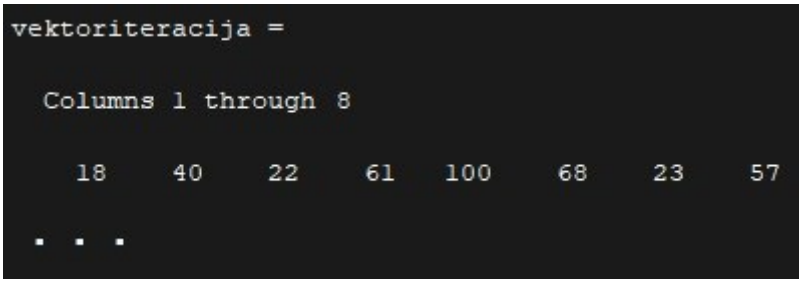
```
vanjPopulacija =

   30.0000    0.7259    0.0399
   30.0000    0.7623    0.1944
   30.0000    0.4508    0.1919
   30.0000    0.7654    0.1022
   30.0000    0.6529    0.1621
   30.0000    0.4390    0.0370
   30.0000    0.5114    0.0901
   30.0000    0.6188    0.1840
   30.0000    0.7830    0.1605
   30.0000    0.7860    0.1923
```

Slika 54. Primjer inicijalne populacije vanjskog evolucijskog algoritma (višemodalna kontinuirana funkcija)

Uvjet zaustavljanja vanjskog evolucijskog algoritma definiran je kao 10 odrađenih iteracija glavne iteracijske petlje. Unutar populacijske petlje, unutarnji evolucijski algoritam izvodi se 50 puta za svaku kombinaciju hiperparametara. Unutarnji evolucijski algoritam zaustavlja se prilikom pronalaska globalnog maksimuma ili nakon odrađenih 100 iteracija unutar kojih globalni optimum nije pronađen. Analogno OneMin problemu, u vektor iteracija se nakon

svakog izvođenja unutarnjeg evolucijskog algoritma upisuje broj iteracija unutar kojeg je globalni optimum pronađen ili se upisuje vrijednost 100 ako globalni optimum nije pronađen unutar 100 iteracija. Razlog tome već je objašnjen u prethodnom poglavlju. Primjer vektora iteracija za ovu implementaciju prikazan je na slici 55.



Slika 55. Primjer vektora iteracija (višemodalna kontinuirana funkcija)

Nakon popunjavanja vektora iteracija slijedi evaluacija pojedinca koja je definirana sljedećom funkcijom dobrote:

$$\Phi_3 = \frac{1{,}25 \cdot 10^9 \cdot L}{g \cdot \bar{\mathbf{v}}^3}. \qquad \textbf{(16)}$$

Gdje su:

L broj ponavljanja unutarnjeg evolucijskog algoritma s istim hiperparametrima

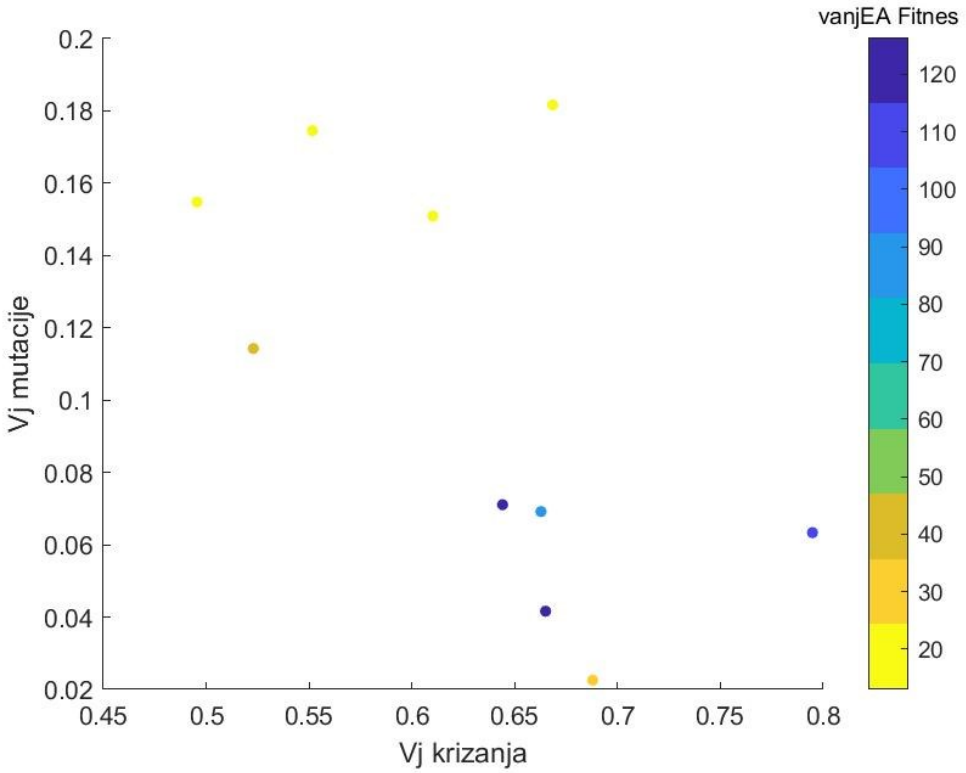
g broj izvođenja fitnes funkcije unutarnjeg algoritma

Nakon evaluacije svih članova populacije slijedi postupak selekcije, križanja i mutacije koji se provodi analogno postupku prikazanom kod prethodnih primjera implementacije. Prilikom provođenja mutacije, za vjerojatnost križanja zadani su podintervali: donji [0,4, 0,45], srednji (0,45, 0,75) i gornji [0,75, 0,8]. Dok su za vjerojatnost mutacije zadani podintervali: donji [0,01, 0,05], srednji (0,05, 0,15) i gornji [0,15, 0,2].

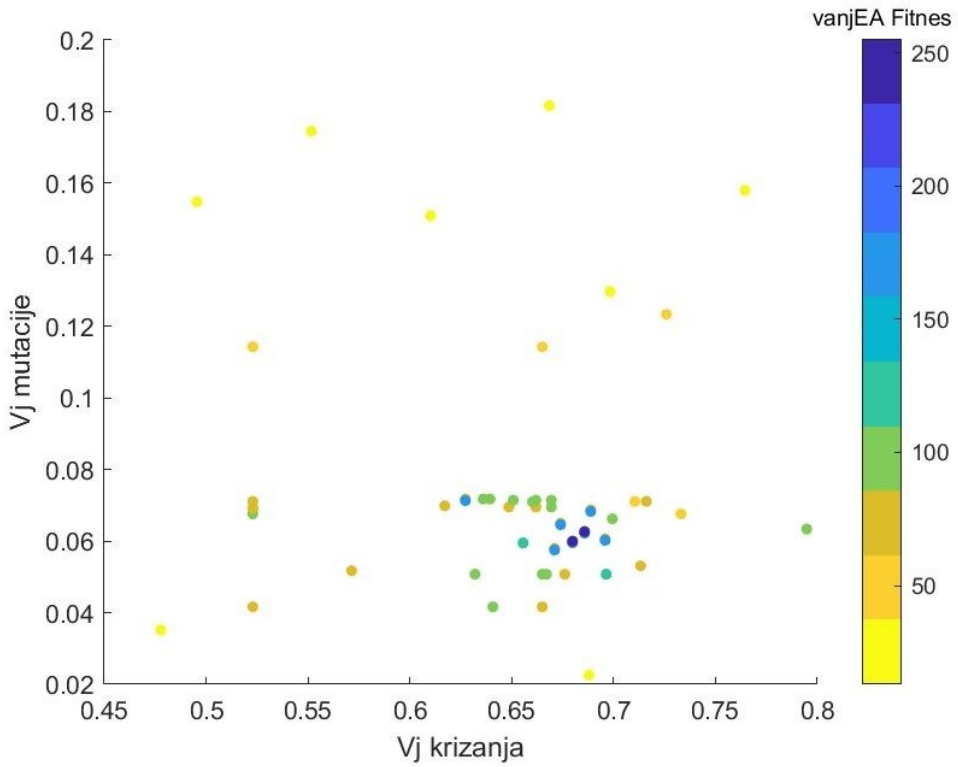
Nakon provođenja dubokog evolucijskog algoritma dobiveni su rezultati koji su prikazani na slikama 56, 57 i 58. Na slici 57 može se vidjeti da je vanjski evolucijski algoritam konvergirao prema lokalnom optimumu koji je pronađen unutar inicijalne populacije kao što

se može vidjeti na slici 56.. Na slici 57 može se vidjeti da prilikom prvih nekoliko iteracija vanjskog algoritma maksimalni iznos ostvarene dobrote raste te nakon ostvarene maksimalne vrijednosti stagnira.

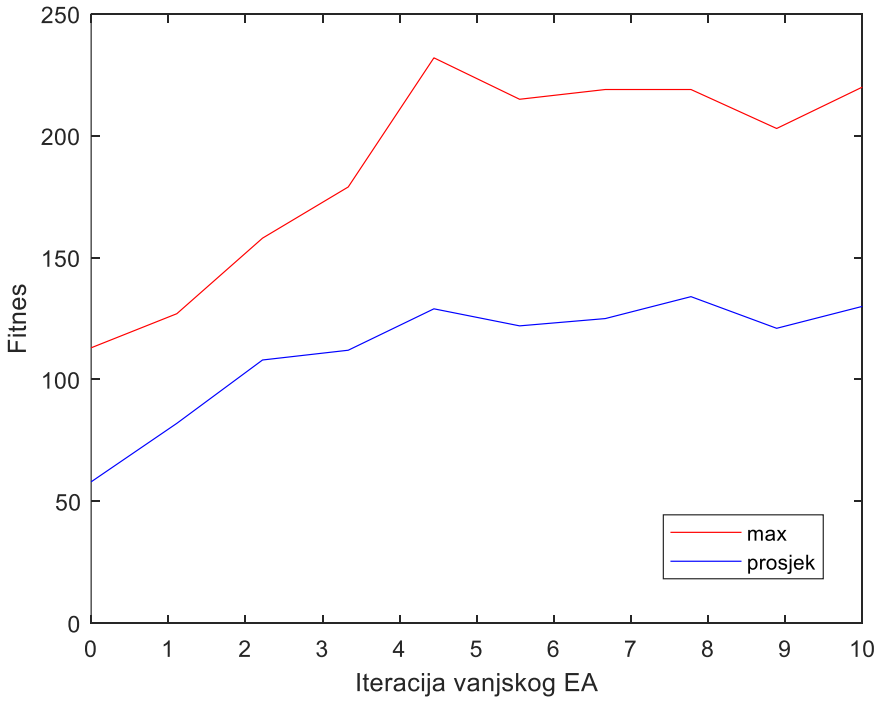
Prosječno ostvarena dobrota prilikom inicijalizacije iznosila je oko 60 što bi odgovaralo prosječno iskazanoj dobroti prilikom korištenja nasumične metode postavljanja hiperparametara. Vrijednost najveće iskazane dobrote tijekom provođenja dubokog evolucijskog algoritma iznosi oko 240 što predstavlja gotovo dupli porast u odnosu na maksimalnu iskazanu dobrotu unutar inicijalne populacije.



Slika 56. Prikaz evaluiranih kombinacija hiperparametara inicijalne populacije vanjskog evolucijskog algoritma (višemodalna kontinuirana funkcija)



Slika 57. Prikaz evaluiranih kombinacija hiperparametara nakon svih odrađenih iteracija glavne iteracijske petlje (višemodalna kontinuirana funkcija)



Slika 58. Grafički prikaz maksimalnih i prosječnih vrijednosti ostvarene dobrote za svaku iteraciju glavne iteracijske petlje (višemodalna kontinuirana funkcija)

5. ZAKLJUČAK

U ovom diplomskom radu razvijena je metoda dubokog evolucijskog algoritma temeljena na ugnježđivanju dvaju evolucijskih algoritama s ciljem pronalaska kvalitetne kombinacije hiperparametara algoritama za različite optimizacijske probleme. U radu je također dan pregled različitih struktura evolucijskih algoritama te metoda za postavljanje njihovih hiperparametara. U konačnici, razvijena metoda dubokog evolucijskog algoritma implementirana je na tri standardna ispitna zadatka te su prikazani dobiveni rezultati.

Rezultati ukazuju da razvijena metoda uistinu funkcionira te da kao takva može biti korištena za pronalazak kvalitetnih hiperparametara prethodno definirane strukture unutarnjeg evolucijskog algoritma. Stohastička priroda evolucijskih algoritama još više dolazi do izražaja u ovoj metodi s obzirom na to da se temelji na ugnježđivanju dvaju evolucijskih algoritama. Iz tog razloga bilo bi potrebno napraviti opširnu statističku analizu kako bi se navedena metoda usporedila s drugim klasičnim pristupima. Isto tako, opširna statistička analiza bila bi potrebna kako bi se odredila kvalitetna kombinacija hiperparametara i optimalna struktura vanjskog evolucijskog algoritma te utvrdilo ima li oblik unutarnjeg evolucijskog algoritma utjecaj na traženu kombinaciju hiperparametara vanjskog evolucijskog algoritma. Kako bi se provele navedene analize, bilo bi potrebno optimirati kod te analizu provesti korištenjem superračunala zbog računalne kompleksnosti ove metode i ograničene procesne moći komercijalno dostupnih računala današnjice.

LITERATURA

[1] Darwin C. On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life. London: John Murray; 1859.

[2] Edwards AWF. G.H. Hardy (1908) and Hardy–Weinberg Equilibrium. Genetics. 2008 July;179(3):1143-1150. doi. 10.1534/genetics.104.92940.

[3] Rabbers I, Van Heerden JH, Nordholt N, Bachmann H, Teusink B, Bruggeman FJ. Metabolism at Evolutionary Optimal States. Metabolites. 2015 June;5(2):311-343. doi. 10.3390/metabo5020311.

[4] Eiben AE, Smith JE. Evolutionary Computing: The Origins. 2nd ed. Berlin: Springer Berlin Heidelberg; 2015.

[5] Schwefel HP. Evolution and Optimum Seeking. New York: Wiley; 1995.

[6] Fogel LJ, Owens AJ, Walsh MJ. Artificial Intelligence through Simulated Evolution. Chichester: Wiley; 1966.

[7] Holland JH. Adaption in Natural and Artificial Systems. Cambridge: MIT Press; 1992.

[8] Koza JR. Genetic Programming. Cambridge: MIT Press; 1992.

[9] Ma L, Hu S, Qiu M, Li Q, Ji Z. Energy consumption optimization of high sulfur natural gas purification plant based on backpropagation neural network and genetic algorithms. Journal of Applied Engineering Science. 2020 July;18(3):338-345. doi. 10.5937/jaes18-25687.

[10] Yan Y, Hong L, He X, Ouyang M, Peeta S, Chen X. Pre-disaster investment decisions for strenghtening the Chinese railway system under earthquakes. Transportation Research Part E: Logistics and Transportation Review. 2017 Sep;105(C):39-59. doi. 10.1016/j.tre.2017.07.001.

[11] Chih-Chin L, Chuan-Yu C. A hierarchical evolutionary algorithm for automatic medical image segmentation. Expert Systems with Applications. 2009 Jan;36(1):248-259. doi. 10.1016/j.eswa.2007.09.003.

[12] Ćurković P. Evolutivni algoritam za upravljanje višeagentskim robotskim sustavom [doktorski rad]. Zagreb: Fakultet strojarstva i brodogradnje; 2010.

[13] Mühlenbein, H, Schlierkamp-Voosen D. Predictive Models for the Breeder Genetic Algorithm I. Continuous Parameter Optimization. Evolutionary Computation. 1993 March;1(1):25-49. doi. 10.1162/evco.1993.1.1.25.

[14] Chawdhry PK, Roy R, Pant RK. Soft Computing in Engineering Design and Manufacturing. London: Springer-Verlag London Limited; 1998.

[15] Davis LD. Genetic Algorithms and Simulated Annealing. San Mateo: Morgan Kaufmann Publishers; 1987.

[16] Rawlins GJE. Foundations of Genetic Algorithms. San Mateo: Morgan Kaufmann Publishers; 1991.

[17] The Third International Conference on Genetic Algorithms, 4th to 7th June 1989, San Mateo. 1st edition. ed. San Mateo: Morgan Kaufmann Publishers; 1989.

[18] The Fourth International Conference on Genetic Algorithms, 13th to 16th July 1991, San Mateo. 1st edition. ed. San Mateo: Morgan Kaufmann Publishers; 1991.

[19] Smit SK, Eiben AE. Comparing parameter tuning methods for evolutionary algorithms. IEEE Congress on Evolutionary Computation, Trondheim 2009; Trondheim Norway, 18.05.-21.05.2009. Trondheim: Nova Conference Centre and Cinema; 2009.

[20] Wolpert DH, Macready WG. No free lunch theorems for optimization. IEEE Transaction on Evolutionary Computation. 1997 May;1(1):67–82. doi. 10.1109/4235.585893.

[21] Krishnaiah K, Shahabudeen P. Applied Design of Experiments and Taguchi Methods. New Delhi: PHI Learning Private Limited; 2012.

[22] Myers R, Hancock ER. Empirical modelling of genetic algorithms. Evolutionary Computation. 2001 Dec;9(4):461-493. doi. 10.1162/10636560152642878.

[23] Sipper M, Fu W, Ahuja K, Moore JH. Investigating the parameter space of evolutionary algorithms. BioData Mining. 2018 Feb;11(2):15-28. doi. 10.1186/s13040-018-0164-x.

PRILOZI

I. CD-R disk

II. MATLAB kodovi

PRILOG II.

MATLAB kodovi

1. OneMin

```
%%%%%%%%%% ONEMIN %%%%%%%%%%

clear
clc

tStart=tic;

vanjSUM=0;

vanjB= 20;
BB = 20 + 180*rand(vanjB,1);
vanjL = 40;
vanjpc=0.7;
vanjpm=0.2;

A=20;
L=200;
ponavljanje=1000;

vanjPopulacija=[BB-mod(BB,2), 0.4 + 0.4*rand(vanjB,1), 0.01 +
0.19*rand(vanjB,1)]



for MMM=1:vanjL

    for clan=1:vanjB
        tic;
        B=vanjPopulacija(clan,1);
        pc=vanjPopulacija(clan,2);
        pm=vanjPopulacija(clan,3);
        vektoriteracija=ones(1,ponavljanje)*L;
        brojevaluacijafitnesa=0;

        for c=1:ponavljanje
            Populacija=randi([0,1],B,A);
            SUM=0;
            clear Novo Fitnes
            %---------POÈETAK UNUTARNJEG EVOL ALGORITAMA (ONEMIN)---------%


            for MM=1:L

                for i=1:B
                    Fitnes(i)=A - sum(Populacija(i,:));
```

```
                    brojevaluacijafitnesa= brojevaluacijafitnesa + 1;
                end

                [el1,el2]=max(Fitnes);

                if el1==A
                    vektoriteracija(c)=MM;
                    break
                end


                Vjerojatnost=Fitnes/sum(Fitnes);

                %Biranje roditelja
                for g=1:B
                    r=rand;
                    for k=1:B
                        SUM=SUM+Vjerojatnost(k);
                        if SUM>=r
                            Rod(g,:)=Populacija(k,:);
                            SUM=0;
                            break
                        end
                    end
                end

                %Križanje
                for h=2:2:B
                    PAR_Trenutni=Rod(h-1:h,:);
                    kk=rand;

                    if kk<=pc
                        QQ=round(1+(A-2)*rand);
                        PAR_Trenutni_krizani=[[PAR_Trenutni(1,1:QQ)
PAR_Trenutni(2,QQ+1:A)];[PAR_Trenutni(2,1:QQ) PAR_Trenutni(1,QQ+1:A)]];

                    else
                        PAR_Trenutni_krizani=PAR_Trenutni;
                    end
                    Novo(h-1:h,:)=[PAR_Trenutni_krizani];
                end

                %Mutacija
                mutm=rand(B,A);
                Novoprijemutacije=Novo;
                Novo(mutm<=pm & Novoprijemutacije==1)=0;
                Novo(mutm<=pm & Novoprijemutacije==0)=1;

                %Elitizam
                rand_el=round(1+rand*(B-1));
                Novo(rand_el,:)=Populacija(el2,:);
                Populacija=Novo;


            end

            %---------ZAVRŠETAK UNUTARNJEG EVOL ALGORITMA (ONEMIN)--------%
```

```
        end
        %ispis rezultata i evaluacija
        vrijeme=toc
        brojevaluacijafitnesa
        provjeramediana=median(vektoriteracija)
        broj200=nnz(vektoriteracija==200)
        vanjFitnes(clan)=
(1.2500e+10)*ponavljanje/(brojevaluacijafitnesa*(mean(vektoriteracija))^3);
        FITNESTRENUTNOGCLANA=vanjFitnes(clan)
        TRENUTNICLAN=clan
        VANJSKAITERACIJA=MMM
    end
    %vanjEA
    FITVANJSKI=vanjFitnes
    PROSJEKVANJFITNESA(MMM)=mean(vanjFitnes)
    vanjFIT(MMM)=max(vanjFitnes);
    [vanjel1,vanjel2]=max(vanjFitnes);
    vanjPopulacija
    vanjPopulacija(vanjel2,:)

    vanjVjerojatnost=vanjFitnes/sum(vanjFitnes);

    %3D PLOT
    ff=figure(1)
    scatter3(vanjPopulacija(:,1),vanjPopulacija(:,2),vanjPopulacija(:,3),
20, FITVANJSKI, 'filled')
    hold on
    xlabel('Vel populacije')
    ylabel('Vj krizanja')
    zlabel('Vj mutacije')
    cb = colorbar();
    colormap (flipud(parula(10)))
    title(cb, 'vanjEA Fitnes')
    saveas(ff, sprintf('a%d.jpg',MMM));
    drawnow

    %Biranje roditelja
    for gg=1:vanjB
        rr=rand;
        for pp=1:vanjB
            vanjSUM=vanjSUM+vanjVjerojatnost(pp);
            if vanjSUM>=rr
                RodRod(gg,:)=vanjPopulacija(pp,:);
                vanjSUM=0;
                break
            end
        end
    end

    %Križanje
    for hh=2:2:vanjB
        vanjPAR_Trenutni=RodRod(hh-1:hh,:);
        kkuku=rand;

        if kkuku<=vanjpc
```

```
            QQ=round(1+rand);
            vanjPAR_Trenutni_krizani=[[vanjPAR_Trenutni(1,1:QQ)
vanjPAR_Trenutni(2,QQ+1:3)];[vanjPAR_Trenutni(2,1:QQ)
vanjPAR_Trenutni(1,QQ+1:3)]];

        else
            vanjPAR_Trenutni_krizani=vanjPAR_Trenutni;
        end
        vanjNovo(hh-1:hh,:)=[vanjPAR_Trenutni_krizani];
    end

    %Mutacija
    vanjmutm=rand(vanjB,3);

    %za prvi stupac, odnosno B parametar
    vanjmutmB1=find(vanjmutm(:,1)<=vanjpm & vanjNovo(:,1)>=180);
    vanjmutmB2=find(vanjmutm(:,1)<=vanjpm & vanjNovo(:,1)<180 &
vanjNovo(:,1)>40);
    vanjmutmB3=find(vanjmutm(:,1)<=vanjpm & vanjNovo(:,1)<=40);

    for iB1=1:length(vanjmutmB1)
        randB1=-100+100*rand;
        vanjNovo(vanjmutmB1(iB1),1)=vanjNovo(vanjmutmB1(iB1),1)+randB1-
mod(randB1,2);
    end

    for iB2=1:length(vanjmutmB2)
        randB2=-20+40*rand;
        vanjNovo(vanjmutmB2(iB2),1)=vanjNovo(vanjmutmB2(iB2),1)+randB2-
mod(randB2,2);
    end

    for iB3=1:length(vanjmutmB3)
        randB3=100*rand;
        vanjNovo(vanjmutmB3(iB3),1)=vanjNovo(vanjmutmB3(iB3),1)+randB3-
mod(randB3,2);
    end

    %za drugi stupac, odnosno pc parametar
    vanjmutmpc1=find(vanjmutm(:,2)<=vanjpm & vanjNovo(:,2)>=0.75);
    vanjmutmpc2=find(vanjmutm(:,2)<=vanjpm & vanjNovo(:,2)>0.45 &
vanjNovo(:,2)<0.75);
    vanjmutmpc3=find(vanjmutm(:,2)<=vanjpm & vanjNovo(:,2)<=0.45);

    for ipc1=1:length(vanjmutmpc1)
        randpc1=-0.2*rand;

vanjNovo(vanjmutmpc1(ipc1),2)=vanjNovo(vanjmutmpc1(ipc1),2)+randpc1;
    end

    for ipc2=1:length(vanjmutmpc2)
        randpc2=-0.05+0.1*rand;

vanjNovo(vanjmutmpc2(ipc2),2)=vanjNovo(vanjmutmpc2(ipc2),2)+randpc2;
    end

    for ipc3=1:length(vanjmutmpc3)
        randpc3=0.2*rand;
```

```
vanjNovo(vanjmutmpc3(ipc3),2)=vanjNovo(vanjmutmpc3(ipc3),2)+randpc3;
    end

    %za treæi stupac, odnosno pm parametar
    vanjmutmpm1=find(vanjmutm(:,3)<=vanjpm & vanjNovo(:,3)>=0.15);
    vanjmutmpm2=find(vanjmutm(:,3)<=vanjpm & vanjNovo(:,3)>0.05 &
vanjNovo(:,2)<0.15);
    vanjmutmpm3=find(vanjmutm(:,3)<=vanjpm & vanjNovo(:,3)<=0.05);

    for ipm1=1:length(vanjmutmpm1)
        randpm1=-0.05*rand;

vanjNovo(vanjmutmpm1(ipm1),3)=vanjNovo(vanjmutmpm1(ipm1),3)+randpm1;
    end

    for ipm2=1:length(vanjmutmpm2)
        randpm2=-0.04+0.04*rand;

vanjNovo(vanjmutmpm2(ipm2),3)=vanjNovo(vanjmutmpm2(ipm2),3)+randpm2;
    end

    for ipm3=1:length(vanjmutmpm3)
        randpm3=0.05*rand;

vanjNovo(vanjmutmpm3(ipm3),3)=vanjNovo(vanjmutmpm3(ipm3),3)+randpm3;
    end

    %Elitizam
    vanjrand_el=round(1+rand*5);
    vanjNovo(vanjrand_el,:)=vanjPopulacija(vanjel2,:);
    vanjPopulacija=vanjNovo;
end

hold off

%Grafièki prikaz
figure(2)
XXX=linspace(0,vanjL,vanjL);
YYY=vanjFIT;
plot(XXX,YYY,'r');
xlabel ('Iteracija vanjskog EA');
ylabel ('Fitnes');
axis([0 vanjL 0 50])

hold on

YYY2=PROSJEKVANJFITNESA;
plot(XXX,YYY2,'b');

legend('max','prosjek')

hold off

toc(tStart)
```

2. TSP

```
%%%%%%%%%% TSP %%%%%%%%%%

clear
clc

tStart=tic;

vanjSUM=0;

vanjB=20;
BB = 50*ones(vanjB,1);
vanjL = 10;
vanjpc=0.7;
vanjpm=0.2;

GR=30;
NI=3000;
ponavljanje=100;

Udaljenost =1000*rand(GR,2);

za_crtanje=10e20;
vanjPopulacija=[BB , 0.2 + 0.7*rand(vanjB,1), 0.01 + 0.39*rand(vanjB,1)]


for MMM=1:vanjL

    for clan=1:vanjB
        tic;
        G=vanjPopulacija(clan,1);
        pc=vanjPopulacija(clan,2);
        pm=vanjPopulacija(clan,3);
        vektoriteracija=ones(1,ponavljanje);

        for c=1:ponavljanje
            for i=1:G
                Populacija(i,:)=randperm(GR);
            end
            SUM=0;
            clear Novo Fitnes
            %---------POÈETAK UNUTARNJEG EVOL ALGORITAM (TSP)-----------%

            for ops=1:NI


                for h=1:G
                    Ras_trenut=Populacija(h,:);
                    Ras_trenut(GR+1)=Ras_trenut(1);
                    for j=1:GR+1
                        a=Ras_trenut(j);
                        Pop_Nova(j,:)=Udaljenost(a,:);

                    end
```

```
                    d1=0;
                    for k=1:GR
                        d1=d1+sqrt((Pop_Nova(k,1)-
Pop_Nova(k+1,1))^2+(Pop_Nova(k,2)-Pop_Nova(k+1,2))^2);
                    end


                    D1(h,1)=d1;
                    kaoo=min(D1);


                end

                kao=min(D1);

                [TE,Ii]=min(D1);

                A=sum(D1);


                AA=A-D1;
                YY=sum(AA);
                YYY=AA/YY;
                E(ops)=kao;

                Nn=min(E);

                if kao <= Nn

                    Minimum=kao;

                    Najnajbolji=Populacija(Ii,:);

                    for opq=1:GR
                        qa=Najnajbolji(opq);
                        Najbolji(opq,:)=Udaljenost(qa,:);

                    end

                end


                for trt=1:G

                    EL=D1(trt);
                    if EL==kao
                        Rod_Elitizam=Populacija(trt,:);
                    end
                end
                EL=0;

                Sum=0;
```

```
for gh=1:G
    vv=rand(1,1);
    for bb=1:G

        Sum=Sum+YYY(bb);

        if Sum>=vv
            Rod(gh,:)=Populacija(bb,:);
            Sum=0;
            break
        end

    end
end
Rod;

Rod(1,:)=Rod_Elitizam;

%Krizanje
for rr=2:2:G
    tt=rr-1;
    PAR_Trenutni=Rod(tt:rr,:);

    rnd=rand(1,1);


    if rnd <= pc

        BP1=2+(GR/2-2)*rand(1,1);
        BP2=(GR/2)+1+(GR-((GR/2)+2))*rand(1,1);
        bp1=round(BP1);
        bp2=round(BP2);

        P1=PAR_Trenutni(1,:);

        P2=PAR_Trenutni(2,:);

        AAA=P1(bp1:bp2);
        BBB=P2(bp1:bp2);

        P1_=P1;
        P2_=P2;

        for iii=1:(1+bp2-bp1)

            c1=AAA(iii);
            c2=BBB(iii);

            for jjj=1:GR

                c3=P1(jjj);

                if c3==c2

                    c4=jjj;
                    c5=P1(c4);
```

```
                    for kkk=1:GR



                        c6=P1(kkk);

                        if c1==c6

                            c7=kkk;
                            c8=P1(c7);
                            P1(c7)=c5;
                            P1(c4)=c8;
                            break

                        end
                    end

                    break

                end
            end
        end

        for iiii=1:(1+bp2-bp1)

            C1=AAA(iiii);
            C2=BBB(iiii);

            for jjjj=1:GR

                C3=P2(jjjj);

                if C3==C1

                    C4=jjjj;
                    C5=P2(C4);

                    for kkkk=1:GR

                        C6=P2(kkkk);

                        if C2==C6

                            C7=kkkk;
                            C8=P2(C7);
                            P2(C7)=C5;
                            P2(C4)=C8;

                            break

                        end
                    end

                    break
```

```
                    end
                end
            end

            PAR_Trenutni_krizani=[P1;P2];

        else

            Potomak=PAR_Trenutni;
            PAR_Trenutni_krizani=Potomak;
        end

        Novo(tt:tt+1,:)=[PAR_Trenutni_krizani];

    end
    Novo_Trenutno=Novo;

    %Mutacija
    for jok=1:G
        for jog=1:GR-1

            MUT=rand(1,1);
            if MUT <= pm

                Novo_Trenutno(jok,jog)=Novo(jok,jog+1);
                Novo_Trenutno(jok,jog+1)=Novo(jok,jog);
                break
            end

        end
    end


    Populacija=Novo_Trenutno;
    Populacija(1,:)=Rod_Elitizam;

    if  Nn<za_crtanje

        za_crtanje=Nn;


    E(ops);
        Najbolji2=Najbolji;
        figure(1);

        Najbolji2(GR+1,:)=Najbolji(1,:);

        xn2=Najbolji2(:,1);
        yn2=Najbolji2(:,2);
        plot(xn2,yn2,'-ks','linewidth',3);
        drawnow
    end

%%%%%%%%%%
end
```

```
            figure(2);
            xxx=linspace(1,ops,ops);
            yyy=E;
            plot (xxx,yyy)


            ylabel ('Ukupni prevaljeni put [km]')
            xlabel ('Generacija')
            title ('Rjesenje TSP problema')
            grid on

            %%%%%%%%%%
            figure(3);
            x=Udaljenost(:,1);
            y=Udaljenost(:,2);
            xx=Pop_Nova(:,1);
            yy=Pop_Nova(:,2);
            xxxx=Najbolji(:,1);
            yyyy=Najbolji(:,2);


            %%%%%%%%%%
            Najbolji1=Najbolji;
            Najbolji1(GR+1,:)=Najbolji(1,:);

            xn1=Najbolji1(:,1);
            yn1=Najbolji1(:,2);


            plot(x,y,'--rs',xn1,yn1,'linewidth',3);

            ylabel ('Y Udaljenost [km]')
            xlabel ('X Udaljenost [km]')
            title ('Prikaz razmjestaja gradova')

            save Udaljenost.txt Udaljenost -ascii
            %---------ZAVRŠETAK UNUTARNJEG EVOL ALGORITMA (TSP)---------%

        vektoriteracija(1,c)=Nn;

        end
        %ispis rezultata i evaluacija
        vrijeme=toc
        provjerameana=mean(vektoriteracija)
        vanjFitnes(clan)= (8.3e+9)*0.8^(provjerameana/100);
        FITNESTRENUTNOGCLANA=vanjFitnes(clan)
        TRENUTNICLAN=clan
        VANJSKAITERACIJA=MMM
    end
    %vanjEA
    FITVANJSKI=vanjFitnes
    PROSJEKVANJFITNESA(MMM)=mean(vanjFitnes)
    vanjFIT(MMM)=max(vanjFitnes);
    [vanjel1,vanjel2]=max(vanjFitnes);
    vanjPopulacija
    vanjPopulacija(vanjel2,:)
```

```
    vanjVjerojatnost=vanjFitnes/sum(vanjFitnes);

    %2D PLOT
    ff=figure(4)
    scatter(vanjPopulacija(:,2),vanjPopulacija(:,3), 20, FITVANJSKI,
'filled')
    hold on

    xlabel('Vj krizanja')
    ylabel('Vj mutacije')
    cb = colorbar();
    colormap (flipud(parula(10)))
    title(cb, 'vanjEA Fitnes')
    saveas(ff, sprintf('b%d.jpg',MMM));
    drawnow

    %Biranje roditelja
    for gg=1:vanjB
        rr=rand;
        for pp=1:vanjB
            vanjSUM=vanjSUM+vanjVjerojatnost(pp);
            if vanjSUM>=rr
                RodRod(gg,:)=vanjPopulacija(pp,:);
                vanjSUM=0;
                break
            end
        end
    end

    %Križanje
    for hh=2:2:vanjB
        vanjPAR_Trenutni=RodRod(hh-1:hh,:);
        kkuku=rand;

        if kkuku<=vanjpc
            vanjPAR_Trenutni_krizani=[[vanjPAR_Trenutni(1,1:2)
vanjPAR_Trenutni(2,3)];[vanjPAR_Trenutni(2,1:2) vanjPAR_Trenutni(1,3)]];

        else
            vanjPAR_Trenutni_krizani=vanjPAR_Trenutni;
        end
        vanjNovo(hh-1:hh,:)=[vanjPAR_Trenutni_krizani];
    end

    %Mutacija
    vanjmutm=rand(vanjB,2);

    %za drugi stupac, odnosno pc parametar
    vanjmutmpc1=find(vanjmutm(:,1)<=vanjpm & vanjNovo(:,2)>=0.8);
    vanjmutmpc2=find(vanjmutm(:,1)<=vanjpm & vanjNovo(:,2)>0.3 &
vanjNovo(:,2)<0.8);
    vanjmutmpc3=find(vanjmutm(:,1)<=vanjpm & vanjNovo(:,2)<=0.3);

    for ipc1=1:length(vanjmutmpc1)
        randpc1=-0.4*rand;

vanjNovo(vanjmutmpc1(ipc1),2)=vanjNovo(vanjmutmpc1(ipc1),2)+randpc1;
```

```
    end

    for ipc2=1:length(vanjmutmpc2)
        randpc2=-0.1+0.2*rand;

vanjNovo(vanjmutmpc2(ipc2),2)=vanjNovo(vanjmutmpc2(ipc2),2)+randpc2;
    end

    for ipc3=1:length(vanjmutmpc3)
        randpc3=0.4*rand;

vanjNovo(vanjmutmpc3(ipc3),2)=vanjNovo(vanjmutmpc3(ipc3),2)+randpc3;
    end

    %za treæi stupac, odnosno pm parametar
    vanjmutmpm1=find(vanjmutm(:,2)<=vanjpm & vanjNovo(:,3)>=0.35);
    vanjmutmpm2=find(vanjmutm(:,2)<=vanjpm & vanjNovo(:,3)>0.05 &
vanjNovo(:,2)<0.35);
    vanjmutmpm3=find(vanjmutm(:,2)<=vanjpm & vanjNovo(:,3)<=0.05);

    for ipm1=1:length(vanjmutmpm1)
        randpm1=-0.2*rand;

vanjNovo(vanjmutmpm1(ipm1),3)=vanjNovo(vanjmutmpm1(ipm1),3)+randpm1;
    end

    for ipm2=1:length(vanjmutmpm2)
        randpm2=-0.04+0.08*rand;

vanjNovo(vanjmutmpm2(ipm2),3)=vanjNovo(vanjmutmpm2(ipm2),3)+randpm2;
    end

    for ipm3=1:length(vanjmutmpm3)
        randpm3=0.2*rand;

vanjNovo(vanjmutmpm3(ipm3),3)=vanjNovo(vanjmutmpm3(ipm3),3)+randpm3;
    end

    %Elitizam
    vanjrand_el=round(1+rand*5);
    vanjNovo(vanjrand_el,:)=vanjPopulacija(vanjel2,:);
    vanjPopulacija=vanjNovo;
end

hold off

%Grafièki prikaz

XXX=linspace(0,vanjL,vanjL);
YYY=vanjFIT;
plot(XXX,YYY,'r');
xlabel ('Iteracija vanjskog EA');
ylabel ('Fitnes');
axis([0 vanjL 0 50])

hold on

YYY2=PROSJEKVANJFITNESA;
```

```
plot(XXX,YYY2,'b');

legend('max','prosjek')

hold off

toc(tStart)
```

3. Višemodalna kontinuirana funkcija

```
%%%%%%%%%% VISEMODALNA KONTINUIRANA FUNKCIJA %%%%%%%%%%

clear
clc

xx=linspace(-100,100,10000);
yy=0.5-(sin(xx).^2-0.5)./(1+0.001.*xx.^2).^2;

xSlike = []
yslike = [],

tStart=tic;

vanjSUM=0;

vanjB= 10;
BB = 30*ones(vanjB,1);
vanjL = 10;
vanjpc=0.7;
vanjpm=0.2;

A=22;
L=100;
ponavljanje=50;

vanjPopulacija=[BB, 0.4 + 0.4*rand(vanjB,1), 0.01 + 0.19*rand(vanjB,1)]



for MMM=1:vanjL

    for clan=1:vanjB
        tic;
        B=vanjPopulacija(clan,1);
        pc=vanjPopulacija(clan,2);
        pm=vanjPopulacija(clan,3);
        vektoriteracija=ones(1,ponavljanje)*L;
        brojevaluacijafitnesa=0;

        for c=1:ponavljanje
            Pop=randi([0,1],B,A);
            fiti=0;
            SUM=0;
```

```
            clear Novo Fitnes
            clear fiti
            clear SUM
            %---------POÈETAK UNUTARNJEG EVOL ALGORITAM (VKF)-----------%


            for Main=1:L
                Main;
                Pop_int=gc2dec(Pop);


                for h=1:B
                    Pop_real(h,1)=(-100+(Pop_int(h,:))*(200/(2^22-1)));
                end


                for i=1:B
                     Pop_fin(i,1)=0.5-(sin(Pop_real(i,:)).^2-
0.5)/(1+0.001*Pop_real(i,:)^2)^2;
                     brojevaluacijafitnesa= brojevaluacijafitnesa + 1;
                end
                fiti(Main)=max(Pop_fin);


                for i=1:B
                    if Pop_fin(i,1)>=max(fiti)
                        Elil=Pop(i,:);
                        Ko=i;
                    end
                end

                tr=gc2dec(Elil);
                xmax=(-100+tr*(200/(2^22-1)));
                ymax=0.5-(sin(xmax)^2-0.5)/(1+0.001*xmax^2)^2;


            if max(fiti)>0.9999
                vektoriteracija(c)=Main;
                break
            end
                Suma=sum(Pop_fin);
                Vjerojatnost=Pop_fin/Suma;


                SUM=0;

                for j=1:B
                    r=rand(1,1);
                    for k=1:B
                        SUM=SUM+Vjerojatnost(k);
                        if SUM>r
                            Rod(j,:)=Pop(k,:);
                            SUM=0;
                            break
                        end
                    end
                end
```

```
                %Krizanje
                for k=2:2:B
                    z=k-1;
                    PAR_trenutni=Rod(z:k,:);

                    rr=rand(1,1);

                    if rr<=pc
                        qq=1+(A-1)*rand(1,1);
                        QQ=round(qq);

                        P1=PAR_trenutni(1,:);
                        P2=PAR_trenutni(2,:);


                        P11=[P1(1:QQ) P2(QQ+1:A)];
                        P22=[P2(1:QQ) P1(QQ+1:A)];
                        PAR_trenutni_krizani=[P11;P22];

                    else
                        Potomak=PAR_trenutni;
                        PAR_trenutni_krizani=Potomak;
                    end

                    Novo(z:k,:)=[PAR_trenutni_krizani];
                end

                %Mutacija
                for n=1:B
                    for o=1:A
                        pmut=rand(1,1);
                        if pmut<=pm & Novo(n,o)==1
                            Novo(n,o)=0;
                        elseif pmut<=pm & Novo(n,o)==0
                            Novo(n,o)=1;
                        end
                    end
                end
                Pop=Novo;
                Pop(1,:)=Elil;

            %%%%%%%%%%
            figure(1)
            plot(
xx,yy,Pop_real(:),Pop_fin(:),'ks','MarkerSize',6,'MarkerFaceColor','g'),hol
d on

plot(xmax,ymax,'k^','MarkerSize',8,'LineWidth',2,'MarkerFaceColor','m'),
hold off
            axis([-100 100 -0.05 1.05]);
            xlabel('x')
            ylabel('f(x)')
            title('Prikaz rada GA')
            text(-90,0.8,'f(x)=0.5-(sin(x)^2-
0.5)/(1+0.001*x^2)^2','FontSize',7)
            pause(.05);
            xSlike = linspace(0,Main,Main);
```

```
            %%%%%%%%%%
            figure(2)
            plot(xSlike,fiti,'b--','LineWidth',4), hold on

plot(Main,ymax,'k^','MarkerSize',8,'LineWidth',2,'MarkerFaceColor','m'),
hold off
            axis([0 L fiti(1) 1.05]);
            pause(0.05)
            end
            Pop_fin;
            %%%%%%%%%%
            figure(3);
            XXX=linspace(0,Main,Main);
            YYY=fiti;
            plot(XXX,YYY,XXX,3);
            xlabel('Generacija broj')
            ylabel('Fitnes')
            axis([0 Main 0 1.5]);
            title('Konvergencija EA')

             %---------ZAVRŠETAK UNUTARNJEG EVOL ALGORITMA (VKF)---------%

        end
        %ispis rezultata i evaluacija
        vektoriteracija
        vrijeme=toc
        brojevaluacijafitnesa
        provjeramediana=median(vektoriteracija)
        provjerameana=mean(vektoriteracija)
        broj100=nnz(vektoriteracija==100)
        vanjFitnes(clan)=
(1.2500e+9)*ponavljanje/(brojevaluacijafitnesa*(mean(vektoriteracija))^3);
        FITNESTRENUTNOGCLANA=vanjFitnes(clan)
        TRENUTNICLAN=clan
        VANJSKAITERACIJA=MMM
    end
    %vanjEA
    FITVANJSKI=vanjFitnes
    PROSJEKVANJFITNESA(MMM)=mean(vanjFitnes)
    vanjFIT(MMM)=max(vanjFitnes);
    [vanjel1,vanjel2]=max(vanjFitnes);
    vanjPopulacija
    vanjPopulacija(vanjel2,:)

    vanjVjerojatnost=vanjFitnes/sum(vanjFitnes);

    %2D PLOT
    ff=figure(4)
    scatter(vanjPopulacija(:,2),vanjPopulacija(:,3), 20, FITVANJSKI,
'filled')
    hold on

    xlabel('Vj krizanja')
    ylabel('Vj mutacije')
    cb = colorbar();
    colormap (flipud(parula(10)))
    title(cb, 'vanjEA Fitnes')
    saveas(ff, sprintf('c%d.jpg',MMM));
    drawnow
```

```
    %Biranje roditelja
    for gg=1:vanjB
        rr=rand;
        for pp=1:vanjB
            vanjSUM=vanjSUM+vanjVjerojatnost(pp);
            if vanjSUM>=rr
                RodRod(gg,:)=vanjPopulacija(pp,:);
                vanjSUM=0;
                break
            end
        end
    end

    %Križanje
    for hh=2:2:vanjB
        vanjPAR_Trenutni=RodRod(hh-1:hh,:);
        kkuku=rand;

        if kkuku<=vanjpc
            vanjPAR_Trenutni_krizani=[[vanjPAR_Trenutni(1,1:2)
vanjPAR_Trenutni(2,3)];[vanjPAR_Trenutni(2,1:2) vanjPAR_Trenutni(1,3)]];

        else
            vanjPAR_Trenutni_krizani=vanjPAR_Trenutni;
        end
        vanjNovo(hh-1:hh,:)=[vanjPAR_Trenutni_krizani];
    end

    %Mutacija
    vanjmutm=rand(vanjB,3);


    %za drugi stupac, odnosno pc parametar
    vanjmutmpc1=find(vanjmutm(:,2)<=vanjpm & vanjNovo(:,2)>=0.75);
    vanjmutmpc2=find(vanjmutm(:,2)<=vanjpm & vanjNovo(:,2)>0.45 &
vanjNovo(:,2)<0.75);
    vanjmutmpc3=find(vanjmutm(:,2)<=vanjpm & vanjNovo(:,2)<=0.45);

    for ipc1=1:length(vanjmutmpc1)
        randpc1=-0.2*rand;

vanjNovo(vanjmutmpc1(ipc1),2)=vanjNovo(vanjmutmpc1(ipc1),2)+randpc1;
    end

    for ipc2=1:length(vanjmutmpc2)
        randpc2=-0.05+0.1*rand;

vanjNovo(vanjmutmpc2(ipc2),2)=vanjNovo(vanjmutmpc2(ipc2),2)+randpc2;
    end

    for ipc3=1:length(vanjmutmpc3)
        randpc3=0.2*rand;

vanjNovo(vanjmutmpc3(ipc3),2)=vanjNovo(vanjmutmpc3(ipc3),2)+randpc3;
    end

    %za treæi stupac, odnosno pm parametar
```

```
    vanjmutmpm1=find(vanjmutm(:,3)<=vanjpm & vanjNovo(:,3)>=0.15);
    vanjmutmpm2=find(vanjmutm(:,3)<=vanjpm & vanjNovo(:,3)>0.05 &
vanjNovo(:,2)<0.15);
    vanjmutmpm3=find(vanjmutm(:,3)<=vanjpm & vanjNovo(:,3)<=0.05);

    for ipm1=1:length(vanjmutmpm1)
        randpm1=-0.05*rand;

vanjNovo(vanjmutmpm1(ipm1),3)=vanjNovo(vanjmutmpm1(ipm1),3)+randpm1;
    end

    for ipm2=1:length(vanjmutmpm2)
        randpm2=-0.04+0.04*rand;

vanjNovo(vanjmutmpm2(ipm2),3)=vanjNovo(vanjmutmpm2(ipm2),3)+randpm2;
    end

    for ipm3=1:length(vanjmutmpm3)
        randpm3=0.05*rand;

vanjNovo(vanjmutmpm3(ipm3),3)=vanjNovo(vanjmutmpm3(ipm3),3)+randpm3;
    end

    %Elitizam
    vanjrand_el=round(1+rand*5);
    vanjNovo(vanjrand_el,:)=vanjPopulacija(vanjel2,:);
    vanjPopulacija=vanjNovo;
end

hold off

%Grafièki prikaz
figure(5)
XXX=linspace(0,vanjL,vanjL);
YYY=vanjFIT;
plot(XXX,YYY,'r');
xlabel ('Iteracija vanjskog EA');
ylabel ('Fitnes');
axis([0 vanjL 0 50])

hold on

YYY2=PROSJEKVANJFITNESA;
plot(XXX,YYY2,'b');

legend('max','prosjek')

hold off

toc(tStart)
```

4. gc.dec funkcija

```
%%%%%%%%%% GRAY TO DECIMAL %%%%%%%%%%
function dec = gc2dec(gra)
```

```
if (nargin ~= 1)
   error('Error (gc2dec): must have only 1 input argument.');
end

s1 = size(gra,1);
s2 = size(gra,2);
dec = zeros(s1,1);
bin = char(zeros(1,s2));


%gray to binary conversion
for j1 = 1:1:s1
    for j2 = s2:-1:2
        temp = mod(sum(gra(j1,1:j2-1)),2);
        if temp == 1
            bin(j2) = num2str(1 - gra(j1,j2));
        else
            bin(j2) = num2str(gra(j1,j2));
        end
    end
    bin(1) = num2str(gra(j1,1));
    dec(j1,1) = bin2dec(bin);
end
```