Usporedba rada evolucijskog algoritma i algoritma roja na standardnim testnim funkcijama

Čehulić, Lovro

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Lovro Čehulić

Zagreb, 2017.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor: Student:

Doc. dr. sc. Petar Ćurković, dipl. ing. Lovro Čehulić

Zagreb, 2017.

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

Zahvaljujem se mentoru dr. sc. Petru Ćurkoviću na pruženoj podršci i korisnim savjetima tijekom izrade i pisanja ovog rada.

Lovro Čehulić

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

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

ZAVRŠNI ZADATAK

Student: **LOVRO ČEHULIĆ** Mat. br.:11714443752

Naslov rada na hrvatskom jeziku: **Usporedba rada evolucijskog algoritma i algoritma roja na standardnim testnim funkcijama**

Naslov rada na engleskom jeziku: **Comparison of evolutionary and swarm algorithm on standard benchmark test functions**

Opis zadatka:

Heurističke metode optimiranja pokazuju dobre rezultate u pronalaženju globalnih ekstrema kod nederivabilnih, multimodalnih, i nepovezanih prostora pretrage. Dodatna je prednost njihov rad s populacijom potencijalnih rješenja, čime se na prirodan način omogućuje identifikacija skupa kvalitetnih rješenja koja pripadaju Pareto fronti.

U ovom radu potrebno je implementirati i usporediti rad dva heuristička populacijska algoritma - evolucijski algoritam i algoritam roja. Usporedbu provesti na skupu standardnih testnih (*benchmark*) funkcija različite složenosti i dimenzionalnosti.

U Matlab okružju treba osigurati mogućnost izbora testne funkcije, njezinu vizualizaciju, vizualizaciju procesa optimiranja, te izbor veličine diskretizacijskog koraka. Izbor implementirati u grafičkom korisničkom sučelju (*GUI*).

Za usporedbu rada algoritama treba provesti odgovarajuću statističku analizu te temeljeno na njezinim rezultatima dati kritičko mišljenje o prednostima i nedostatcima ovih algoritama.

Zadatak zadan:

30. studenog 2016.

Zadatak zadao:

Doc. dr. sc. Petar Ćurković

Rok predaje rada:
1. rok: 24. veljače 2017.
2. rok (izvanredni): 28. lipnja 2017.
3. rok: 22. rujna 2017.

Predviđeni datumi obrane:
1. rok: 27.2. - 03.03. 2017.
2. rok (izvanredni): 30. 06. 2017.
3. rok: 25.9. - 29. 09. 2017.

v.d. predsjednik Povjerenstva:

Izv. prof. dr. sc. Branko Bauer

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
c_1	-	Kognitivni faktor ubrzanja
c_2	-	Socijalni faktor ubrzanja
f_i	-	Vrijednost fitnesa člana i
$\mathbf{g}^{best}$	-	Vektor najboljih globalnih vrijednost optimizacijskih parametara
L_i	-	Donja granica intervala
$\mathbf{p}^{best}$	-	Vektor najboljih osobnih vrijednost optimizacijskih parametara svakog člana
p_c	-	Vjerojatnost križanja
p_m	-	Vjerojatnost mutacije
s	-	Standardna devijacija
t	-	Trenutna iteracija
t_max	-	Ukupni broj iteracija
U_i	-	Gornja granica intervala
$\mathbf{v}$	-	Vektor brzine promjene optimizacijskog parametra
w	-	Faktor inercije
$\mathbf{x}$	-	Vektor optimizacijskih parametara
α	-	Faktor aritmetičkog križanja

SAŽETAK

Genetski algoritam i optimizacija rojem čestica su metaheuristički optimizacijski alati. Genetski algoritam pripada području evolucijskog računarstva, a inspiriran je evolucijom na genetskoj razini. Optimizacija rojem čestica pripada području inteligencije roja i inspirirana je rojevima kukaca i jatima ptica čija je specijalizacija problem traženja. U ovome radu, ova dva optimizacijska algoritma testirana su na raznim testnim funkcijama te su dobivena dobra rješenja uz određena ograničenja. Odabrane testne funkcije su uglavnom multimodalni, višedimenzijski minimizacijski problemi. Optimizacijom rojem čestica postignuti su rezultati visoke točnosti za probleme koji su uspješno optimizirani. Genetski algoritam dao je rezultate koji nisu tako visoke točnosti, ali su njime pronađena rješenja onih funkcija za koje Optimizacija rojem čestica nije dala rezultate.

Ključne riječi: Genetski algoritam, optimizacija rojem čestica, standardne testne funkcije.

SUMMARY

Genetic Algorithm and Particle Swarm Optimization are Metaheuristic optimization tools. Genetic Algorithm belongs to the field of evolutionary computing, and it is inspired by evolution on genetic level. Particle Swarm Optimization belong to the field of swarm inteligence inspired by swarms of insects and flocks of birds whose specialty is search problem. In this thesis, these two optimization algorithms were tested on various (benchmark) test functions and have obtained good solutions with certain restrictions. The selected test functions are mainly multimodal, multidimensional minimization problems. Particle Swarm Optimization results have been achieved with high precision on the successfully optimized problems. Genetic Algorithm has provided results with not as high accuracy, but it found solutions for those functions that Particle Swarm Optimization did not.

Key words: Genetic Algorithm, Particle Swarm Optimization, standard (benchmark) test functions.

1. UVOD

Kroz povijest čovječanstva ljudi su se uvijek bavili određenim problemima koji su zahtijevali neki oblik optimizacije kako bi se brže ili jednostavnije riješili. Danas se optimizacija koristi u širokom rasponu djelatnosti koji uključuje optimizaciju: oblika konstrukcija, putanja, proizvodnih procesa i mnoge druge. Iz tog razloga razvili su se brojne metode optimizacije odnosno algoritmi koji su nam omogućili brzo i jednostavno dobivanje optimalnih rješenja za naše potrebe.

U novije vrijeme sve se više razvijaju algoritmi koji su inspirirani prirodom, odnosno organizmima i procesima koji se mogu pronaći u prirodi. Tako su od sredine prošlog stoljeća osmišljene razne metode koje uključuju neuronske mreže, inspirirane radom neurona u ljudskom mozgu, evolucijske tehnike inspirirane procesima evolucije i inteligencija roja koja oponaša ponašanje životinje koje se skupljaju u grupe.

Pregled po poglavljima:

U drugom poglavlju ukratko su opisani optimizacijski problemi te je dan pregled metoda za njihovo rješavanje. Osim toga govori se o matematičkim funkcijama kojima se testira kvaliteta optimizacijskih algoritama i prikazane su funkcije na kojima su testirani algoritmi koji su tema ovog rada.

U trećem poglavlju govori se o evolucijskim algoritmima, njihovom razvoju koji je inspiriran raznim prirodnim procesima te svojstvima ovih algoritama. Na kraju je opisan rad jednog od najpoznatijih evolucijskih algoritma, genetskog algoritma.

Tema četvrtog poglavlja su algoritmi roja čiji se rad temelji na ponašanju životinja koje se u prirodu skupljaju u jata ili rojeve. Opisan je princip rada jednog od takvih algoritama zvanim optimizacija rojem čestica.

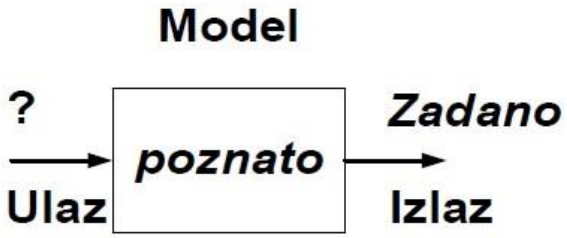
U petom poglavlju korišteni algoritmi su testirani na standardnim testnim funkcijama, opisanim u drugom poglavlju, kako bi se mogli usporediti temeljem brzine konvergencije i točnosti dobivenih rješenja.

2. MATEMATIČKO OPTIMIRANJE

U matematici i računarstvu matematičko optimiranje, ili jednostavno optimizacija, predstavlja rješavanje optimizacijskog problema. Optimiranje se razlikuje od problema modeliranja gdje unaprijed znamo ulazne i izlazne podatke, a tražimo model sustava i simuliranja gdje znamo ulazne podatke i model sustava, a tražimo izlaz. Kako bi se postigao cilj optimizacijskog problema potrebno je pronaći kombinaciju parametara (neovisnih varijabli) koji optimiziraju određenu količinu, a mogu biti podvrgnuti određenim ograničenjima unutar zadanog opsega. Količina koja se optimira (minimizira ili maksimizira) je nazvana funkcijom cilja, parametri koji se mogu promijeniti u procesu traženja optimuma zovu se kontrolne varijable. Ako se formulacija optimizacijskog problema prikaže kao slijed koraka, oni su:

1. Odabir varijabli (kontrolne varijable i varijable stanja)
2. Formuliranje ograničenja
3. Formuliranje funkcije cilja
4. Postavljanje limita varijabli
5. Odabir algoritma za rješavanje problema
6. Rješavanje problema kako bi se postiglo optimalno rješenje

Kontrolne varijable su parametri za koje se smatra da će značajno utjecati na izlaz, odabir najboljeg kompleta kontrolnih varijabli ponekad može biti izazovan zato što je teško odrediti koje varijable utječu na određeni aspekt simulacije. Domena potencijalnih vrijednosti za kontrolne varijable je uglavnom ograničena skupom ograničenja koje definira korisnik [1].



Slika 1. Prikaz optimizacijskog problema [2]

2.1. Optimizacijske tehnike

Postoji velik broj optimizacijskih algoritama koji koriste razne metode prikladne samo određenom tipu problema. Zato je važno prepoznati karakteristike problema kako bi se mogla

identificirati prikladna tehnika rješavanja. Unutar svake klase problema postoje različite metode optimizacije koje imaju različite računalne zahtjeve, svojstva konvergencije rješenja, itd. Optimizacijski problemi se klasificiraju sukladno matematičkim karakteristikama funkcije cilja, ograničenja i kontrolnih varijabli. Klasifikacija prema karakteristikama funkcije cilja prikazana je u sljedećoj tablici:

Tablica 1. Klasifikacija funkcija cilja [1]

Karakteristike	Svojstvo	Klasifikacija
Broj kontrolnih varijabli	Jedna	Jednovarijabilna
	Više od jedne	Viševarijabilna
Tip kontrolnih varijabli	Realni brojevi	Kontinuirana
	Cijeli brojevi	Diskretna
	Realni brojevi i cijeli brojevi	Miješano cjelobrojna
Funkcije problema	Linearne funkcije kontrolnih varijabli	Linearna
	Kvadratne funkcije kontrolnih varijabli	Kvadratna
	Druge nelinearne funkcije kontrolnih varijabli	Nelinearna
Formulacija problema	Ima ograničenja	Ograničena
	Nema ograničenja	Neograničena

Postoje dvije osnovne klase optimizacijskih metoda prema tipu rješenja.

a) Kriterij optimalnosti

   Analitičke metode – kada se postignu uvjeti optimalnog rješenja onda se:

   - Rješenje se testira kako bi se provjerilo da li zadovoljava uvjet.
   - Jednadžbe koje su proizašle iz kriterija optimalnosti se rješavaju analitički kako bi se utvrdilo optimalno rješenje.

b) Metode traženja

   Numeričke metode – odabere se inicijalno testno rješenje, korištenjem zdravog razuma ili nasumično, te se ocjenjuje funkcija cilja. Napravi se pomak u novu točku (drugo testno rješenje) te se funkcija cilja ponovno ocjenjuje. Ako je rješenje bolje od prethodnog, zadržava se te se radi novi korak. Proces se ponavlja dok se ne postigne optimalno rješenje. Metode traženja koriste se kada:

   - Imamo velik broj varijabli i rješenja.
   - Zadane funkcije (cilja i ograničenja) su jako nelinearne.

- Zadane funkcije (cilja i ograničenja) su zapletene u pogledu kontrolnih varijabli što otežava procjenu izvedene informacije.

U prvu skupinu spadaju takozvane konvencionalne metode u koje čiji su predstavnici: linearno programiranje, nelinearno programiranje, kvadratično programiranje, i mnoge druge. Ove metode koriste se za rješavanje specijaliziranih optimizacijskih problema, ali kod njih su prisutni određeni problemi zbog kojih su algoritmi koji koriste evolucijske tehnike popularniji.

U drugu skupinu spadaju evolucijske tehnike kod kojih početna populacija rješenja i ažuriranje informacija unutar populacije povećavaju konvergenciju prema optimalnom rješenju. Evolucijske tehnike spadaju u metaheurističke metode. Konstruirane su za brzo rješavanje problema i za traženje približnog rješenja optimizacijskog problema kada su klasične metode prespore ili ne mogu pronaći točno rješenje. To se postiže žrtvovanjem preciznosti za brzinu. Metaheuristike uzorkuju skup rješenja koji je prevelik da bi se cijeli pregledao, to odrađuju stohastički tako da se rješenja pronalaze ovisno o skupu nasumično odabranih vrijednosti [1].

2.2. Standardne testne funkcije

Kod matematičkog optimiziranja, kvaliteta optimizacijskih algoritama se često procjenjuje koristeći standardne testne funkcije. Uglavnom se ocjenjuju određena svojstva kao što su:

- Brzina konvergencije
- Preciznost
- Robusnost

Postoji više tipova funkcija ovisno o vrsti optimizacijskog problema za koji želimo ispitati algoritam. U ovome radu ispitane su testne funkcije za probleme optimizacije jedinstvene funkcije cilja. Ove testne funkcije podijeljene su po složenosti na nekoliko klasa:

- Unimodalne, konveksne, višedimenzionalne
- Multimodalne, dvodimenzionalne s malim brojem lokalnih ekstrema,
- Multimodalne, dvodimenzionalne s velikim brojem lokalnih ekstrema,
- Multimodalne, višedimenzionalne s velikim brojem lokalnih ekstrema.

Modalnost funkcije je karakteristika koja označava količinu točaka koje predstavljaju lokalne optimume, odnosno broj točaka čija je vrijednost bolja od svih njihovih susjednih točaka. Unimodalne funkcije samo jednu takvu točka koje predstavlja globalni optimum, dok multimodalne funkcije imaju više lokalnih optimuma od kojih je barem jedan ujedno i globalni.

Prva skupina sadrži zahtjevnije primjerke koji uzrokuju slabu ili sporu konvergenciju rješenja do jedinstvenog globalnog ekstrema kao i jednostavne funkcije. Druga skupina je posrednik između prve i ostalih skupina i koristi se za testiranje kvalitete standardnih optimizacijskih procesa u okolinama s malo lokalnih ekstrema i jednim globalnim. Ostale dvije skupine su preporučene za testiranje inteligentnih i robusnih optimizacijskih metoda. One se smatraju teškim problemima. Treća skupina, koja sadrži dvodimenzionalne funkcije, je u jednu ruku umjetna i koristi se za prikazivanje optimizacijskih procesa budući da se njihovo ponašanje obično jednostavno može prikazati i objasniti na 2D površini. Nažalost dvodimenzionalni problemi se vrlo rijetko pojavljuju u praksi budući da skoro svi praktični optimizacijski problemi sadrže velik broj dimenzija te spadaju u posljednju skupinu.

Sve sljedeće funkcije predstavljaju probleme minimizacije, ali neovisno o tome mogu se iskoristiti i za probleme maksimizacije jednostavnom promjenom predznaka funkcije.

2.2.1. Ackley funkcija

Ackley testna funkcija je široko korišteni multimodalni, višedimenzijski minimizacijski problem. U dvodimenzionalnom prikazu karakterizira ju naizgled ravno vanjsko područje s velikom rupom u sredini i definirana je kao:

$$f(\mathbf{x}) = -20\mathrm{e}^{-0{,}2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - \mathrm{e}^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)} + 20 + \mathrm{e} \tag{2.1}$$

Gdje *n* predstavlja broj dimenzija, a $x_i \in [-32{,}32]$ za *i* = 1, ..., *n*.

Globalni optimum: $f(\mathbf{x}) = 0$ za $x_i = 0$ za *i* = 1, ..., *n*.

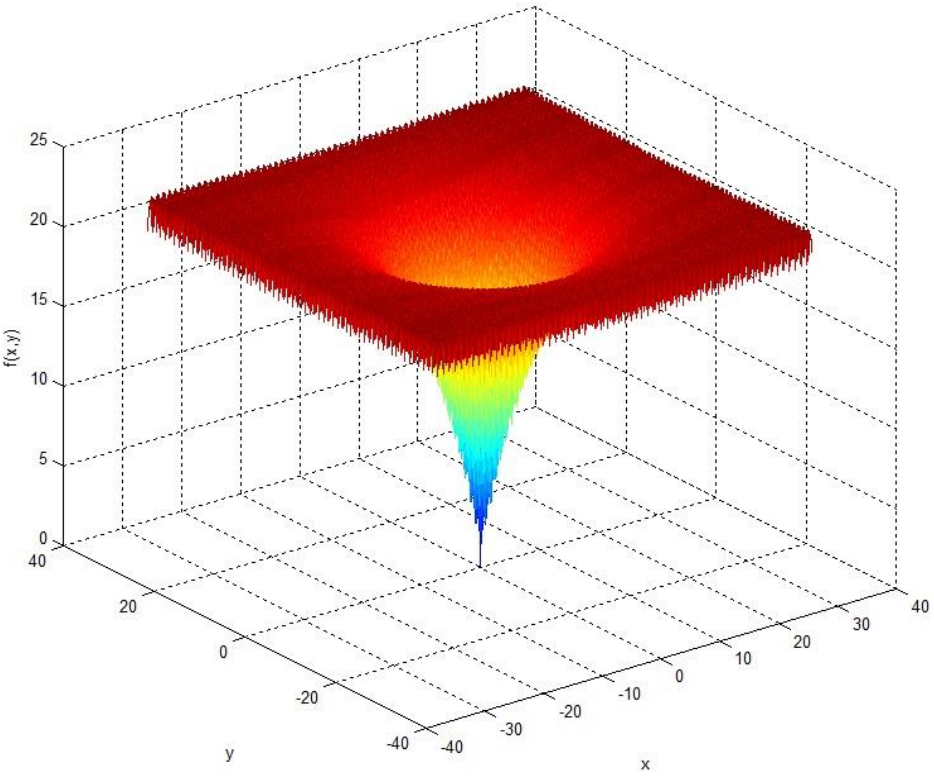
2.2.2. Rastrigin funkcija

Rastrigin testna funkcija je multimodalni, višedimenzijski minimizacijski problem i definirana je kao:

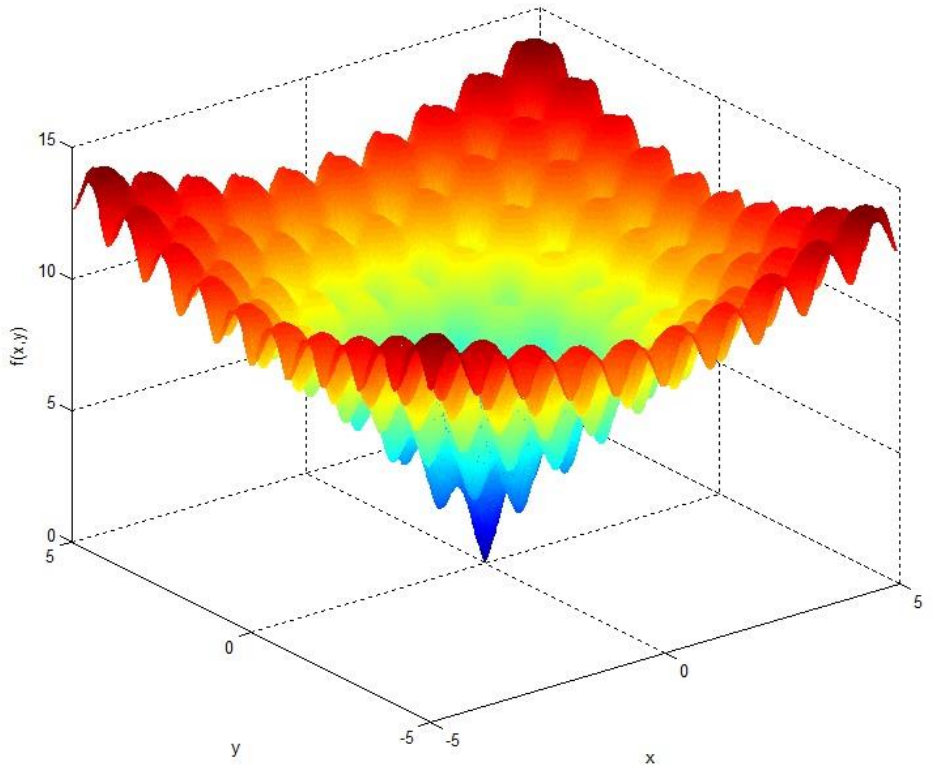
$$f(\mathbf{x}) = 10n \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)] \tag{2.2}$$

Gdje *n* predstavlja broj dimenzija, a $x_i \in [-5{,}5]$ za *i* = 1, ..., *n*.

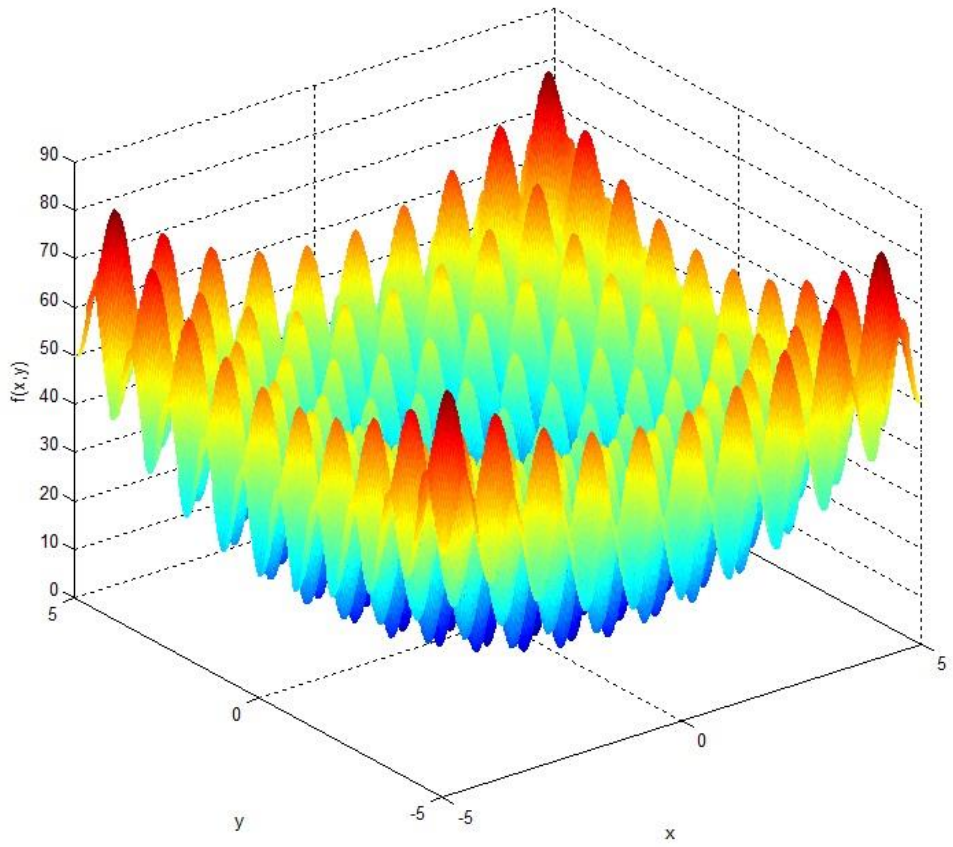
Globalni optimum: $f(\mathbf{x}) = 0$ za $x_i = 0$ za *i* = 1, ..., *n*.



Slika 2. 2D Ackley funkcija



Slika 3. Zumirana 2D Ackley funkcija



Slika 4. 2D Rastrigin funkcija

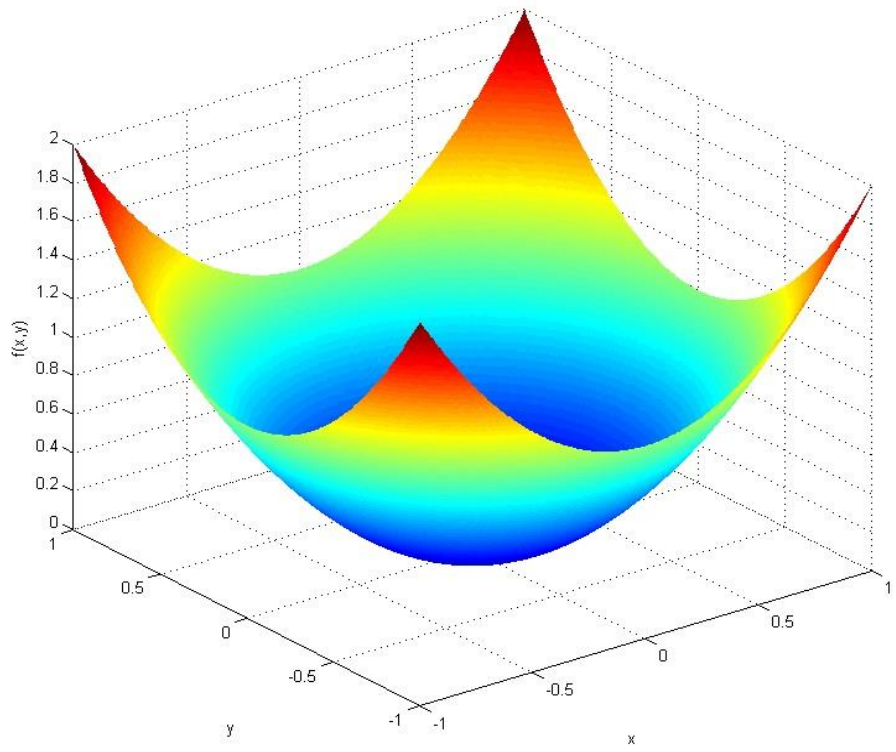
2.2.3. Sferna funkcija

Sferna testna funkcija je unimodalni, višedimenzijski minimizacijski problem koji ujedno spada i u kvadratne problem, ovo je najjednostavnija testna funkcija koja će se koristiti u ovom radu, definirana je kao:

$$f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 \tag{2.3}$$

Gdje n predstavlja broj dimenzija, a $x_i \in [-1,1]$ za $i = 1, ..., n$.

Globalni optimum: $f(\mathbf{x}) = 0$ za $x_i = 0$ za $i = 1, ..., n$.



Slika 5. 2D Sferna funkcija

2.2.4. DropWave funkcija

DropWave testna funkcija je multimodalni, dvodimenzijski minimizacijski problem i definirana je kao:

$$f(\mathbf{x}) = -\frac{1+\cos\left(12\sqrt{\sum_{i=1}^{n} x_i^2}\right)}{2+0{,}5\sum_{i=1}^{n} x_i^2} \tag{2.4}$$

Gdje *n* predstavlja broj dimenzija, a $x_i \in [-5{,}5]$ za $i = 1, 2$.

Globalni optimum: $f(\mathbf{x}) = -1$ za $x_i = 0$ za $i = 1, 2$.

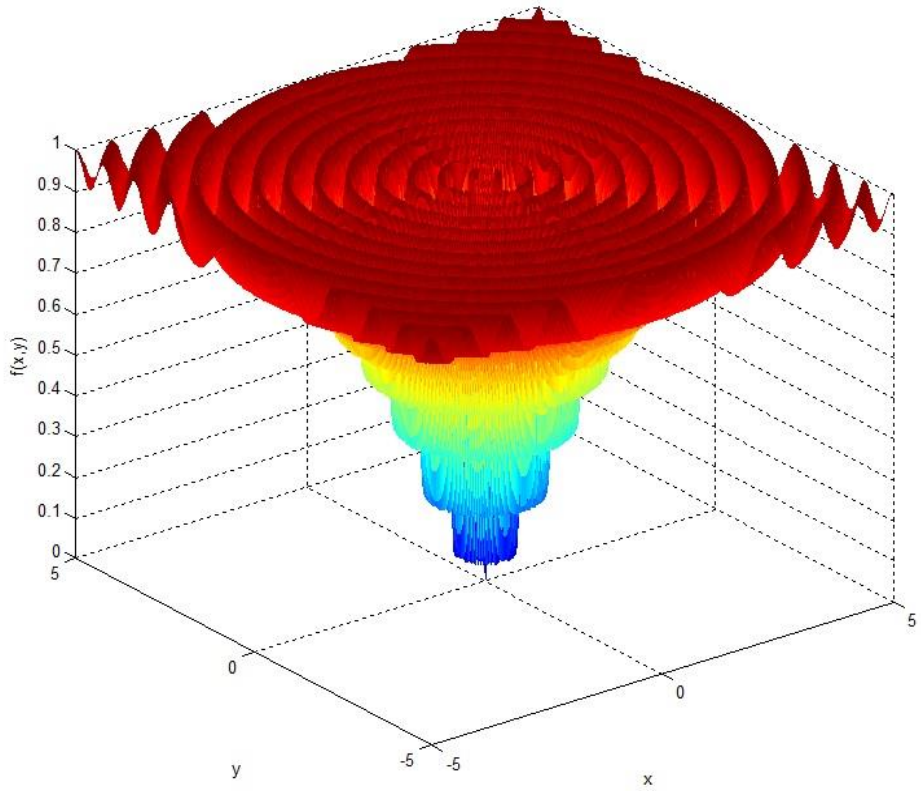
2.2.5. Easom funkcija

Easom testna funkcija je multimodalni, dvodimenzijski minimizacijski problem i karakterizira ju jako usko područje globalnog optimuma, definirana je kao:

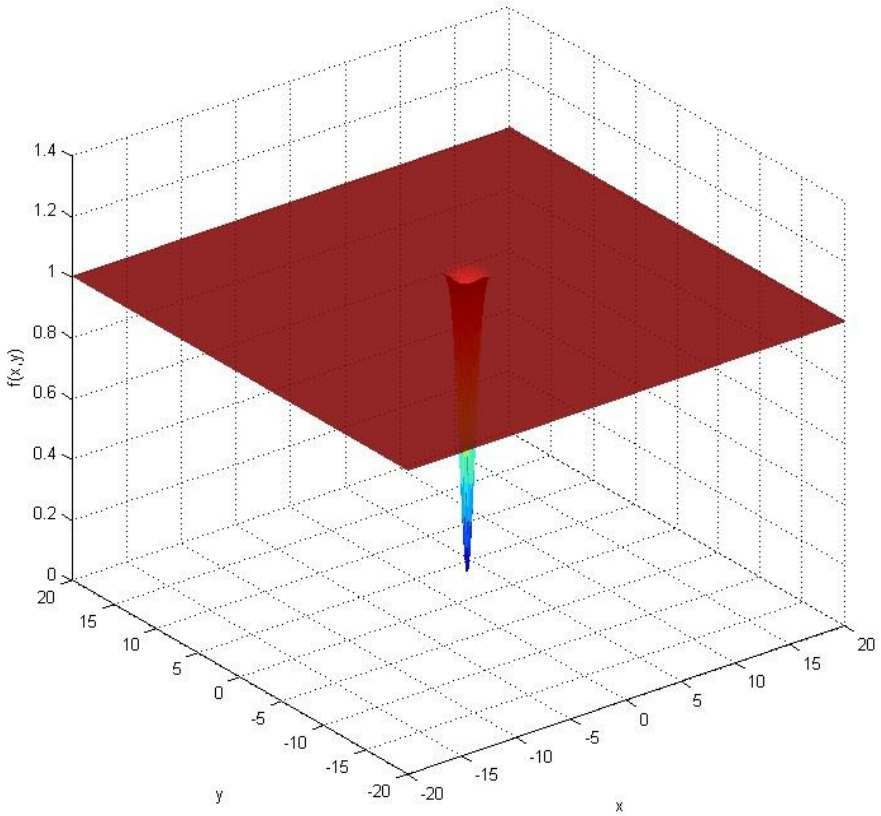
$$f(\mathbf{x}) = -\cos(x_1)\cos(x_2)\,\mathrm{e}^{\left(-(x_1-\pi)^2-(x_2-\pi)^2\right)} \tag{2.5}$$

Funkcija se obično prikazuje na kvadratu $x_i \in [-100, 100]$ za $i = 1, 2$.

Globalni optimum: $f(\mathbf{x}) = -1$ za $x_i = \pi$ za $i = 1, 2$.



Slika 6. 2D DropWave funkcija



Slika 7. Easom funkcija

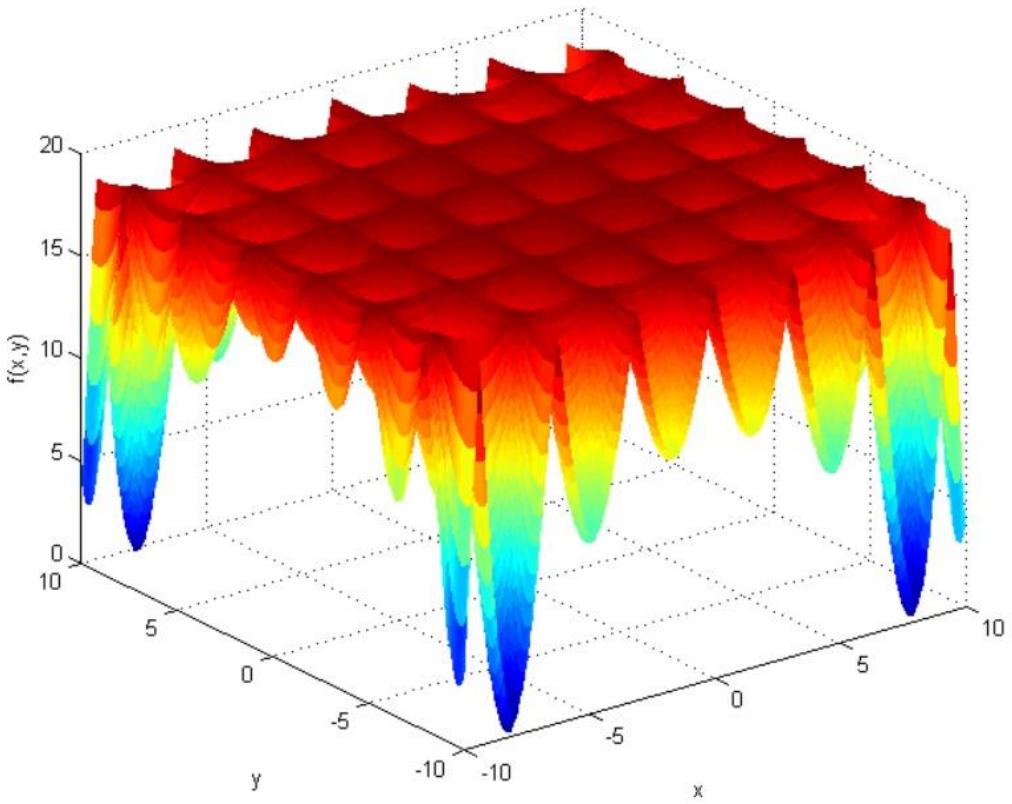
2.2.6. HolderTable funkcija

HolderTable testna funkcija je multimodalni, dvodimenzijski minimizacijski problem s četiri globalna optimuma, definirana je kao:

$$f(\mathbf{x}) = -\left|\sin(x_1)\cos(x_2)\,\mathrm{e}^{\left|1-\frac{\sqrt{x_1+x_2}}{\pi}\right|}\right| \tag{2.6}$$

Funkcija se obično prikazuje na kvadratu $x_i \in [-10, 10]$ za $i = 1, 2$.

Globalni optimum: $f(\mathbf{x}) = -19{,}2085$ za $x_1 = \pm 8{,}05502$, $x_2 = \pm 9{,}66459$



Slika 8. HolderTable funkcija

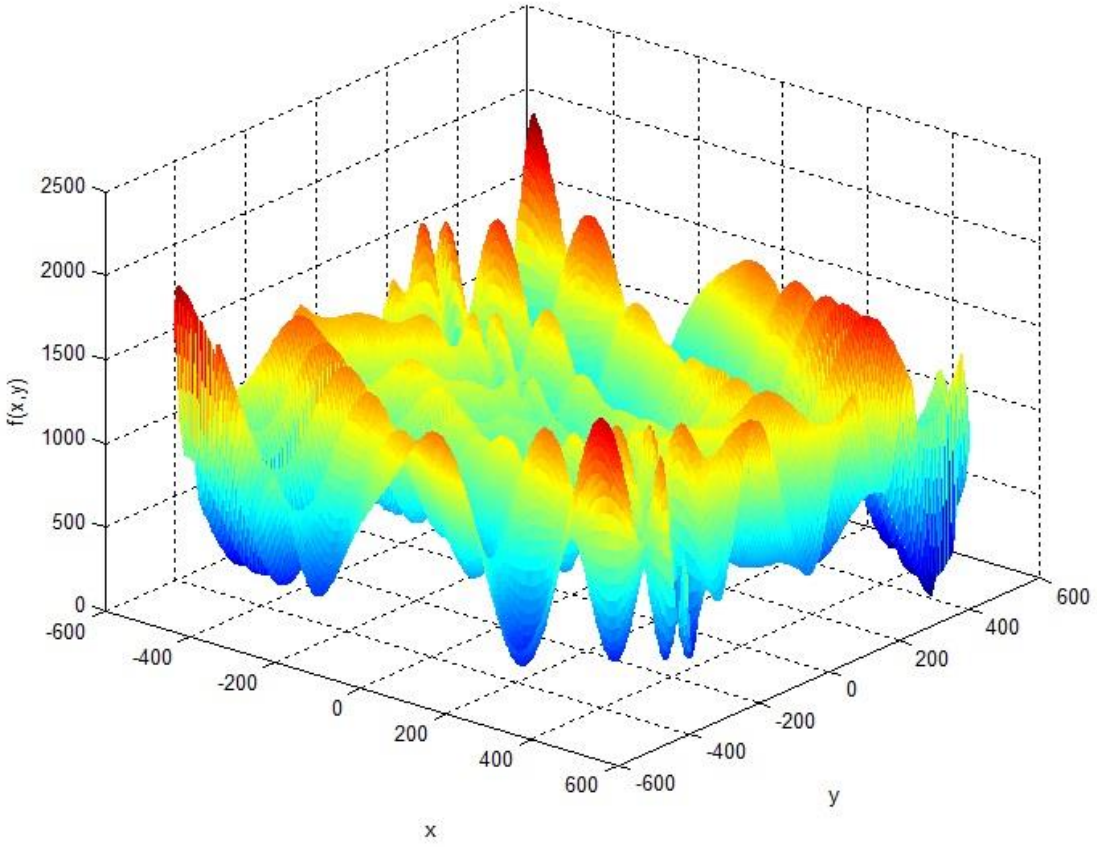
2.2.7. EggHolder funkcija

EggHolder testna funkcija je multimodalni, dvodimenzijski minimizacijski problem, definirana je kao:

$$f(\mathbf{x}) = -x_1 \sin\left(\sqrt{|x_1 - x_2 - 47|}\right) - (x_2 + 47)\sin\left(\sqrt{\left|\tfrac{1}{2}x_1 + x_2 + 47\right|}\right) \tag{2.7}$$

Funkcija se obično prikazuje na kvadratu $x_i \in [-512, 512]$ za $i = 1, 2$.

Globalni optimum: $f(\mathbf{x}) = -959{,}6407$ za $x_1 = 512$, $x_2 = 404{,}2319$



Slika 9. EggHolder funkcija

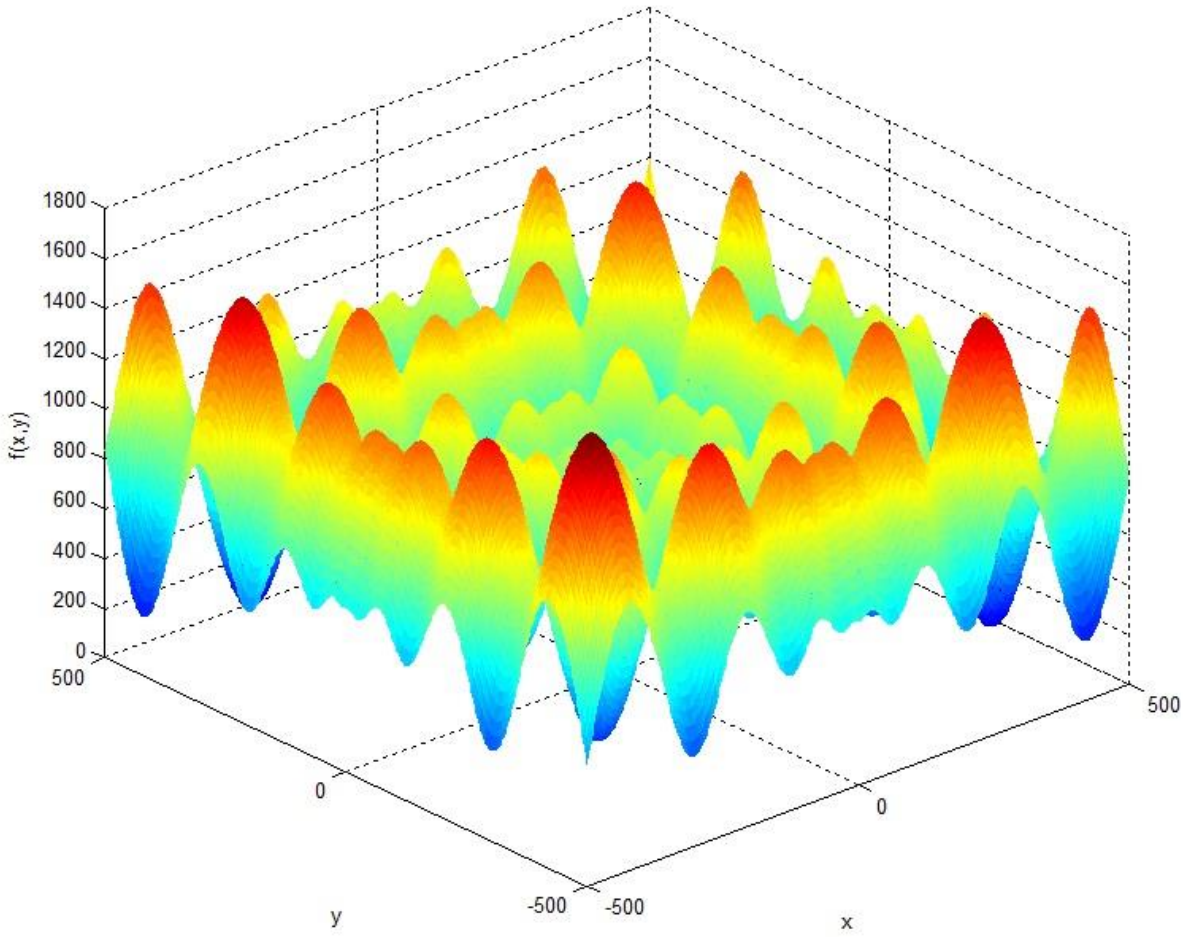
2.2.8. Schwefel26 funkcija

Schwefel26 testna funkcija je multimodalni, višedimenzijski minimizacijski problem i definirana je kao:

$$f(\mathbf{x}) = 418{,}9829n - \sum_{i=1}^{n} x_i \sin\left(\sqrt{|x_i|}\right) \tag{2.8}$$

Funkcija se obično prikazuje na kvadratu $x_i \in [-500, 500]$, za i = 1, ..., n.

Globalni optimum: $f(\mathbf{x}) = 0$, za $x_i = 420{,}9687$, za i = 1, ..., n.



Slika 10. 2D Schwefel funkcija

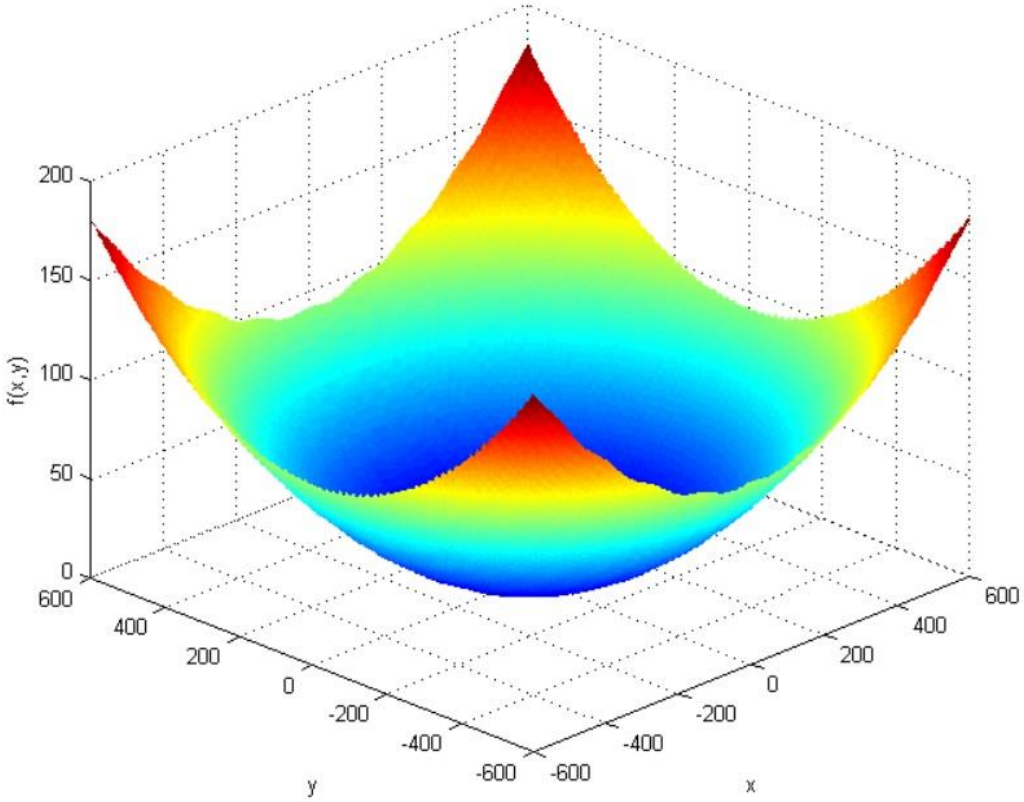
2.2.9. Griewank funkcija

Griewank testna funkcija je multimodalni, višedimenzijski minimizacijski problem i definirana je kao:

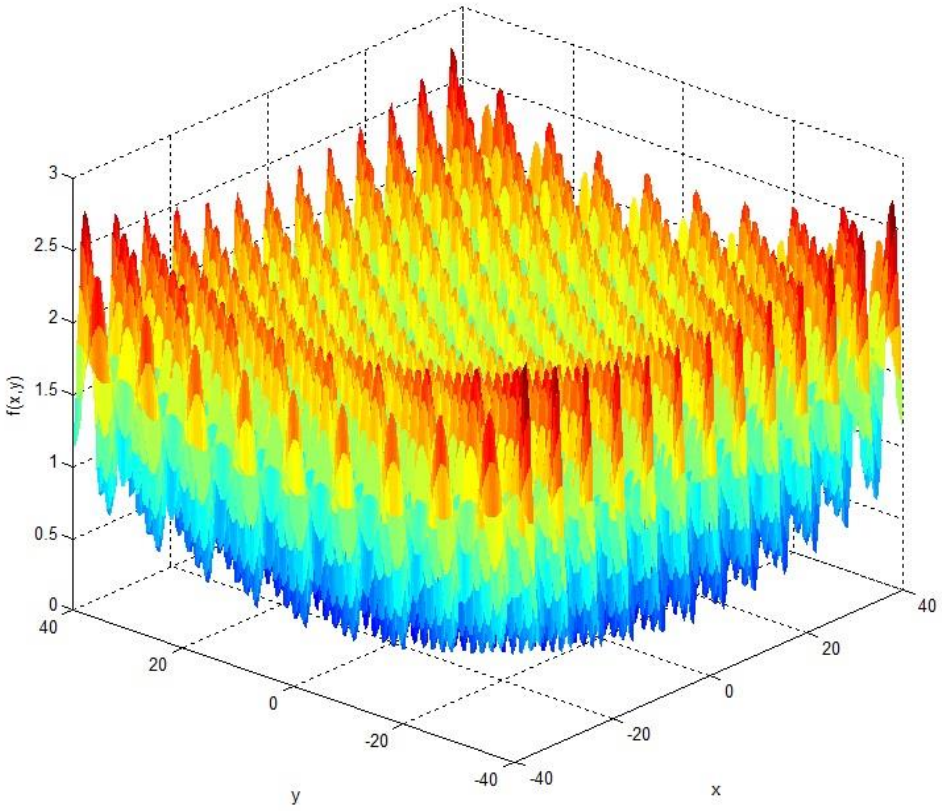
$$f(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \qquad \textbf{(2.9)}$$

Funkcija se obično prikazuje na kvadratu $x_i \in [-600, 600]$, za i = 1, ..., n.

Globalni optimum: $f(\mathbf{x}) = 0$, za $x_i = 0$, za i = 1, ..., n.



Slika 11. 2D Griewank testna funkcija



Slika 12. Zumirana 2D Griewank funkcija

3. EVOLUCIJSKI ALGORITMI

Evolucijski algoritmi pripadaju području evolucijskog računarstva koje se bavi proučavanjem računalnih metoda inspiriranih prirodom, a posebno postupcima i mehanizmima evolucije. Izbor evolucije kao izvor inspiracije nije iznenađujuć kada se u obzir uzme raznolikost biljnog i životinjskog svijeta i njegovu prilagođenost okolišu u kojemu se nalazi. Osnova evolucijskih algoritama proizlazi iz metode kojom evolucija rješava probleme, a zove se metoda pokušaja i pogrešaka [3].

3.1. Razvoj

Ideja primjene Darwinovih principa u automatskom rješavanju problema datira iz četrdesetih godina prošlog stoljeća kada je 1948. Turing predložio „genetsko ili evolucijsko traženje“. Do 1962. Bremermann je izveo prve pokuse na računalu „optimizacija kroz evoluciju i križanje“. Od tada pa do devedesetih godina razvijale su se različite implementacije osnovne ideje, koje su kasnije ujedinjene u područje evolucijskog računarstva, a to su:

- evolucijsko programiranje, Fogel, Owens i Walsh
- genetski algoritam, Holland
- strategije evolucije, Rechenberg i Schwefel
- genetsko programiranje, Koza

Od tada do danas održane su brojne međunarodne konferencije, pokrenuta je Europska istraživačka mreža EvoNet, pokrenuti su časopisi te su izdane tisuće članaka i radova, sve na temu evolucijskog računarstva [2].

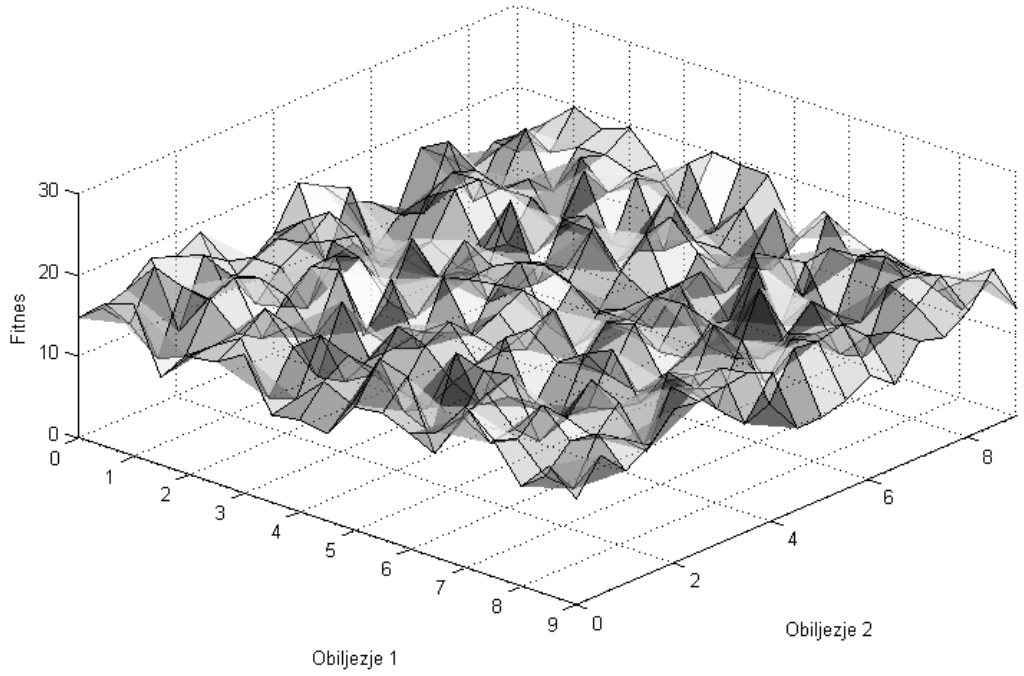
3.2. Inspiracije iz biologije

3.2.1. Teorija evolucije

Darwinova teorija evolucije nudi objašnjenje biološke raznolikosti i osnovnih mehanizama evolucije. Glavnu ulogu u procesu evolucije ima prirodna selekcija. Svaka okolina ima ograničenu količinu resursa što znači da može opskrbljivati samo određen broj članova populacije. Kako je osnovni instinkt živih bića usmjeren prema reprodukciji, preživjet će jedino oni članovi populacije koji se najuspješnije bore za dostupne resurse. Ovaj fenomen nazivamo preživljavanje najboljih i predstavlja jedan od dva osnovna principa evolucije. Drugi predstavlja fenotipska obilježja unutar populacije. Fenotipska obilježja su fizičke razlike i razlike u ponašanju članova populacije koje utječu na odziv prema okolini. Ova obilježja su

jedinstvena svakom članu populacije te ih ocjenjuje okolina. U slučaju pozitivne ocjene rezultiraju povećanom šansom za reprodukciju te se prenose na sljedeću generaciju. Također se prilikom reprodukcije mogu dogoditi male varijacije odnosno mutacije u fenotipskim obilježjima potomaka. Kroz te varijacije dolazi do novih kombinacija obilježja od kojih se najbolja dalje reproduciraju i preživljavaju. Kako prolazi proces evolucije dolazi do potpune promjene u strukturi populacije.

Ovaj proces se može opisati metaforom adaptivnog krajolika ili adaptivne površine. Adaptivni krajolik može prikazati populaciju s n obilježja u n+1 dimenzionalnom prostoru pri čemu je visina mjera dobrote, odnosno fitnesa. Svaka jedinka predstavlja jednu kombinaciju obilježja, odnosno točku na krajoliku. Evolucija je ustvari proces postepenog pomaka populacije prema područjima višeg fitnesa ovisno o varijacijama i prirodnoj selekciji [2].



Slika 13. Adaptivni krajolik s dva obilježja

Ovisno o izgledu adaptivnog krajolika probleme dijelimo na unimodalne i multimodalne. Multimodalni su oni kod kojih imamo velik broj lokalnih optimuma od kojih je jedan ujedno i globalni, dok su unimodalni oni kod kojih postoji samo jedan optimum.

U procesu evolucije može doći do pojave genetskog drifta. Ovaj fenomen predstavlja situaciju u kojoj članovi s visokim fitnesom nestanu iz populacije ili dođe do gubitka određenih obilježja. Pri tome se može dogoditi da cijela populacija napusti lokalne optimume, ali može se dogoditi i pronalaženje boljeg lokalnog ili čak globalnog optimuma.

3.2.2. Prirodna genetika

Proces evolucije se može opisati na mikroskopskoj razini zahvaljujući molekularnoj genetici. Sve potrebne informacije za izgradnju nekog organizma nalaze se u DNA tog organizma [2]. Fenotipska svojstva svakog organizma određeni su njegovim genotipskim svojstvima. Geni utječu na fenotipska svojstva složenim preslikavanjem:

- Jedan gen može utjecati na više obilježja (pleiotropija)
- Više gena može utjecati na jedno obilježje (poligenija)

Fenotipske varijacije (npr. visina, boja kose, boja očiju) su uvijek uzrokovane genotipskim varijacijama koje su opet posljedica mutacija gena i križanja gena prilikom reprodukcije. Sav genetski materijal neke jedinke nazivamo genom, genom se nalazi u dijelovima DNA koji se nazivaju kromosomi. Ljudske stanice sadrže 23 para kromosoma, 23 kromosoma od oca i 23 kromosoma od majke, koji zajedno definiraju naša fizička svojstva. Prilikom mejoze stanica dolazi do pojave križanja kromosoma. Par kromosoma se duplicira te se jedan od nova dva para križa, time imamo par koji je kopija kromosoma od oca i majke i jedan novi par kromosoma s potpuno novim svojstvima. Povremeno se određeni dijelovi genetskog koda malo promjene, te dolazi do novih dijelova gena koji nisu naslijeđeni od roditelja. Ovaj proces nazivamo mutacija, a njegova posljedica može biti:

- negativna – potomak nije sposoban za život
- neutralna – nova obilježja nemaju utjecaj
- pozitivna – dolazi do novih vrlo dobrih obilježja

Evolucijski algoritmi se bave proučavanjem računalnih sustava koji sliče pojednostavljenim verzijama procesa i mehanizama evolucije, kako bi se postigli ishodi tih procesa i mehanizama. Mehanizmi evolucijskih algoritama oponašaju kako se evolucija odvija kroz proces izmjene i širenja genetskog materijala (proteina), općenito se koriste zamjene za mehanizme evolucije kao što su genetska rekombinacija (križanje) i mutacija. Algoritmi koji iskorištavaju svojstva iz povezanih područja evolucijske biologije, razvojne biologije, populacijske genetike i populacijske ekologije, također pripadaju evolucijskom računarstvu [3].

3.3. Osnovni elementi evolucijskih algoritama

Kako bi algoritam mogao izvršavati svoju funkciju on uz proces inicijalizacije i uvjet prekida mora imati određene komponente i procese:

1) Prikazivanje jedinki: definira način na koji će članovi populacije (fenotip), biti prikazani u algoritmu. Na primjer, ako neki cijeli broj u algoritmu prikazujemo njegovom binarnom vrijednošću onda je ona njegov genotip. Prikazivanje ustvari predstavlja kodiranje, prebacivanje fenotipa u genotip, i dekodiranje, prebacivanje genotipa u fenotip.
2) Funkcija procjene kvalitete: predstavlja zahtjeve okoline na koje se populacija mora prilagoditi. Ova funkcija svakom fenotipu dodjeljuje jedinstvenu vrijednost (fitnes) koja čini osnovu selekcije. Fenotip je bolji što je njegov fitnes optimalniji.
3) Populacija: skup genotipa koji predstavljaju prikaze mogućih rješenja. Populacija je konstantne veličine, a njezina raznolikost prikazuje broj različitih fitnesa i genotipa prisutnih u populaciji.
4) Mehanizam odabira roditelja: svakom članu populacije dodjeljuje različitu vrijednost u odnosu na iznos njihovog fitnesa. Kvalitetniji pojedinci imaju veću vjerojatnost postanka roditeljima, ali čak i najlošiji član može biti izabran.
5) Operatori varijacije: imaju ulogu stvoriti nove članove populacije. Rekombinacija ili križanje uzima značajke roditelja kako bi se stvorili potomci u nadi za boljim svojstvima. Mutacija stvara nove značajke koje nisu postojale kod roditelja.
6) Mehanizam selekcije: određuje koje će jedinke biti uključene u novoj generaciji populacije. To može biti bazirano na fitnesu gdje se zadržavaju najbolji pojedinci roditelja i potomaka, na starosti gdje se zadržavaju samo potomci ili kombinacija.

Inicijalizacija algoritma se uglavnom izvodi stohastički, pri čemu mora postojati ravnomjeran raspored i miješanje gena u populaciji. Ponekad se prilikom kreiranja populacije koristi iskustvo pa se odabiru poznate vrijednosti koje daju dobra rješenja [2].

Uvjet prekida se izvršava kada:

- Dosegnemo željeni fitnes.
- Prijeđemo maksimalno dopušteni broj generacija.
- Raznovrsnost populacije padne ispod određenog minimuma.
- Određeni broj generacija završi bez poboljšanja fitnesa

POČETAK

INICIJALIZIRAJ populaciju slučajnih kandidata;

OCIJENI svakog kandidata;

PONAVLJAJ DO (*UVJET PREKIDA ZADOVOLJEN*)

1 *IZABERI* roditelje;

2 *REKOMBINIRAJ* parove roditelja;

3 *MUTIRAJ* potomke;

4 *OCIJENI* nove kandidate;

5 *IZABERI* jedinke za novu generaciju;

prekid

KRAJ

Slika 14. Pseudokod tipičnog EA [2]

3.4. Genetski algoritam

Genetic Algorithm, GA.

Genetski algoritam je evolucijski algoritam za globalno optimiranje i pripada području evolucijskog računarstva. Genetski algoritam je najpoznatiji primjer evolucijskih algoritama, a početno je zamišljen kao sredstvo za proučavanje adaptivnog ponašanja. Sličan je drugim evolucijskim algoritmima kao što su: genetsko programiranje, strategije evolucije i evolucijsko programiranje te je osnova velikom broju varijanti algoritama i njihovih potpodručja. Genetski algoritam je inspiriran populacijskom genetikom (uključujući nasljedstvo i učestalost gena), evolucijom na razini populacije kao i strukturom (kromosomi, geni, aleli) i mehanizmima (križanje i mutacija) evolucije. Time predstavljanju modernu sintezu evolucijske biologije [3].

3.4.1. Prikazivanje jedinki kod GA

Genetski algoritam koristi sve osnovne elemente evolucijskih algoritama za koje postoji velik broj mogućih izvedbi. Zbog velikog broja kombinacija izvedbi osnovnih elemenata, možemo odabrati one koje će odgovarati željenoj primjeni [2][4][5].

Prikazivanje jedinki, kao što je već opisano, definira prikaz genotipa populacije. Kod odabira načina prikazivanja važno je odabrati onaj koji će biti najprikladniji problemu koji se rješava. Odabir pravog načina prikaza je jedan od najtežih koraka prilikom konstruiranja genetskog algoritma.

Najčešći načini prikazivanja jedinki su:

- Prikazivanje binarnim vrijednostima.
- Prikazivanje cijelim brojevima.
- Prikazivanje realnim brojevima.

Prikazivanje binarnim vrijednostima je jedan od najjednostavnijih i najranijih načina prikaza. Genotip je prikazan nizom binarnih znamenki za koji je potrebno odlučiti koliko će biti dugačak i kako će se dekodirati da bi se dobio fenotip. Ovaj pristup je često neučinkovit.

Prikazivanje cijelim vrijednostima je puno prirodnije od binarnog kod problema gdje su rješenja također cijeli brojevi jer genotip direktno prikazuje fenotip i nije ga potrebno dekodirati.

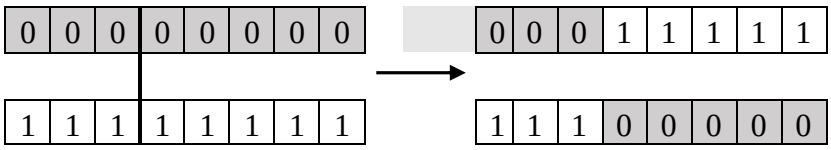
Prikazivanje realnim vrijednostima je često najbolji način prikazivanja kandidata rješenja. U ovom slučaju genotip za rješenje s *n* gena postaje vektor $\langle x_1, \dots, x_n \rangle$ gdje su $x_i \in \mathbb{R}$.

3.4.2. Rekombinacija kod GA

Rekombinacija ili križanje je proces kojim se stvara novo rješenje iz informacija koja su sadržavala stara rješenja [2][4][5]. Križanje je inspirirano prirodnim procesima i jedna od najvažnijih značajki genetskih algoritama. Kod križanja se uvodi vjerojatnost križanja p_c, čija je vrijednost najčešće u intervalu [0,5 ,1], koja se uspoređuje s nasumičnom vrijednosti u intervalu $[0,1\rangle$. Ako je nasumična vrijednost manja od vjerojatnosti križanja onda dolazi do križanja. Ako je veća stara rješenja se jednostavno kopiraju.

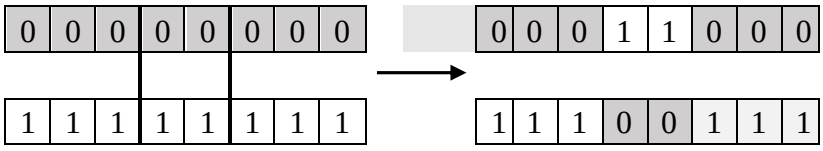
Kod križanja binarnih i cjelobrojnih vrijednosti postoje tri standardna načina križanja kod kojih se od dva roditelja dobivaju dva potomka:

1. Križanje u jednoj točki. Odabere se nasumični broj iz intervala [1, *l*-1] (gdje je *l* dužina gena), genomi oba roditelja se razdvoje u toj točki te im se zamijene repovi.

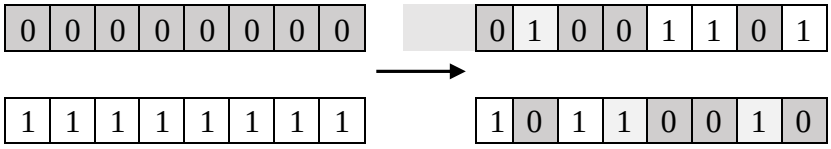


Slika 15. Križanje u jednoj točki

2. Križanje u više točaka. Kao i kod prethodnog primjera, mjesta križanja se odaberu nasumično samo što ih ima više.
3. Ujednačeno križanje. U prethodna dva primjera roditelji su se podijelili na određeni broj dijelova te su se geni presložili kako bi se dobili potomci. Kod ujednačenog križanja svaki se gen gleda zasebno te se nasumično odabire od kojeg će se roditelja naslijediti.



Slika 16. Križanje u više točaka, primjer s dvije točke

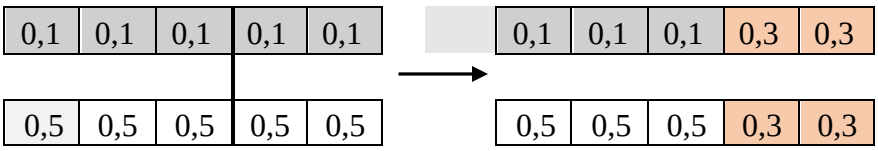


Slika 17. Ujednačeno križanje

Kod križanja realnih vrijednosti koristi se aritmetičko križanje. Križanje se može provesti samo na jednom genu, na dijelu genoma ili na cijelom genomu. Kada se odaberu geni koji će se križati, vrijednosti gena potomka se računaju prema formuli:

$$z_i = \alpha x_i + (1 - \alpha) y_i \tag{3.1}$$

Pri čemu je *i* mjesto gena koji se križa. Za drugi potomak se zamijene vrijednosti x_i i y_i, a oni geni koji se ne križaju se jednostavno prepišu.

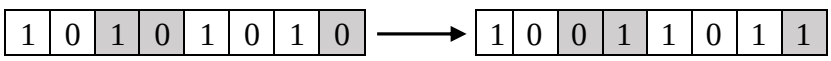


Slika 18. Aritmetičko križanje: α=0,5

3.4.3. Mutacija kod GA

Mutacija je opći naziv za one operatore varijacije koji koriste samo jednog roditelja kako se dobio jedan potomak primjenjujući neku nasumičnu promjenu u genotipu [2][4][5]. Mutacija je također inspirirana prirodom i značajna je zato što omogućava uvođenje svojstava koja nisu postojala kod roditelja u genotip, time se može povećati raznolikost genotipa u populaciji. Kod mutacije se uvodi vjerojatnost mutacije p_m. Način mutacije se odabire ovisno o načinu prikazivanja jedinki.

Kod mutacije binarnih vrijednosti najčešća metoda svaki gen uzima zasebno i ovisno o nasumičnoj vrijednosti zamjenjuje gen (0 u 1 ili 1 u 0). Ako je nasumična vrijednost manja od vjerojatnosti mutacije onda dolazi do mutacije. Ako je veća ne dolazi do promjene gena.



Slika 19. Mutacija kod binarnih vrijednosti

Kod mutacije cjelobrojnih vrijednosti mogu se koristiti dva načina mutacije. Prvi je identičan mutaciji kod binarnih vrijednosti, ovisno o vjerojatnosti mutacije p_m odaberu se nasumične znamenke koje se zamjenjuju drugom nasumičnom vrijednosti. Drugi način odabranim znamenkama dodaje malu (pozitivnu ili negativnu) vrijednost. Ovaj način mutacije češće uvodi manje promjene od prethodnog, a potrebno je kontrolirati veći broj parametara.

Kod mutiranja realnih vrijednosti, vrijednosti gena se nasumično mijenjaju unutar intervala ograničenog donjom L_i i gornjom U_i vrijednosti. Tako za genotip s n gena dolazi do transformacije: $\langle x_1, \dots, x_n \rangle \rightarrow \langle x'_1, \dots, x'_n \rangle$, gdje su $x_i, x'_i \in [L_i, U_i]$. Ovisno o vjerojatnosti distribucije vrijednosti iz koje se odabire nova vrijednost gena, razlikuju se ujednačena mutacija i neujednačena mutacija. Kod ujednačene mutacije vrijednosti x'_i se odabiru nasumično iz intervala $[L_i, U_i]$, dok se kod neujednačene mutacije odabiru nasumično uz normalnu raspodjelu unutar istog intervala.

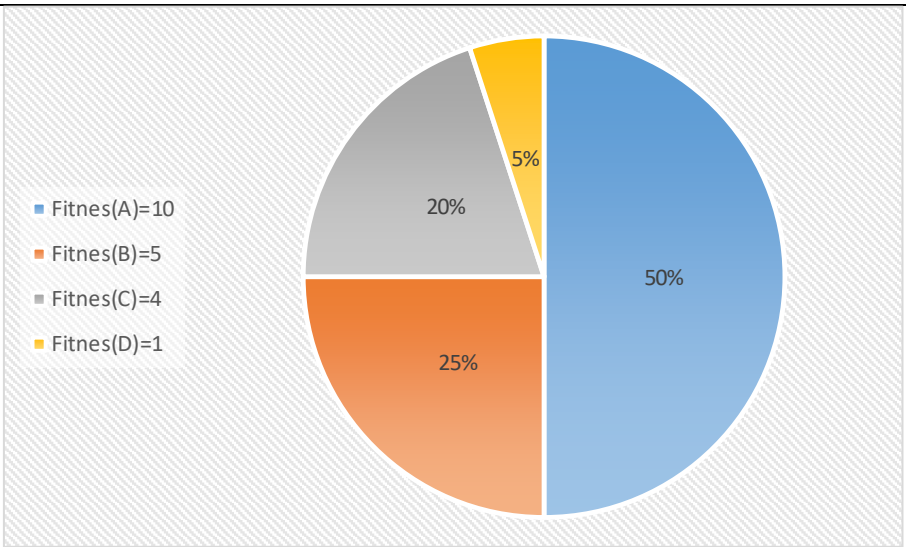
3.4.4. Odabir roditelja kod GA

Prilikom odabira članova populacije koji će postati roditelji želja nam je uzeti one članove koji imaju najbolja svojstva (najbolji fitnes) kako bi i potomci naslijedili dobra svojstva [2][4][5]. Ipak ovaj pristup ne daje uvijek najbolje rezultate zato što može dovesti do preuranjene konvergencije, odnosno zaustavljanja u određenom lokalnom optimumu. Zato se uvode mehanizmi koji probabilističkim metodama odabiru roditelje iz populacije. Primjenjuje se ruletno pravilo kod kojega svaki član ima vjerojatnost biti odabran u odnosu na kvalitetu svog fitnesa. Kvalitetniji članovi imaju veću vjerojatnost, ali i najlošiji član može biti odabran. Vjerojatnost da će u populaciji n članova član f_i biti odabran iznosi:

$$\frac{f_i}{\sum_{j=1}^{n} f_j} \tag{3.2}$$

Što znači da je vjerojatnost odabira nekog člana jednaka omjeru njegovog fitnesa i sume fitnesa cijele populacije. U slučaju da između vrijednosti fitnesa ne postoje velike razlike, svaki član ima gotovo jednaku vjerojatnost odabira. Ovaj slučaj se može primijetiti u kasnijim generacijama algoritma kada već postoji konvergencija prema rješenju i daljnje poboljšavanje fitnesa napreduje jako sporo.

Prije ruletnog pravila moguće je i rangirati članove ovisno o pojedinačnim vrijednostima fitnesa i izbaciti određeni postotak populacije iz skupa kandidata. Preostali članovi se nakon toga koriste u ruletnom pravilu s vjerojatnosti izbora ovisnoj o mjestu koje su dobili prilikom rangiranja, a ne ovisno o vrijednosti njihovog fitnesa.



Slika 20. Ruletno pravilo s četiri člana

3.4.5. Selekcija kod GA

Nakon što su odabrani roditelji mehanizmima križanja i mutacije dobili potomke, pitanje je koji će članovi biti dio nove populacije, a koji će biti izbačeni [2][4][5]. Ovisno o kriterijima izbora članova nove populacije postoje dvije glavne podjele:

1) Selekcija ovisno o starosti: ove metode se koriste kod jednostavnijih GA. Budući da je broj roditelja i potomaka jednak, svaka jedinka postoji samo jednu generaciju. Svi roditelji se odbacuju i zamjenjuju potomcima.
2) Selekcija ovisno o fitnesu: ove metode uzimaju vrijednosti fitnesa roditelja kao i potomaka prilikom izvršavanja selekcije, ali mnogo metoda uz to uključuju i starost kao kriterije odabira. Najznačajniji ovakvi mehanizmi su zamjena najgorih članova i elitizam.

Kod zamjene najgorih članova, određeni broj najgorih roditelja se zamjenjuje s najboljim potomcima ovisno o vrijednostima njihovih fitnesa. Ova metoda može dovesti do preuranjene konvergencije rješenja u lokalni optimum zato što se populacija fokusira oko najboljeg trenutnog člana.

Elitizam se koristi u kombinaciji sa selekcijom ovisno o starosti i određenim stohastičkim selekcijama kako bi se spriječio gubitak članova s najboljim fitnesom. Najbolji član unutar svake generacije se proglasi elitnim članom i zadržava se unutar populacije dok se jedan nasumični potomak izbacuje.

4. ALGORITMI ROJA

4.1. Inteligencija roja

Inteligencija roja je područje proučavanja računalnih sustava inspirirano 'kolektivnom inteligencijom'. Kolektivna inteligencija proizlazi iz suradnje velikog broja homogenih posrednika u okolini. Primjeri uključuju jata riba i ptica te kolonije mrava. Ovakav tip inteligencije je decentraliziran, samoorganiziran te raspršen u okolini. Pojedinci u populaciji su u konstantnoj komunikaciji s drugim članovima, uče kroz vlastita iskustva i iskustva cijelog roja dok se postepeno približavaju cilju. U prirodi se ovakvi sustavi uobičajeno koriste kako bi se riješili problemi efikasne potrage za hranom, izbjegavanje grabežljivaca ili preseljenje kolonije. Informacija je uobičajeno pohranjena kroz sudjelovanje homogenih posrednika ili je pohranjena ili prenesena kroz samu okolinu, npr. plesom pčela, udaljenošću kod riba i ptica ili korištenjem feromona kod mrava [3][6].

Područje se sastoji od dva dominantna potpodručja:

1) Optimiziranje rojem čestica (Particle Swarm Optimization, PSO), koristi se za razne primjene, a istražuje probabilističke algoritme inspirirane skupljanjem životinja u rojeve, jata ili stada.
2) Optimiziranje kolonijom mrava (Ant Colony Optimization, ACO), koje istražuje probabilističke algoritme inspirirane ponašanjem mrava kroz njihovu koordinaciju i traganjem za hranom. ACO je posebno učinkovit za aplikacije koje uključuju usmjeravanje, odnosno određivanje optimalnog puta.

Kao i kod evolucijskog računarstva, algoritmi i strategije inteligencije roja se smatraju adaptivnim strategijama i uobičajeno se koriste kod problema potrage i optimizacije. Dok računalni sustavi postaju sve složeniji i teško je imati potpunu kontrolu, sustavi koji koriste inteligenciju roja su robusni, jako efikasni, relativno jednostavni te se mogu brzo implementirati za razne optimizacijske probleme.

Postoje mnogi različiti algoritmi i klase algoritama koji pripadaju području inteligencije roja, samo neki od njih su:

- roj čestica
- algoritmi mrava
- algoritmi pčela
- algoritmi bakterija

4.2. Optimizacija rojem čestica

Particle Swarm Optimization, PSO.

Optimizacija rojem čestica pripada području inteligencije roja i kolektivne inteligencije te je potpodručje računalne inteligencije. Razvili su je James Kennedy i Russel Eberhart 1995. i brzo je postala jedan on najpopularnijih metaheurističkih algoritama inspiriranih prirodom. Optimizacija rojem čestica je povezana s drugim algoritmima inteligencije roja kao što je optimizacija kolonijom mrava i osnova je mnogim algoritmima.

Optimizacija rojem čestica inspirirana je socijalnim ponašanjem nekih životinja kao što su ponašanje jata ptica ili jata riba dok tragaju za hranom. Čestice roja lete kroz okolinu prateći sposobnije članove roja i općenito ispravljaju svoje kretanje prema povijesno dobrim područjima u njihovoj okolini.

Cilj algoritma je da sve čestice pronađu optimum u višedimenzijskom prostoru. To se postiže dodjeljivanjem početnih nasumičnih pozicija i dodjeljivanjem malih početnih nasumičnih brzina svim česticama u prostoru. Algoritam se izvršava kao simulacija, unaprjeđivanjem pozicije svake čestice ovisno o njezinoj brzini, najboljoj prethodnoj poziciji čestice i najboljoj poznatoj globalnoj poziciji u zadanom prostoru. To se odvija probabilistički, korištenjem nasumično odabranih vrijednosti. Funkcija cilja je testirana nakon svake promjene pozicije. Nakon nekog vremena, kroz kombinaciju istraživanja i iskorištavanja dobrih poznatih pozicija u zadanom prostoru, čestice se grupiraju i zajedno konvergiraju oko optimuma ili nekoliko optimuma[3][6].

4.2.1. Princip rada algoritma roja

Na početku je potrebno inicijalizirati populaciju čestica. Svakoj čestici se pridijeli nasumična vrijednost za svaki parametar koji se optimizira. Tako se dobije vektor s *k* članova, gdje je *k* broj optimizacijskih parametara:

$$\mathbf{x}_i = [x_1, x_2, \dots, x_k] \tag{4.1}$$

Gdje je *i* broj čestice u populaciji. Ovaj vektor u stvari predstavlja poziciju čestice u prostoru. Budući da je to početna pozicija čestice, ona je ujedno i najbolja pozicija koji je čestica imala do tog trenutka tako da se stvara novi vektor koji će sadržati najbolje osobne pozicije svake čestice za svaki optimizacijski parametar:

$$\mathbf{p}_i^{\text{best}} = \left[p_1^{\text{best}}, p_2^{\text{best}}, \dots, p_k^{\text{best}}\right] \tag{4.2}$$

Uz najbolju osobnu vrijednost svake čestice definira se i najbolja globalna vrijednost koja je postignuta unutar cijele populacije za svaki optimizacijski parametar:

$$\mathbf{g}^{\text{best}} = \left[g_1^{\text{best}}, g_2^{\text{best}}, \dots, g_k^{\text{best}}\right] \tag{4.3}$$

Uz brzinu se svakoj čestici dodijeli i mala nasumična početna vrijednost brzine za svaki optimizacijski parametar:

$$\mathbf{v}_i = [v_1, v_2, \dots, v_k] \tag{4.4}$$

Roj se u početku rada čini kao neorganizirana populacija čestica koje se unutar prostora naizgled gibaju u nasumičnim smjerovima. Čestice se ustvari kreću u zadanom prostoru pod utjecajem vlastite najbolje prošle pozicije i najbolje prošle pozicije unutar cijelog roja. Brzina *i*-te čestice se ažurira u svakoj iteraciji *t* koristeći sljedeću jednadžbu prema [7]:

$$\begin{aligned}\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + &\left(c_1 \cdot rand \cdot \left(\mathbf{p}_i^{\text{best}} - \mathbf{x}_i(t)\right)\right) + \\ &\left(c_2 \cdot rand \cdot \left(\mathbf{g}^{\text{best}} - \mathbf{x}_i(t)\right)\right)\end{aligned} \tag{4.5}$$

Gdje je $\mathbf{v}_i(t+1)$ nova brzina *i*-te čestice, a funkcija *rand* generira nasumične vrijednosti $\in \langle 0{,}1\rangle$. Prva zagrada je kognitivna komponenta brzine, ona utječe na novu brzinu ovisno o prethodnim iskustvima te čestice čija se brzina mijenja. c_1 je kognitivni faktor ubrzanja koji određuje koliki utjecaj u novoj brzini imaju vlastita iskustva te čestice. Druga zagrada je socijalna komponenta brzine, ona utječe na novu brzinu ovisno o prethodnim iskustvima cijelog roja uzimajuću najbolju poziciju koju je roj pronašao. c_2 je socijalni faktor ubrzanja koji određuje koliki utjecaj u novoj brzini imaju iskustva cijelog roja. Ovi faktori su ograničeni tako da je $c_1 + c_2 \leq 4$, ali uglavnom se uzima da vrijednost oba faktora iznosi 2.

Također se uvodi i kontrolni parametar inercije *w* koji dodatno kontrolira utjecaj prethodne brzine na trenutnu. U početnom stadiju algoritma, velika inercija omogućava bolje globalno pretraživanje, gdje u kasnijim stadijima kada se već postiže konvergencija omogućava bolje lokalno pretraživanje. Uglavnom se kao početna vrijednost inercije uzima 0,9, a za krajnju se uzima 0,4. Inercija se računa prema sljedećoj formuli [8]:

$$w = w_{\max} - \frac{(w_{\max} - w_{\min})}{t_{\max}} t \tag{4.6}$$

Nakon uvođenja inercije umjesto formule (4.5) za izračun brzine koristi se formula (4.7) [8]:

$$\begin{aligned}\mathbf{v}_i(t+1) = w\ \mathbf{v}_i(t) + &\left(c_1 \cdot rand \cdot \left(\mathbf{p}_i^{\text{best}} - \mathbf{x}_i(t)\right)\right) + \\ &\left(c_2 \cdot rand \cdot \left(\mathbf{g}^{\text{best}} - \mathbf{x}_i(t)\right)\right)\end{aligned} \tag{4.7}$$

Pozicija čestice se ažurira koristeći sljedeću jednadžbu:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \times \mathrm{d}t \quad \textbf{(4.8)}$$

Gdje je d*t* vremenska konstanta.

Često se nakon ažuriranja nove pozicije čestice, pogotovo u početnim stadijima kada je brzina velika, dogodi da čestica izađe iz prostora koji je određen optimizacijskom funkcijom. U tom slučaju česticama koje su izašle iz prostora pretrage daju se nove nasumične vrijednosti.

Za novo dobivene pozicije čestica se testira vrijednost funkcije, ažuriraju se najbolje osobne i najbolja globalna pozicija te se ovaj proces ponavlja dok se ne pronađe optimalno rješenje ili se ne premaši predviđeni broj iteracija.

5. USPOREDBA RADA GA I PSO

U ovom poglavlju uspoređen je rad algoritama na prethodno opisanim testnim funkcijama. Određene funkcije su malo izmijenjene u odnosu na prethodni prikaz, to se odnosi na DropWave, Easom, HolderTable i EggHolder funkcije. Budući da kod ovih funkcija globalni optimum ne iznosi nula, zbog jednostavnijeg prikaza, funkcijama je dodana konstanta čime su one samo pomaknute tako da im se optimum nalazi u nuli.

Za svaku testnu funkciju algoritmi su testirani 100 puta te je iz dobivenih podataka prikazano najbolje rješenje i izračunata je standardna devijacija svih rješenja:

$$s = \left(\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2 \right)^{\frac{1}{2}} \tag{5.1}$$

Gdje je :

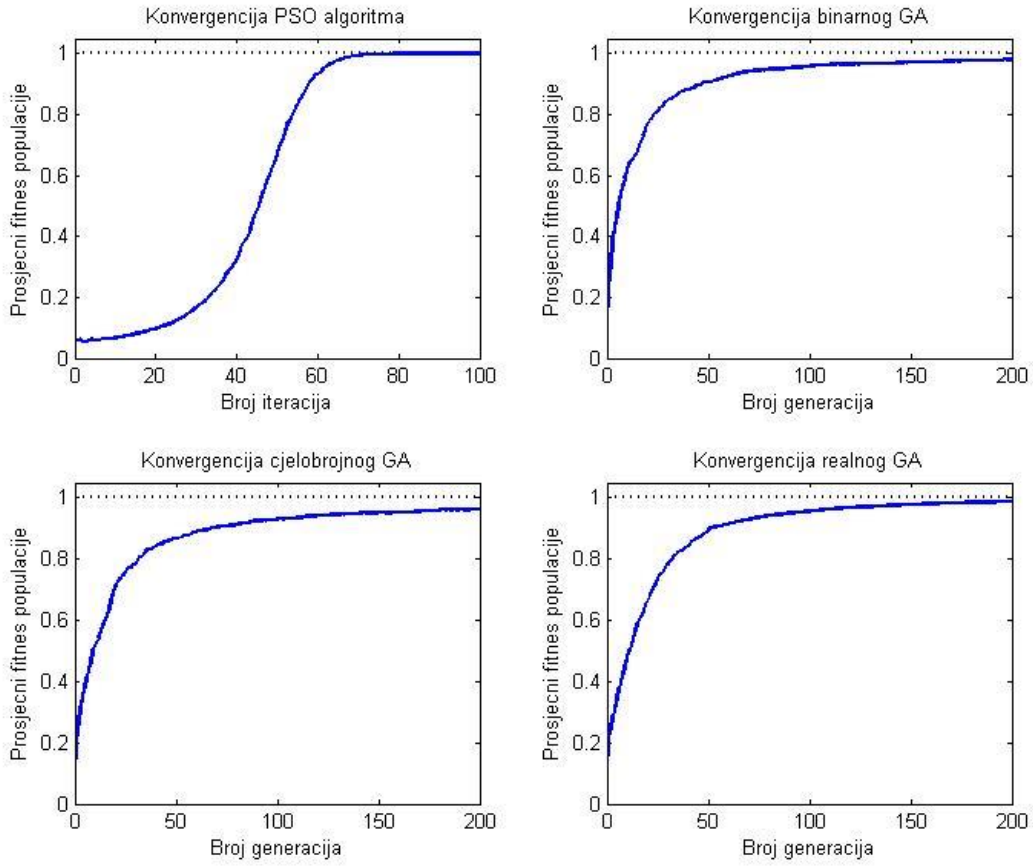
$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{5.2}$$

Prilikom testiranja funkcija kod PSO algoritma je uzeta veličina populacije od 20 članova, dok je kod genetskih algoritama uzeta populacija od 40 članova, za GA sa binarnim i cjelobrojnim vrijednostima duljina gena iznosi 20.

Tablica 2. Usporedba algoritama za Ackley funkciju

Algoritam	Najbolji $f(\mathbf{x})$	Standardna devijacija	Broj iteracija
PSO	$4{,}7269 \times 10^{-11}$	$4{,}0116 \times 10^{-9}$	100
GA-binarne vrijednosti	$5{,}0415 \times 10^{-4}$	$1{,}5839 \times 10^{-2}$	200
GA-cjelobrojne vrijednosti	$4{,}3413 \times 10^{-4}$	$4{,}3972 \times 10^{-2}$	200
GA-realne vrijednosti	$2{,}9022 \times 10^{-4}$	$1{,}1918 \times 10^{-2}$	200

Iz dobivenih rezultata kod Ackley funkcije vidi se kako je PSO algoritam pronašao globalni optimum te postigao jako dobro rješenje. Sve verzije GA su također pronašle globalni optimum, ali s manjom točnošću i u više iteracija od PSO algoritma. Može se primijetiti kako sve verzije GA imaju približno jednaku točnost te da im je krivulja konvergencije rješenja slična.

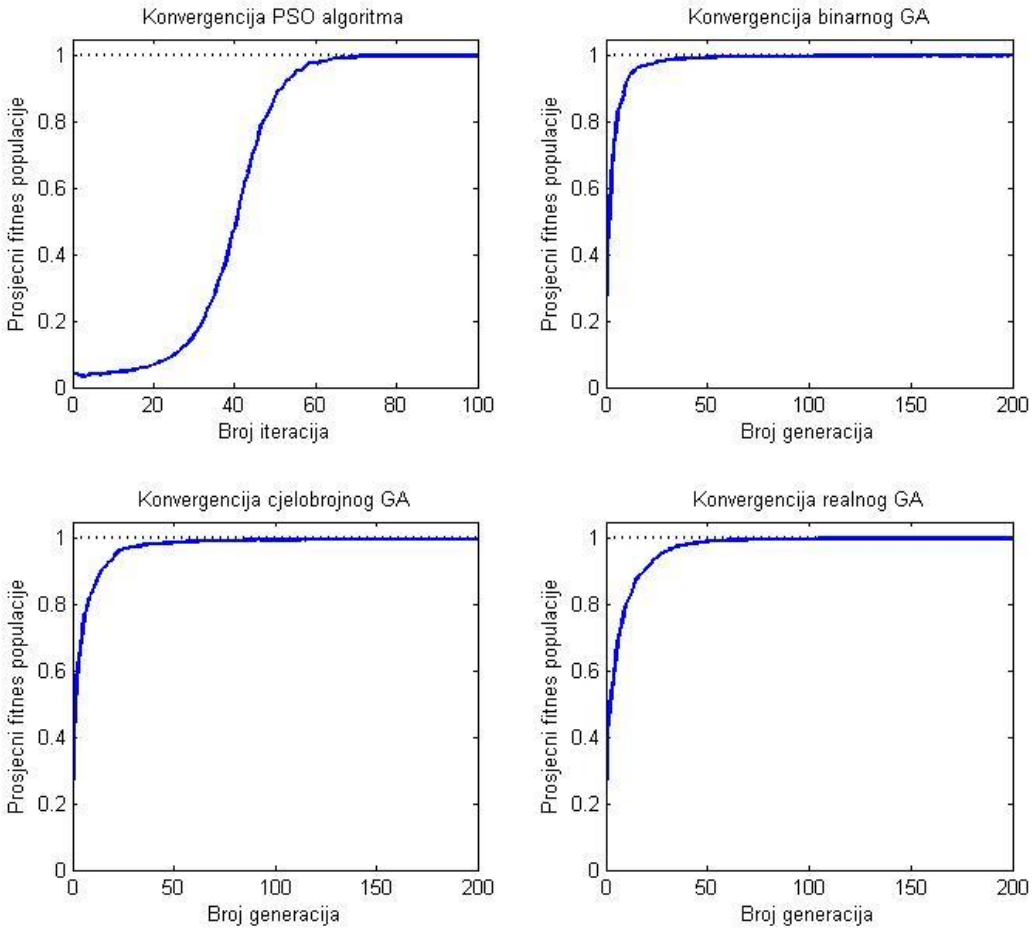


Slika 21. Konvergencije rješenja algoritama za Ackley funkciju

Tablica 3. Usporedba algoritama za Rastrigin funkciju

Algoritam	Najbolji $f(\mathbf{x})$	Standardna devijacija	Broj iteracija
PSO	0	0	100
GA-binarne vrijednosti	$2{,}4269 \times 10^{-6}$	$6{,}6579 \times 10^{-4}$	200
GA-cjelobrojne vrijednosti	$6{,}7299 \times 10^{-8}$	$6{,}3864 \times 10^{-3}$	200
GA-realne vrijednosti	$3{,}6844 \times 10^{-9}$	$4{,}2971 \times 10^{-4}$	200

Prilikom testiranja algoritama Rastrigin testnom funkcijom, PSO algoritam postigao je točno rješenje u svih 100 testiranja. Ovaj rezultat vjerojatno je prouzročen greškom u Matlab programskom paketu jer za ovu funkciju Matlab ne može prikazati vrijednost funkcije manju od 2×10^{-16}, ali to govori da je dobivena točnost barem tolika. GA je također postigao jako dobre rezultate.

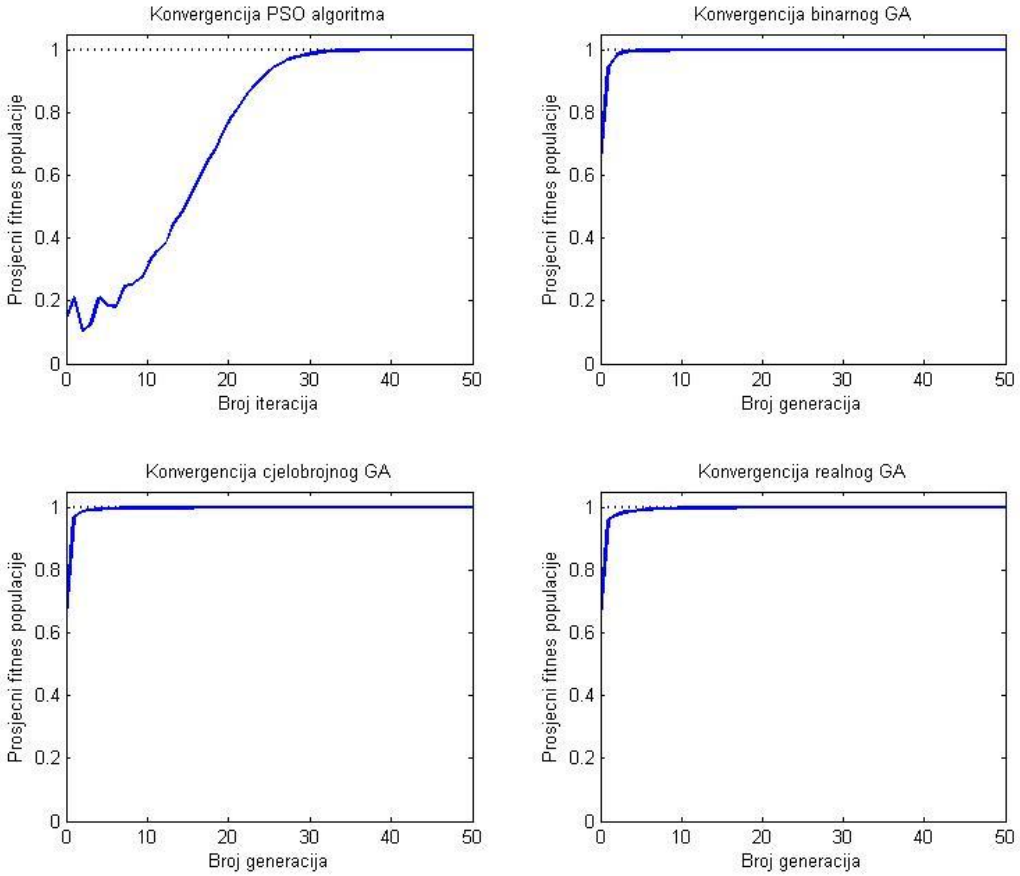


Slika 22. Konvergencije rješenja algoritama za Rastrigin funkciju

Tablica 4. Usporedba algoritama za Sfernu funkciju

Algoritam	Najbolji $f(\mathbf{x})$	Standardna devijacija	Broj iteracija
PSO	$1{,}5109 \times 10^{-14}$	$1{,}2014 \times 10^{-11}$	50
GA-binarne vrijednosti	$7{,}8989 \times 10^{-8}$	$5{,}8987 \times 10^{-5}$	50
GA-cjelobrojne vrijednosti	$6{,}3313 \times 10^{-8}$	$1{,}5073 \times 10^{-4}$	50
GA-realne vrijednosti	$3{,}1893 \times 10^{-10}$	$1{,}5133 \times 10^{-4}$	50

Budući da je Sferna funkcija najjednostavnija od iskorištenih funkcija, točnost postignutih rješenja je očekivana. Iz prikaza konvergencije rješenja može se vidjeti kako GA puno brže pronađu optimalno rješenje od PSO algoritma, ali imaju više poteškoća prilikom 'peglanja' postignutog rješenja budući da su na kraju opet imali manju točnost od PSO algoritma.

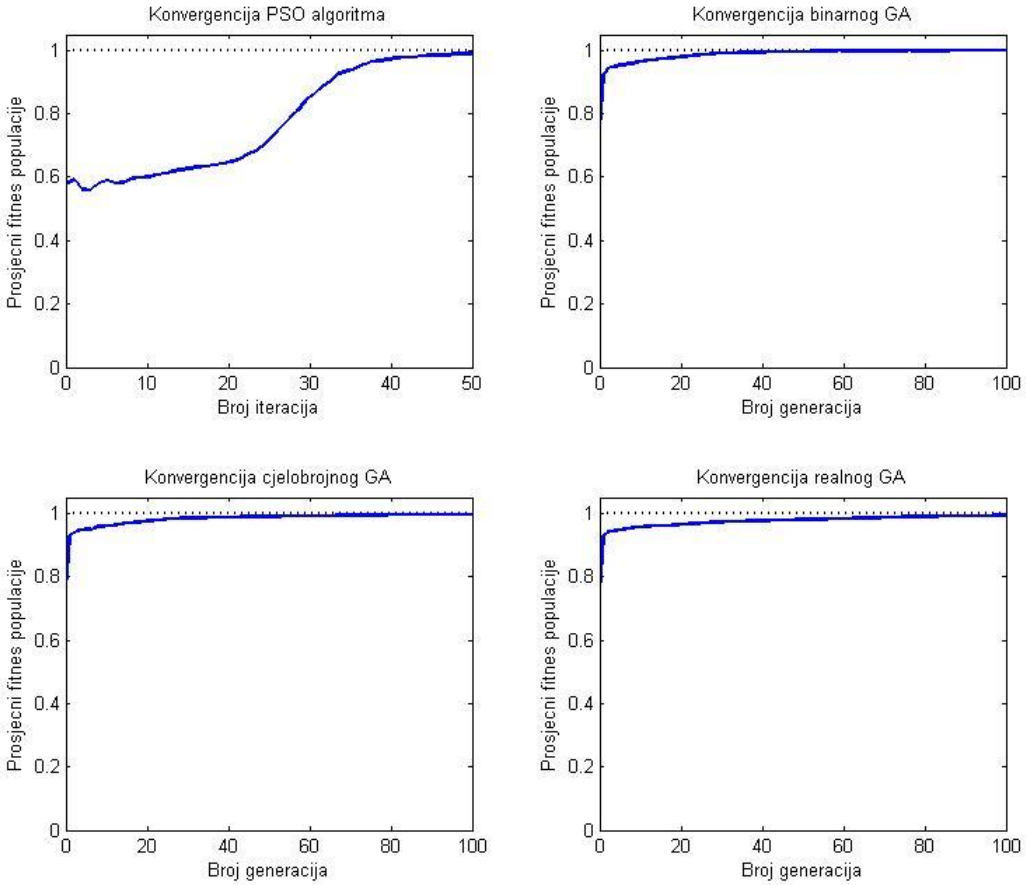


Slika 23. Konvergencije rješenja algoritama za Sfernu funkciju

Tablica 5. Usporedba algoritama za DropWave funkciju

Algoritam	Najbolji $f(\mathbf{x})$	Standardna devijacija	Broj iteracija
PSO	$2{,}6512 \times 10^{-13}$	$6{,}3435 \times 10^{-3}$	50
GA-binarne vrijednosti	$1{,}4919 \times 10^{-6}$	$1{,}2770 \times 10^{-3}$	100
GA-cjelobrojne vrijednosti	$1{,}1205 \times 10^{-5}$	$1{,}2361 \times 10^{-2}$	100
GA-realne vrijednosti	$1{,}1132 \times 10^{-6}$	$1{,}9062 \times 10^{-2}$	100

I kod DropWave funkcije svi algoritmi su postigli veliku točnost rješenja, s tim da je PSO algoritam ponovno postigao točnost koja je značajno bolja u usporedbi s GA.

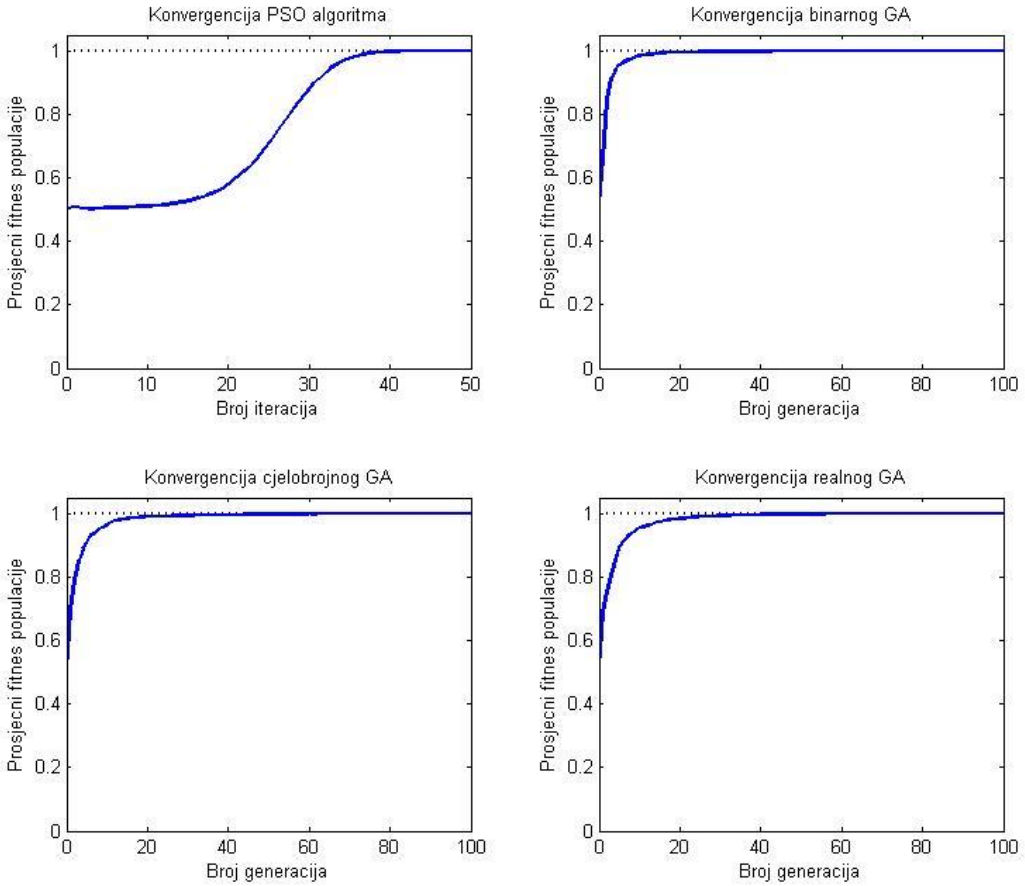


Slika 24. Konvergencije rješenja algoritama za DropWave funkciju

Tablica 6. Usporedba algoritama za Easom funkciju

Algoritam	Najbolji $f(\mathbf{x})$	Standardna devijacija	Broj iteracija
PSO	$4{,}9905 \times 10^{-13}$	$3{,}1401 \times 10^{-10}$	50
GA-binarne vrijednosti	$3{,}3097 \times 10^{-7}$	$1{,}5779 \times 10^{-4}$	100
GA-cjelobrojne vrijednosti	$1{,}3565 \times 10^{-7}$	$1{,}0625 \times 10^{-3}$	100
GA-realne vrijednosti	$1{,}6328 \times 10^{-8}$	$5{,}0827 \times 10^{-4}$	100

Kao i kod Sferne funkcije, kod Easom funkcije se može primijetiti da su GA puno ranije pronašli optimalno rješenje od PSO algoritma, ali kao što je to bio slučaj i ranije PSO algoritam je uspio postići veću točnost na kraju testiranja.

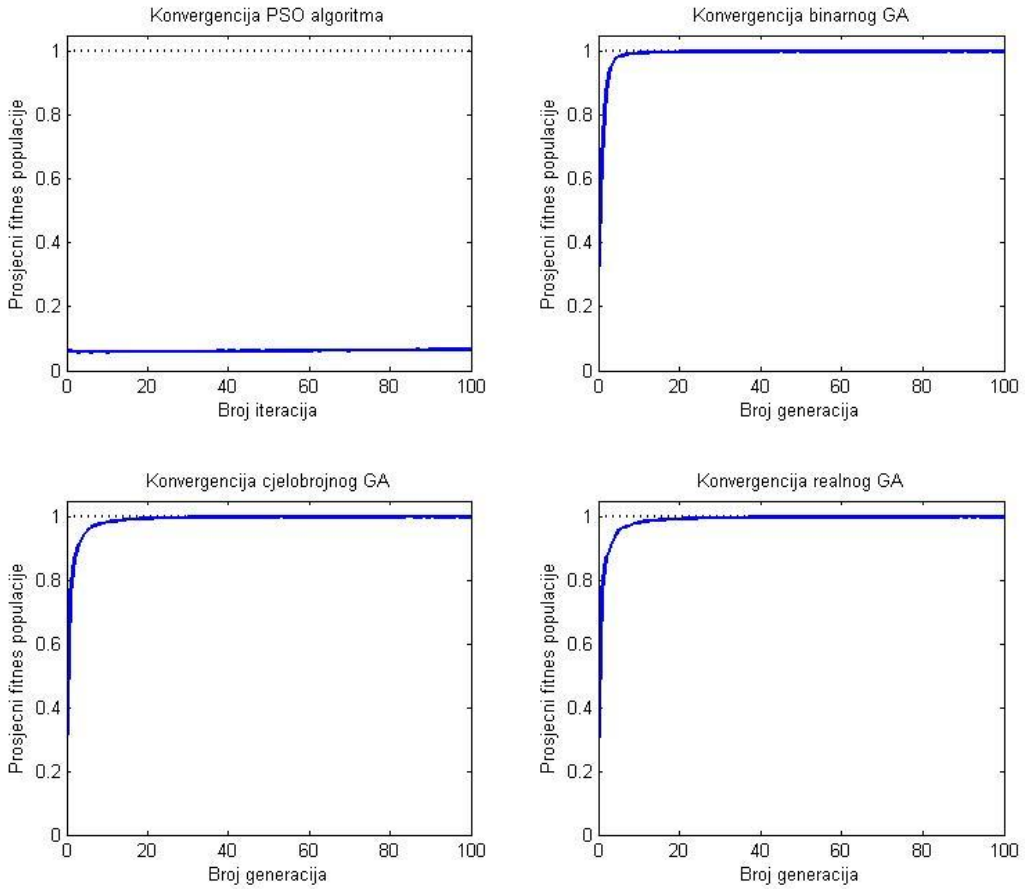


Slika 25. Konvergencije rješenja algoritama za Easom funkciju

Tablica 7. Usporedba algoritama za HolderTable funkciju

Algoritam	Najbolji $f(\mathbf{x})$	Standardna devijacija	Broj iteracija
PSO	Neuspješno	Neuspješno	100
GA-binarne vrijednosti	$2{,}4517 \times 10^{-6}$	$9{,}5703 \times 10^{-5}$	100
GA-cjelobrojne vrijednosti	$2{,}5148 \times 10^{-6}$	$2{,}8911 \times 10^{-4}$	100
GA-realne vrijednosti	$1{,}7750 \times 10^{-6}$	$5{,}2326 \times 10^{-4}$	100

Kod HolderTable funkcije GA su ponovno postigli veliku točnost rješenja. Za razliku od prethodnih testnih funkcija gdje je imao izvrsne rezultate PSO algoritam nije uspio pronaći optimum. Razlog tome je činjenica da se svi optimumi funkcije nalaze blizu granica u kojima je funkcija definirana zbog čega se dogodi da čestice izađu iz domene problema te ih algoritam vrati na neko nasumično mjesto unutar prostora pretrage. Osim toga funkcija ima četiri globalna optimuma što je dodatno otežalo potragu česticama.

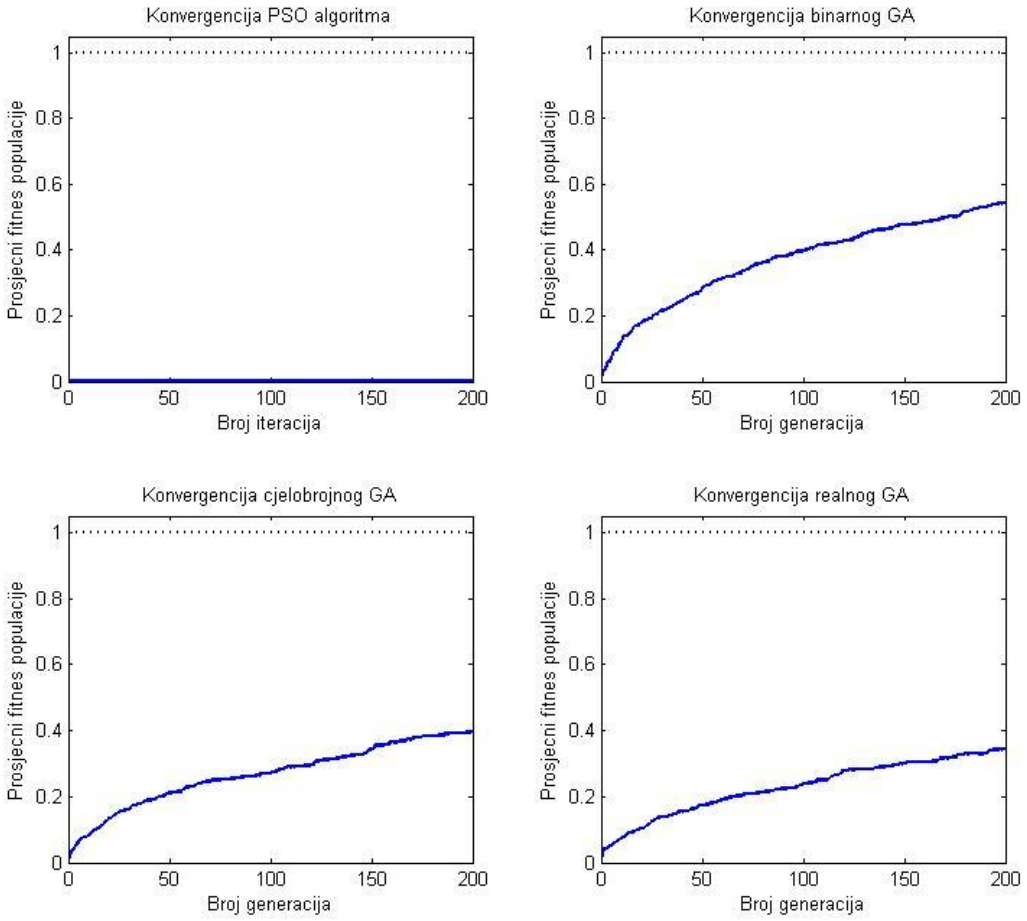


Slika 26. Konvergencije rješenja algoritama za HolderTable funkciju

Tablica 8. Usporedba algoritama za EggHolder funkciju

Algoritam	Najbolji $f(\mathbf{x})$	Standardna devijacija	Broj iteracija
PSO	Neuspješno	Neuspješno	200
GA-binarne vrijednosti	Neuspješno	Neuspješno	200
GA-cjelobrojne vrijednosti	Neuspješno	Neuspješno	200
GA-realne vrijednosti	Neuspješno	Neuspješno	200

Rješenje EggHolder funkcije nije uspio pronaći niti jedan korišteni algoritam. Iz krivulja konvergencije vidi se da bi GA možda pronašli rješenje nakon jako velikog broja iteracija dok PSO algoritam nikada niti ne bi otkrio točno rješenje. Razlog tomu je velika složenost funkcije i činjenica da se optimalno rješenje ove funkcije nalazi na samom rubu domene problema.

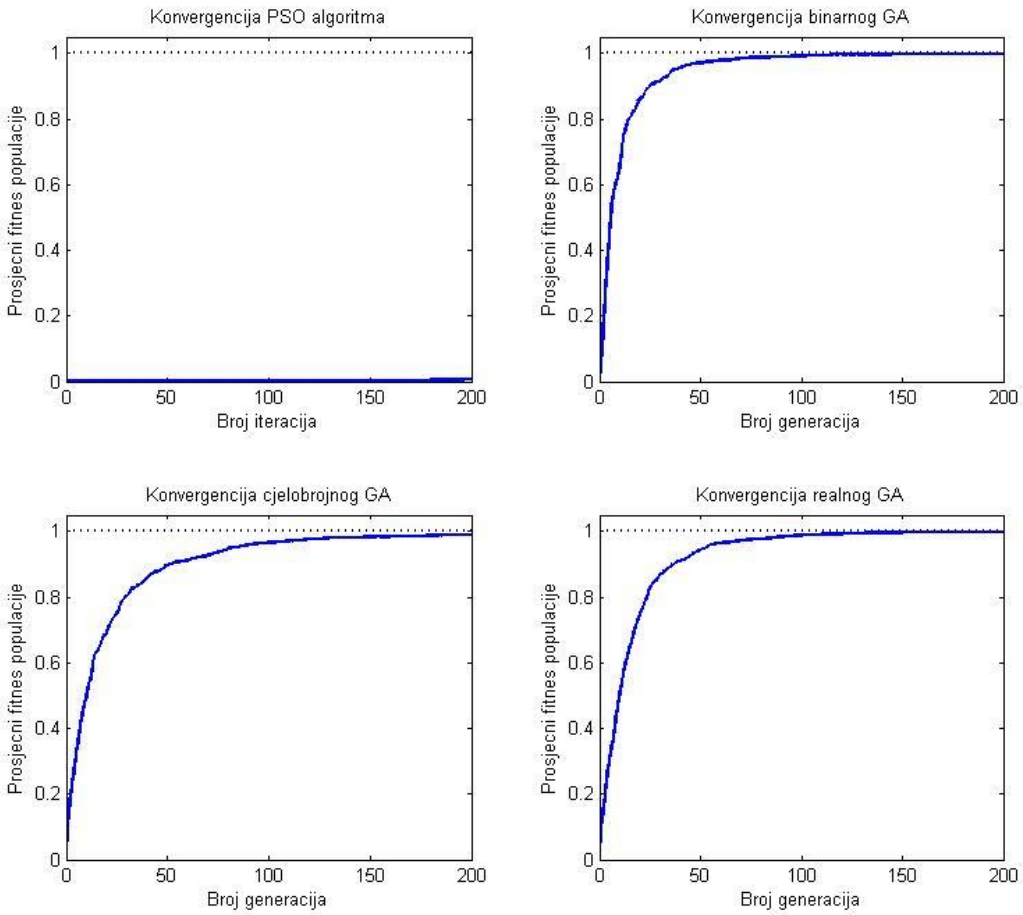


Slika 27. Konvergencije rješenja algoritama za EggHolder funkciju

Tablica 9. Usporedba algoritama za Schwefel26 funkciju

Algoritam	Najbolji $f(\mathbf{x})$	Standardna devijacija	Broj iteracija
PSO	Neuspješno	Neuspješno	200
GA-binarne vrijednosti	$3{,}0222 \times 10^{-5}$	$2{,}2329 \times 10^{-3}$	200
GA-cjelobrojne vrijednosti	$2{,}5507 \times 10^{-5}$	$2{,}0980 \times 10^{-3}$	200
GA-realne vrijednosti	$2{,}5841 \times 10^{-5}$	$2{,}3197 \times 10^{-3}$	200

Kao ni kod prethodne dvije funkcije, kod Schwefel26 funkcije PSO algoritam nije uspio pronaći rješenje iz istih razloga. Treba naglasiti kako je mali broj članova populacije uspio pronaći optimalno rješenje, ali cilj algoritma je da ga pronađu svi. GA je ipak uspio pronaći rješenje zato što se ono ne nalazi na samom rubu domene problema iako je Schwefel26 funkcija gotovo jednako složena kao i EggHolder funkcija.



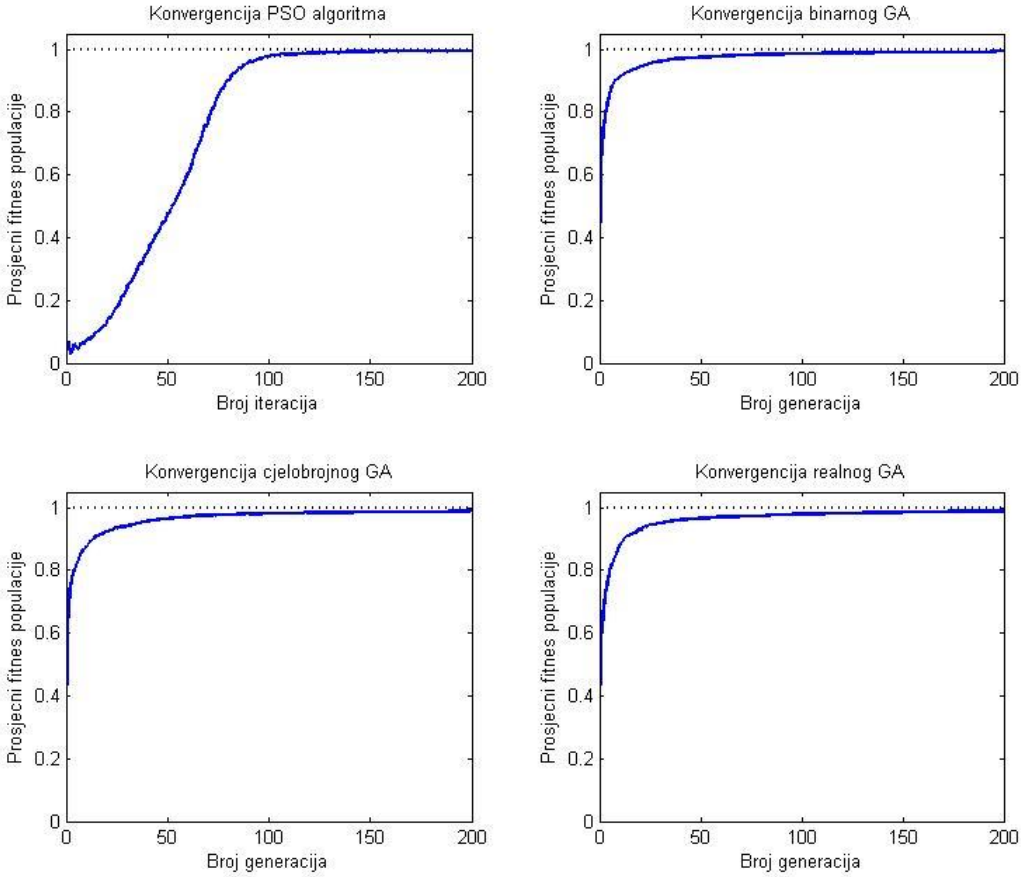
Slika 28. Konvergencije rješenja algoritama za Schwefel26 funkciju

Tablica 10. Usporedba algoritama za Griewank funkciju

Algoritam	Najbolji $f(\mathbf{x})$	Standardna devijacija	Broj iteracija
PSO	0	$4{,}4222 \times 10^{-3}$	200
GA-binarne vrijednosti	$3{,}3674 \times 10^{-5}$	$5{,}5623 \times 10^{-3}$	200
GA-cjelobrojne vrijednosti	$5{,}5336 \times 10^{-4}$	$8{,}9683 \times 10^{-3}$	200
GA-realne vrijednosti	$5{,}6888 \times 10^{-6}$	$8{,}6598 \times 10^{-3}$	200

Može se primijetiti kako je za Griewank testnu funkciju PSO algoritam postigao točno najbolje rješenje. Unatoč tome treba naglasiti kako je to postigao samo u polovici testova dok je u drugoj polovici zapeo u nekom od lokalnih minimuma koji se nalaze u neposrednoj blizini globalnog minimuma. GA su pronašli rješenje s manjom točnošću nego što su je imali u prethodnim

testovima. Ipak to se može pripisati samoj funkciji. Prema [10] ovu funkciju je uspjelo optimizirati samo 6,08% dostupnih optimizacijskih algoritama.



Slika 29. Konvergencije rješenja algoritama za Griewank funkciju

6. ZAKLJUČAK

U ovom radu implementirana su dva heuristička populacijska algoritma, genetski algoritam koji pripada skupini evolucijskih algoritama i algoritam za optimizaciju rojem čestica koji pripada skupini algoritama inteligencije roja. Algoritmi su implementirani i testirani na devet različitih testnih funkcija i to u Matlab okružju. Sve testne funkcije predstavljale su probleme minimizacije gdje je bilo potrebno pronaći njihov globalni minimum između velikog broja lokalnih minimuma.

Iz rezultata dobivenih testiranjem algoritama na testnim funkcijama može se zaključiti da PSO i GA općenito daju jako dobre rezultate. PSO je postigao visoku točnost kod funkcija kojima je uspio pronaći optimalno rješenje, no nije ga uspio pronaći kod HolderTable, EggHolder i Schwefel26 funkcija. Razlog tome je činjenica da se optimumi ovih funkcija nalaze blizu granica u kojima su funkcije definirane. Kod PSO algoritma čestice koje putuju kroz prostor i konvergiraju prema rješenju mogu slučajno izaći izvan granica prostora u kojemu je funkcija definirana. Takve čestice se automatski vraćaju unutar granica, ali im se daje nasumična pozicija u prostoru čime čestica koja je bila na putanji prema rješenju gubi svoj početni napredak u konvergenciji. Iz tog razloga može se zaključiti da PSO algoritam nije prikladan za probleme čija su rješenja na rubovima domene te se PSO algoritam treba izbjegavati kod ovakvih problema ili im se treba proširiti područje definicije, ako je to moguće.

GA nije postigao razinu točnosti kao PSO algoritam, ali i dalje je ostvario jako dobre rezultate kod većine testnih funkcija te je za većinu testnih funkcija imao bržu početnu konvergenciju od PSO algoritma. Jedina funkcija kod koje nije pronađeno optimalno rješenje je EggHolder funkcija, ali i to se može pripisati položaju optimuma koji je nalazi na samoj granici domene problema. Kako su to stohastički algoritmi, izvršavanje svih procesa u algoritmu ovisi o nasumičnim vrijednostima, a kod ovakve funkcije, koja uz takav položaj globalnog optimuma ima nepravilno raspoređene lokalne optimume koji ne mogu pomoći približavanju globalnom, teško je postići konvergenciju rješenju. Međusobno uspoređujući verzije GA vidljivo je kako su podjednako dobri za rješavanje ovakvih optimizacijskih problema, budući da su postignute točnosti kao i dobivene standardne devijacije otprilike istog reda veličine. Ovaj rezultat može se pripisati odabranim početnim parametrima za GA s binarnim i cjelobrojnim vrijednostima budući da je odabrana velika duljina gena koja je znatno utjecala na njihovu točnost. Iz tog razloga prednost se može dati GA s realnim vrijednostima .

LITERATURA

[1] Soliman, S. A. H., Mantawy, A. A. H.: Modern Optimization Techniques with Applications in Electric Power Systems, Springer, 2012.

[2] Eiben, A. E., Smith, J. E.: Introduction to Evolutionary Computing, Springer, 2003.

[3] Brownlee, J.: Clever Algorithms: Nature-Inspired Programming Recipes, Creative Commons, Australia, 2011.

[4] Goldberg, D. E.: Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Publishing Company, Inc, USA, 1989.

[5] Parkinson, A.R., Balling, R., Hedengren J.D., Optimization Methods for Engineering Design, Brigham Young University, 2013.

[6] Kennedy, J., Eberhart, R. C.: Swarm Intelligence, Morgan Kaufmann Publishers, San Francisco, CA, USA, 2001.

[7] Kennedy, J., Eberhart, R. C.: Particle swarm optimization, in: IEEE International Conference on Neural Networks, Vol. 4, 1995, pp 1942-1948.

[8] Xinjie Yu, Mitsuo Gen: Introduction to Evolutionary Algorithms, Springer, 2010.

[9] Momin, J., Xin-She Yang, A literature survey of benchmark functions for global optimization problems, Int. Journal of Mathematical Modelling and Numerical Optimisation, Vol. 4, No. 2, pp. 150–194, 2013.

[10] http://infinity77.net/global_optimization/index.html (12.2.2017)

[11] www.sfu.ca/~ssurjano/optimization.html (12.2.2017)

[12] https://www.mathworks.com/help/matlab/ (12.2.2017)

PRILOZI

I. CD-R disc

II. Maltlab kod algoritama

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Genetski algoritam s binarnim vrijednostima
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all
clc

funkcija=[0];
lf=1;
while funkcija<lf
   %  odabir funkcije
   choice=menu('Odaberite optimizacijsku funkciju','Ackley','Rastrigin','Sferna','Drop wave',...
      'Easom','Holder table','Eggholder','Schwefel26','Griewank','Kraj rada');
   funkcija=[1,2,3,4,5,6,7,8,9,10];
   lf=length(funkcija);
   funkcija=funkcija(choice);

   if funkcija==10
      break
   end

   % Pocetni parametri
   prompt={'Upisite zeljeni broj gena(parno):','Upisite zeljenu duzinu gena','Upisite zeljeni
broj iteracija:'};% Tekst za zeljenu varijablu
   answer=inputdlg(prompt);
   B = str2num(answer{1});     % Broj gena - mora biti paran
   A = str2num(answer{2});     % Duzina gena
   ite = str2num(answer{3});   % Zeljeni broj iteracija

   pc=0.15; % vjerojatnost za krizanje
   pm=0.4; % vjerojatnost za mutaciju

   if mod (B,2)~=0
      disp('Greska, prekid programa')
      break
   end

   SUM=0;
   [LB,UB,zmin,zmax]=poc_UvjFunkc(funkcija);    % Pocetni uvjeti ovisno o funkciji
   diskr=(UB-LB)/100;

   % Funkcija
   xx=[LB:diskr:UB];
   yy=[LB:diskr:UB];
   zz=[0];
   lx=length(xx);
   ly=length(yy);

   for i=1:lx;
```

```
    xi=xx(i);
    for j=1:ly;
      yi=yy(j);
      zi=odabir_OptFunkc(funkcija,xi,yi);
      zz(j,i)=zi;
    end
  end

  % Inicijalizacija
  x1=[0];
  y1=[0];
  fitnes=[0];
  fitmax=[0];
  novo=zeros(B,2*A);
  rod=zeros(B,2*A);
  elitx=[0];
  elity=[0];
  pop=[];
  pop_xy=[];
  pop_fun=[];
  proba=[];
  dim=2;  % Broj dimenzija

  popx=randi([0 1],B,A);
  popy=randi([0 1],B,A);
  pop=[popx popy];

  for broj=1:ite
    pop_intx=bi2de(popx,'left-msb') % pretvara binarni zapis u dekadski
    pop_inty=bi2de(popy,'left-msb')

    % Odredivanje fitnesa
    % Cijeli brojevi se pretvaraju u odgovarajuce koordinate x i y osi

    pop_xy(:,1)=LB+pop_intx*(2*UB/(2^A-1));
    pop_xy(:,2)=LB+pop_inty*(2*UB/(2^A-1))

    for i=1:B
      pop_fun(i,1)=odabir_OptFunkc(funkcija,pop_xy(i,1),pop_xy(i,2));
    end
    [fitnes(broj,:),minMjesto]=min(pop_fun)

    % ELITIZAM
    for i=1:B
      for j=1:B
        proba(i,j)=odabir_OptFunkc(funkcija,pop_xy(i,1),pop_xy(j,2));
      end
    end

    [fmin0,index0]=min(proba(:));
```

```
[red,stupac]=ind2sub(size(proba),index0)    % y- red, x- stupac

elitx=popx(red,:);
elity=popy(stupac,:);

xmin=pop_xy(minMjesto,1);
ymin=pop_xy(minMjesto,2);
zmin=fitnes(broj,:);        % za graf

% Vjerojatnost izbora pojedinog gena:
a=1./pop_fun;
vjerojatnost=a/sum(a);

% Generiranje parova za reprodukciju

for i=1:B
   r=rand(1,1);
   for j=1:B
      SUM=SUM+vjerojatnost(j);
      if SUM>r
         rod(i,:)=pop(j,:);
         SUM=0;
         break
      end
   end
end

% Uzimanje  parova za reprodukciju
for u=1:2:B
   t=u+1;
   PAR_trenutni=rod(u:t,:);  % uzimaju  se parovi gena

   if rand<=pc
      qq=1+(2*A-1)*rand(1,1);
      QQ=round(qq);

      p1=PAR_trenutni(1,:);
      p2=PAR_trenutni(2,:);

      % Krizanje
      P11X=[p1(1:QQ) p2(QQ+1:2*A)]; % izmedu  parova se krizaju dijelovi koda
      P22X=[p2(1:QQ) p1(QQ+1:2*A)];
      PAR_trenutni_krizani=[P11X;P22X];
   else
      PAR_trenutni_krizani=PAR_trenutni;  % ako nema krizanja, kopiraju se roditelji
   end
   novo(u:t,:)=[PAR_trenutni_krizani];  % nova populacija
end

% Mutacija
```

```
    for i=1:B
      for j=1:2*A
        pmut=rand(1,1);
        if pmut<=pm && novo(i,j)==1 % ako dolazi do mutacije znamenka na i,j mjestu se
mijenja
          novo(i,j)=0;
        elseif pmut<=pm && novo(i,j)==0
          novo(i,j)=1;
        end
      end
    end
    elit=randi([1,B]);

    popx=novo(:,1:A);
    popx(elit,:)=elitx;
    popy=novo(:,A+1:2*A);
    popy(elit,:)=elity;

    fitmax(broj,:)=zmax/(zmax*0.999999+fitnes(broj,:));

    clf
    figure(1)
    subplot(2,2,[1,3])
    surf(xx,yy,zz,'EdgeColor','none','LineStyle','none'), hold on
    scatter3(pop_xy(:,1),pop_xy(:,2),pop_fun,'MarkerEdgeColor','k','MarkerFaceColor','g')
    scatter3(xmin,ymin,zmin,'MarkerEdgeColor','k','MarkerFaceColor','r'), hold off
    xlabel('x')
    ylabel('y')
    zlabel('f(x,y)')
    title('Prikaz rada algoritma')

    subplot(2,2,2)
    x1=linspace(0,broj,broj);
    y1=fitmax;
    semilogx(x1,y1,x1,1.05);
    xlabel('Broj generacija')
    ylabel('Fitnes')
    title('Konvergencija GA')
    axis([0 broj 0 1.05]);

    subplot(2,2,4)
    x1=0;
    y1=0;
    plot(x1,y1), axis off
    title('Informacije')
    str=char('Broj Generacija:',num2str(broj));
    text(0.2,0.8,0.9,str,'FontSize',15)
    str=char('Fitnes:',num2str(fitmax(broj,:)));
    text(0.2,0.25,0.9,str,'FontSize',14)
    str=char('Najbolja x koordinata:', num2str(xmin));
```

```
        text(-0.9,0.8,0.9,str,'FontSize',14)
        str=char('Najbolja y koordinata:', num2str(ymin));
        text(-0.9,0.25,0.9,str,'FontSize',14)
        str=char('Najbolji f(x,y)=', num2str(min(zmin)));
        text(-0.9,-0.3,0.9,str,'FontSize',14)
        switch funkcija
            case 1 % Ackley
                str=char('fmin(x,y)=0  za x=0, y=0');
            case 2 % Rastrigin
                str=char('fmin(x,y)=0  za x=0, y=0');
            case 3 % Sferna
                str=char('fmin(x,y)=0  za x=0, y=0');
            case 4 % Drop wave
                str=char('fmin(x,y)=0  za x=0, y=0');
            case 5 % Easom
                str=char('fmin(x,y)=0  za x=pi, y=pi');
            case 6 % Holder table
                str=char('fmin(x,y)=0  za x=+/-8.05502, y=+/-9.66469');
            case 7 % Eggholder
                str=char('fmin(x,y)=0  za x=512, y=404.2319');
            case 8 % Schwefel26
                str=char('fmin(x,y)=0  za x=420.9687, y=420.9687');
            case 9 % Griewank
                str=char('fmin(x,y)=0  za x=0, y=0');
        end
        text(-0.9,-0.80,0.9,str,'FontSize',14)
        pause(.05);

        if min(fitnes)<0.00001
            break
        end
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Kraj
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Genetski algoritam s cjelobrojnim vrijednostima
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all
clc

funkcija=[0];
lf=1;
while funkcija<lf

    % odabir funkcije
    choice=menu('Odaberite optimizacijsku funkciju','Ackley','Rastrigin','Sferna',...
        'Drop wave','Easom','Holder table','Eggholder','Schwefel26','Griewank','Kraj rada');
    funkcija=[1,2,3,4,5,6,7,8,9,10];
    lf=length(funkcija);
    funkcija=funkcija(choice);

    if funkcija==10
        break
    end

    % Pocetni parametri
    prompt={'Upisite zeljeni broj gena(parno):','Upisite zeljenu duzinu gena','Upisite zeljeni broj iteracija:'};% Tekst za zeljenu varijablu
    answer=inputdlg(prompt);
    B = str2num(answer{1});     % Broj gena - mora biti paran
    A = str2num(answer{2});     % Duzina gena
    ite = str2num(answer{3});   % Zeljeni broj iteracija

    pc=0.15; % vjerojatnost za križanje
    pm=0.4; % vjerojatnost za mutaciju

    if mod (B,2)~=0
        disp('Greska, prekid programa')
        break
    end

    SUM=0;
    [LB,UB,zmin,zmax]=poc_UvjFunkc(funkcija);   % Pocetni uvjeti ovisno o funkciji
    diskr=(UB-LB)/100;

    % Funkcija
    xx=[LB:diskr:UB];
    yy=[LB:diskr:UB];
    zz=[0];
    lx=length(xx);
    ly=length(yy);
```

```
for i=1:lx;
   xi=xx(i);
   for j=1:ly;
      yi=yy(j);
      zi=odabir_OptFunkc(funkcija,xi,yi);
      zz(j,i)=zi;
   end
end

% Inicijalizacija

x1=[0];
y1=[0];
fitnes=[0];
fitmax=[0];
novo=zeros(B,2*A);
rod=zeros(B,2*A);
elitx=[0];
elity=[0];
pop=[];
pop_xy=[];
pop_fun=[];
proba=[];
dim=2;

popx=randi([0 9],B,A);
popy=randi([0 9],B,A);
pop=[popx popy];

for broj=1:ite
   pop_intx=bi2de(popx,10,'left-msb'); % pretvara binarni zapis u dekadski
   pop_inty=bi2de(popy,10,'left-msb');

   % Odredivanje fitnesa
   % Cijeli brojevi se pretvaraju u odgovarajuce koordinate x i y osi

   pop_xy(:,1)=LB+pop_intx*(2*UB/(10^A-1));
   pop_xy(:,2)=LB+pop_inty*(2*UB/(10^A-1));

   for i=1:B
      pop_fun(i,1)=odabir_OptFunkc(funkcija,pop_xy(i,1),pop_xy(i,2));
   end
   [fitnes(broj,:),minMjesto]=min(pop_fun);

   % ELITIZAM
   for i=1:B
      for j=1:B
         proba(i,j)=odabir_OptFunkc(funkcija,pop_xy(i,1),pop_xy(j,2));
      end
   end
```

```
[fmin0,index0]=min(proba(:));
[red,stupac]=ind2sub(size(proba),index0)    % y- red, x- stupac

elitx=popx(red,:);
elity=popy(stupac,:);

xmin=pop_xy(minMjesto,1);
ymin=pop_xy(minMjesto,2);
zmin=fitnes(broj,:);        % za graf

% Vjerojatnost izbora pojedinog gena:
a=1./pop_fun;
vjerojatnost=a/sum(a);

% Generiranje parova za reprodukciju

for i=1:B
   r=rand(1,1);
   for j=1:B
      SUM=SUM+vjerojatnost(j);
      if SUM>r
         rod(i,:)=pop(j,:);
         SUM=0;
         break
      end
   end
end

% Uzimanje parova za reprodukciju
for u=1:2:B
   t=u+1;
   PAR_trenutni=rod(u:t,:); % uzimaju se parovi gena

   if rand<=pc
      qq=1+(2*A-1)*rand(1,1);
      QQ=round(qq);

      p1=PAR_trenutni(1,:);
      p2=PAR_trenutni(2,:);

      % Krizanje
      P11X=[p1(1:QQ) p2(QQ+1:2*A)]; % izmedu parova se krizaju dijelovi koda
      P22X=[p2(1:QQ) p1(QQ+1:2*A)];
      PAR_trenutni_krizani=[P11X;P22X];
   else
      PAR_trenutni_krizani=PAR_trenutni; % ako nema krizanja, kopiraju se roditelji
   end
   novo(u:t,:)=[PAR_trenutni_krizani]; % nova populacija
end
```

```
% Mutacija
for i=1:B
   for j=1:2*A
      pmut=rand(1,1);
      if pmut<=pm              % ako dolazi do mutacije znamenka  na i,j mjestu se mijenja
         novo(i,j)=randi([0 9]);
      end
   end
end
elit=randi([1,B]);

popx=novo(:,1:A);
popx(elit,:)=elitx;
popy=novo(:,A+1:2*A);
popy(elit,:)=elity;

fitmax(broj,:)=zmax/(zmax*0.99999+fitnes(broj,:));

clf
figure(1)
subplot(2,2,[1,3])
surf(xx,yy,zz,'EdgeColor','none','LineStyle','none'),  hold on
scatter3(pop_xy(:,1),pop_xy(:,2),pop_fun,'MarkerEdgeColor','k','MarkerFaceColor','g')
scatter3(xmin,ymin,zmin,'MarkerEdgeColor','k','MarkerFaceColor','r'), hold off
xlabel('x')
ylabel('y')
zlabel('f(x,y)')
title('Prikaz rada algoritma')

subplot(2,2,2)
x1=linspace(0,broj,broj);
y1=fitmax;
semilogx(x1,y1,x1,1.05);
xlabel('Broj generacija')
ylabel('Fitnes')
title('Konvergencija GA')
axis([0 broj 0 1.05]);

subplot(2,2,4)
x1=0;
y1=0;
plot(x1,y1), axis off
title('Informacije')
str=char('Broj Generacija:',num2str(broj));
text(0.2,0.8,0.9,str,'FontSize',15)
str=char('Fitnes:',num2str(fitmax(broj,:)));
text(0.2,0.25,0.9,str,'FontSize',14)
str=char('Najbolja x koordinata:', num2str(xmin));
text(-0.9,0.8,0.9,str,'FontSize',14)
```

```
      str=char('Najbolja y koordinata:', num2str(ymin));
      text(-0.9,0.25,0.9,str,'FontSize',14)
      str=char('Najbolji f(x,y)=', num2str(min(zmin)));
      text(-0.9,-0.3,0.9,str,'FontSize',14)
      switch funkcija
         case 1 % Ackley
            str=char('fmin(x,y)=0  za x=0, y=0');
         case 2 % Rastrigin
            str=char('fmin(x,y)=0  za x=0, y=0');
         case 3 % Sferna
            str=char('fmin(x,y)=0  za x=0, y=0');
         case 4 % Drop wave
            str=char('fmin(x,y)=0  za x=0, y=0');
         case 5 % Easom
            str=char('fmin(x,y)=0  za x=pi, y=pi');
         case 6 % Holder table
            str=char('fmin(x,y)=0  za x=+/-8.05502, y=+/-9.66469');
         case 7 % Eggholder
            str=char('fmin(x,y)=0  za x=512, y=404.2319');
         case 8 % Schwefel26
            str=char('fmin(x,y)=0  za x=420.9687, y=420.9687');
         case 9 % Griewank
            str=char('fmin(x,y)=0  za x=0, y=0');
      end
      text(-0.9,-0.80,0.9,str,'FontSize',14)
      pause(.05);

      if min(fitnes)<0.0001
         break
      end
   end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Kraj
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Genetski algoritam s realnim vrijednostima
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all
clc

funkcija=[0];
lf=1;
while funkcija<lf

   %  odabir funkcije
   choice=menu('Odaberite optimizacijsku funkciju','Ackley','Rastrigin','Sferna','Drop wave',...
      'Easom','Holder table','Eggholder','Schwefel26','Griewank','Kraj rada');
   funkcija=[1,2,3,4,5,6,7,8,9,10];
   lf=length(funkcija);
   funkcija=funkcija(choice);

   if funkcija==10
      break
   end

   % Pocetni parametri
   prompt={'Upisite zeljeni broj gena:','Upisite zeljeni broj iteracija:'};% Tekst za zeljenu
varijablu
   answer=inputdlg(prompt);
   B = str2num(answer{1});     % Broj gena - mora biti paran
   ite = str2num(answer{2});   % Zeljeni broj iteracija


   alfa=0.9;  % vrijednost kod krizanja
   pc=0.15; % vjerojatnost za krizanje
   pm=0.4; % vjerojatnost za mutaciju

   SUM=0;
   [LB,UB,zmin,zmax]=poc_UvjFunkc(funkcija);    % Pocetni uvjeti ovisno o funkciji
   diskr=(UB-LB)/100;

   % Funkcija
   xx=[LB:diskr:UB];
   yy=[LB:diskr:UB];
   zz=[0];

   lx=length(xx);
   ly=length(yy);

   for i=1:lx;
      xi=xx(i);
      for j=1:ly;
```

```
        yi=yy(j);
        zi=odabir_OptFunkc(funkcija,xi,yi);
        zz(j,i)=zi;
    end
end

% Inicijalizacija

dim=2;  % Broj dimenzija
x1=[0];
y1=[0];
fitnes=[0];
fitmax=[0];
novo=[B,dim];
rod=[B,dim];
elitx=[0];
elity=[0];
pop=[];
pop_fun=[];
proba=[];

for i=1:dim
    pop(:,i)=LB+rand(B,1)*2*UB;
end

for broj=1:ite

    if broj~=1
        elit=randi([1,B]);
        pop=novo;
        pop(elit,1)=elitx;
        pop(elit,2)=elity;
    end

    % Odredivanje fitnesa
    for i=1:B
        pop_fun(i,1)=odabir_OptFunkc(funkcija,pop(i,1),pop(i,2));
    end
    [fitnes(broj,:),minMjesto]=min(pop_fun);

    % ELITIZAM
    for i=1:B
        for j=1:B
            proba(i,j)=odabir_OptFunkc(funkcija,pop(i,1),pop(j,2));
        end
    end

    [fmin0,index0]=min(proba(:));
    [red,stupac]=ind2sub(size(proba),index0);    % y- red, x- stupac
```

```
elitx=pop(red,1);
elity=pop(stupac,2);

xmin=pop(minMjesto,1);
ymin=pop(minMjesto,2);
zmin=fitnes(broj,:);        % za graf

% Vjerojatnost izbora pojedinog gena:
a=1./pop_fun;
vjerojatnost=a/sum(a);

% Generiranje parova za reprodukciju
for i=1:B
   r=rand(1,1);
   for j=1:B
      SUM=SUM+vjerojatnost(j);
      if SUM>r
         rod(i,:)=pop(j,:);
         SUM=0;
         break
      end
   end
end

% Krizanje
if rand<=pc
   k=randi([1 B-1]);
   for i=1:k
      novo(i,:)=rod(i,:);
   end
   for i=k+1:B
      novo(i,1)=alfa*rod(i,1)+(1-alfa)*rod(i,2);
      novo(i,2)=alfa*rod(i,2)+(1-alfa)*rod(i,1);
   end
else
   novo=rod; % ako nema krizanja, kopiraju se roditelji
end

% Mutacija
for i=1:B
   for j=1:dim
      if rand<=pm     % ako dolazi do mutacije clan na i,j mjestu se mijenja
         novo(i,j)=LB+rand*2*UB;
      end
   end
end

fitmax(broj,:)=zmax/(zmax*0.999999+fitnes(broj,:));

clf
```

```
figure(1)
subplot(2,2,[1,3])
surf(xx,yy,zz,'EdgeColor','none','LineStyle','none'), hold on
scatter3(pop(:,1),pop(:,2),pop_fun,'MarkerEdgeColor','k','MarkerFaceColor','g')
scatter3(xmin,ymin,zmin,'MarkerEdgeColor','k','MarkerFaceColor','r'), hold off
xlabel('x')
ylabel('y')
zlabel('f(x,y)')
title('Prikaz rada algoritma')

subplot(2,2,2)
x1=linspace(0,broj,broj);
y1=fitmax;
semilogx(x1,y1,x1,1.05);
xlabel('Broj generacija')
ylabel('Fitnes')
title('Konvergencija GA')
axis([0 broj 0 1.05]);

subplot(2,2,4)
x1=0;
y1=0;
plot(x1,y1), axis off
title('Informacije')
str=char('Broj Generacija:',num2str(broj));
text(0.2,0.8,0.9,str,'FontSize',15)
str=char('Fitnes:',num2str(fitmax(broj,:)));
text(0.2,0.25,0.9,str,'FontSize',14)
str=char('Najbolja x koordinata:', num2str(xmin));
text(-0.9,0.8,0.9,str,'FontSize',14)
str=char('Najbolja y koordinata:', num2str(ymin));
text(-0.9,0.25,0.9,str,'FontSize',14)
str=char('Najbolji f(x,y)=', num2str(min(zmin)));
text(-0.9,-0.3,0.9,str,'FontSize',14)
switch funkcija
   case 1 % Ackley
      str=char('fmin(x,y)=0  za x=0, y=0');
   case 2 % Rastrigin
      str=char('fmin(x,y)=0  za x=0, y=0');
   case 3 % Sferna
      str=char('fmin(x,y)=0  za x=0, y=0');
   case 4 % Drop wave
      str=char('fmin(x,y)=0  za x=0, y=0');
   case 5 % Easom
      str=char('fmin(x,y)=0  za x=pi, y=pi');
   case 6 % Holder table
      str=char('fmin(x,y)=0  za x=+/-8.05502, y=+/-9.66469');
   case 7 % Eggholder
      str=char('fmin(x,y)=0  za x=512, y=404.2319');
   case 8 % Schwefel26
```

```
            str=char('fmin(x,y)=0  za x=420.9687, y=420.9687');
         case 9 % Griewank
            str=char('fmin(x,y)=0  za x=0, y=0');
      end
      text(-0.9,-0.80,0.9,str,'FontSize',14)
      pause(.05);

      if min(fitnes)<0.00001
         break
      end
   end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Kraj
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Funkicija za odabir pocetnih uvjeta
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function  [LB,UB,zmin,zmax]  = poc_UvjFunkc(funkcija)

switch funkcija
   case 1 % Ackley
      UB=32;    % max vrijednost xx i yy
   case 2 % Rastrigin
      UB=5;
   case 3 % Sferna
      UB=5;
   case 4 % Drop wave
      UB=5;
   case 5 % Easom
      UB=20;
   case 6 % Holder table
      UB=10;
   case 7 % Eggholder
      UB=512;
   case 8 % schwefel26
      UB=500;
   case 9 % Griewank
      UB=600;
end
LB=-UB;  % min vrijednost xx i yy
zmin=0;  % min vrijednost opt funkcije
zmax=1;  % max vrijednost opt funkcije za graf za bolji prikaz fitnesa
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Funkicija za odabir optimizacijske funkcije
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function  z = odabir_OptFunkc(funkcija,xi,yi)
switch funkcija
   case 1 % Ackley
      zi=-20*exp(-0.2*sqrt(0.5*(xi^2+yi^2)))-
exp(0.5*(cos(2*pi*xi)+cos(2*pi*yi)))+exp(1)+20;
   case 2 % Rastrigin
      zi=20+(xi^2-10*cos(2*pi*xi))+(yi^2-10*cos(2*pi*yi));
   case 3 % Sferna
      zi=xi^2+yi^2;
   case 4 % Drop wave
      zi=1-(1+cos(12*sqrt(xi^2+yi^2)))/(0.5*(xi^2+yi^2)  + 2);
   case 5 % Easom
      zi=1+(-cos(xi)*cos(yi))*(exp(-(xi-pi)^2-(yi-pi)^2));
   case 6 % Holder table
      zi=19.2085-abs(sin(xi)*cos(yi)*exp(abs(1-sqrt(xi^2+yi^2)/pi)));
   case 7 % Eggholder
      zi=959.6407-(yi+47)*sin(sqrt(abs(xi/2+yi+47)))-xi*sin(sqrt(abs(xi-yi-47)));
   case 8 % Schwefel26
      zi=418.9829*2-(xi*sin(sqrt(abs(xi)))+yi*sin(sqrt(abs(yi))));
   case 9 % Griewank
      zi=(xi^2+yi^2)/4000-(cos(xi/sqrt(1))*cos(yi/sqrt(2)))+1;
end
z=zi;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Optimizacija rojem cestica
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all
clc

funkcija=[0];
lf=1;
while funkcija<lf
    x0=[0];
    y0=[0];
    z=[0];
    f1=[0];
    %  odabir funkcije
    choice=menu('Odaberite optimizacijsku funkciju','Ackley','Rastrigin','Sferna',...
        'Drop wave','Easom','Holder table','Eggholder','Schwefel26','Griewank','Kraj rada');
    funkcija=[1,2,3,4,5,6,7,8,9,10];
    lf=length(funkcija);
    funkcija=funkcija(choice);

    if funkcija==10
        break
    end

    % Pocetni parametri
    prompt={'Upisite zeljenu velicinu populacije:','Upisite zeljeni broj iteracija:'}; % Tekst za
zeljenu varijablu
    answer=inputdlg(prompt);
    n = str2num(answer{1});     % Velicina populacije
    ite = str2num(answer{2});   % Zeljeni broj iteracija

    [LB,UB,zmin,zmax]=poc_UvjFunkc(funkcija);    % Pocetni uvjeti ovisno o funkciji
    diskr=(UB-LB)/100;

    wmax=0.9;   % Inercija
    wmin=0.4;   % Inercija
    c1=1.95;       % faktor ubrzanja (kognitivni)
    c2=2.05;       % faktor ubrzanja (socijalni)

    % Funkcija

    xx=[LB:diskr:UB];
    yy=[LB:diskr:UB];
    zz=[0];

    lx=length(xx);
    ly=length(yy);
```

```
for i=1:lx;
   for j=1:ly;
      zz(j,i)=odabir_OptFunkc(funkcija,xx(i),yy(j));
   end
end

% Inicijalizacija
for i=1:n
   x0(i,:)=(LB+rand()*(UB-LB));
   y0(i,:)=(LB+rand()*(UB-LB));
end

x=x0;              % pocetna populacija koord x
y=y0;              % pocetna populacija koord y
vx=0.1*x0;         % pocetna brzina u smijeru osi x
vy=0.1*y0;         % pocetna brzina u smijeru osi x

for i=1:n;
   for j=1:n;
      z(j,i)=odabir_OptFunkc(funkcija,x(i),y(j));
   end
end

f=z;               % f - trenutni fitnes
f0=f;              % f0 - najbolji fitnes
fit=(sum(f(:))/n^2);
fitnes=zmax/(zmax*0.9999+fit); % pocetni prosjecni fitnes populacije
fitmax=fitnes; % pocetni najbolji fitnes populacije

[fmin0,index0]=min(f0(:));
[red_0,stupac_0] = ind2sub(size(f0),index0);    % y- red, x- stupac

pbestx=x0;      % najbolja osobna vrijednost polozaja svakog clana populacije
pbesty=y0;
gbestx=x0(stupac_0,:);    % x vrijednost polozaja najboljeg clana unutar populacije
gbesty=y0(red_0,:);       % y vrijednost polozaja najboljeg clana unutar populacije

for broj=1:ite

   w=wmax-(wmax-wmin)*(broj/ite);    % Obnavljanje inercije

   % Obnavljanje brzine i polozaja
   for i=1:n
      vx(i,:)=w*vx(i,:)+c1*rand()*(pbestx(i,:)-x(i,:))+c2*rand()*(gbestx-x(i,:));
      vy(i,:)=w*vy(i,:)+c1*rand()*(pbesty(i,:)-y(i,:))+c2*rand()*(gbesty-y(i,:));
      x(i,:)=x(i,:)+vx(i,:)*0.5;
      y(i,:)=y(i,:)+vy(i,:)*0.5;
   end

   % Popravljanje granica
```

```
for i=1:n
   if x(i,:)<LB || x(i,:)>UB
      x(i,:)=(LB+rand()*(UB-LB));
   end
   if y(i,:)<LB || y(i,:)>UB
      y(i,:)=(LB+rand()*(UB-LB));
   end
end

% Procjena fitnesa i obnavljanje pbest i fitnesa
for i=1:n;
   for j=1:n;
      f(j,i)=odabir_OptFunkc(funkcija,x(i),y(j));
   end
end

for i=1:n;
   f1(i,:)=odabir_OptFunkc(funkcija,x(i),y(i));     % f1- za graf
end

for i=1:n
   for j=1:n
      if f(i,j)<f0(i,j)
         pbestx(j,:)=x(j,:);
         pbesty(i,:)=y(i,:);
         f0(i,j)=f(i,j);
      end
   end
end

[fmin,index]=min(f0(:));    % trazenje najboljeg clana populacije
[red,stupac] = ind2sub(size(f0),index);    % y- red, x- stupac

% Obnavljanje gbest i najboljeg fitnesa
if fmin<fmin0
   gbestx=x(stupac,:);    % x vrijednost polozaja najboljeg clana unutar populacije
   gbesty=y(red,:);       % y vrijednost polozaja najboljeg clana unutar populacije
   fmin0=fmin;
end

fit(broj,:)=(sum(f(:))/n^2);
fitnes(broj,:)=zmax/(zmax*0.99999+fit(broj,:)); % fitnes - prosjecni fitnes populacije
fitmax(broj,:)=zmax/(zmax*0.99999+fmin);    % fitmax - najbolji fitnes populacije

% Graf
clf
figure(1)
subplot(3,2,[1,3])
scatter(x,y,'r'), grid on
axis([LB UB LB UB]);
```

```
xlabel('x')
ylabel('y')
title('Prikaz rada PSO algoritma')
text(UB*0.8,UB*0.8,num2str(broj),'FontSize',15)
str=char('Najbolja x koordinata:', num2str(gbestx));
text(LB*0.9,UB*0.84,0.9,str,'FontSize',14)
str=char('Najbolja y koordinata:', num2str(gbesty));
text(LB*0.9,UB*0.54,0.9,str,'FontSize',14)
str=char('Najbolji f(x,y)=', num2str(min(fmin)));
text(LB*0.9,UB*0.24,0.9,str,'FontSize',14)
switch funkcija
   case 1 % Ackley
      str=char('fmin(x,y)=0  za x=0, y=0');
   case 2 % Rastrigin
      str=char('fmin(x,y)=0  za x=0, y=0');
   case 3 % Sferna
      str=char('fmin(x,y)=0  za x=0, y=0');
   case 4 % Drop wave
      str=char('fmin(x,y)=0  za x=0, y=0');
   case 5 % Easom
      str=char('fmin(x,y)=0  za x=pi, y=pi');
   case 6 % Holder table
      str=char('fmin(x,y)=0  za x=+/-8.05502, y=+/-9.66469');
   case 7 % Eggholder
      str=char('fmin(x,y)=0  za x=512, y=404.2319');
   case 8 % Schwefel26
      str=char('fmin(x,y)=0  za x=420.9687, y=420.9687');
   case 9 % Griewank
      str=char('fmin(x,y)=0  za x=0, y=0');
end
text(LB*0.9,LB*0.8,0.9,str,'FontSize',14)

subplot(3,2,[2,4])
surf(xx,yy,zz,'EdgeColor','none','LineStyle','none'),  hold on
scatter3(x,y,f1,'filled','r'), hold off
xlabel('x')
ylabel('y')
zlabel('f(x,y)')
title('Prikaz rada PSO algoritma u 3D')

subplot(3,2,5)
x1=linspace(0,broj,broj);
y1=fitnes;
plot(x1,y1), grid on
xlabel('Broj iteracija');
ylabel('Prosjecni fitnes populacije');
axis([0 broj 0 1.05]);
title('Konvergencija PSO algoritma');
str=char('', num2str(fitnes(broj,:)));
text(broj*0.8,0.3,0.9,str,'FontSize',14)
```

```
        subplot(3,2,6)
        x1=linspace(0,broj,broj);
        y1=fitmax;
        plot(x1,y1), grid on
        xlabel('Broj iteracija');
        title('Fitnes najboljeg clana populacije');
        axis([0 broj 0 1.05]);
        str=char('', num2str(fitmax(broj,:)));
        text(broj*0.8,0.3,0.9,str,'FontSize',14)
        pause(.05);

        if max(fitnes)>0.99999
            break
        end
    end

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Kraj
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```