**Univerzitet u Banjoj Luci**

**Elektrotehnički fakultet**

**Katedra za automatiku**

**Metodi vještačke inteligencije**

Izvještaj o urađenom projektnom zadatku

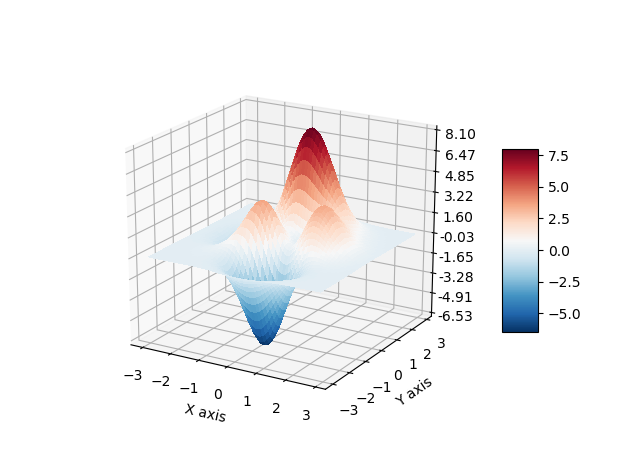
**Genetički algoritmi**

Student:

Nikola Blagojević, 11136-15

# Opis problema

Na slici je prikazan grafik funkcije dvije promjenljive.



Slika 1 - Grafik funkcije dvije promjenljive

Analitički izraz funkcije čiji je grafik prikazan na Slici1 dat je sa:

Korišćenjem genetičkog algoritma potrebno je odrediti globalni minimum i globalni maksimum posmatrane funkcije na intervalu na kome je funkcija grafički prikazana na Slici1.

# Rješenje

Da bi se odredilo rješenje ovog problema potrebno je opseg vrijednosti promjenljivih *x* i *y* u kojem data funkcija *z* ima minimum i maksimum. Ukoliko funkcija ima minimum (maksimum), to znači da postoji neki interval *I* = (Gd,Gg) takav da postoji i za koje važi:

Sa slike 1 zaključujemo da je taj interval isti za vrijednosti *x* i *y* i iznosi *I* = [-3,3].

Nakon utvrđivanja intervala za vrijednosti *x* i *y*, potrebno je izabrati početnu populaciju.

Početna populacija je birana pseudoslučajno. Za potrebe genetičkog algoritma moguća rješenja su kodovana binarno. Kodovanje je izvršeno tako da sve dekodovane vrijednosti pripadaju intervalu na kom se traži rješenje procesom binarnog kodovanja decimalnih brojeva.

Dobijeni interval je podijeljen na manje intervale sa širinom intervala od 10-p, pri čemu je *p* željena preciznost i pri izvođenju eksperimenata rađeno je sa preciznošću *p=2*. Svaki broj iz manjeg intervala zamijenjen je donjom granicom tog intervala i kodovan rednim brojem tog intervala. Binarno kodovanje decimalnog broja je tada binarno kodovanje tog rednog broja.

Veza između preciznosti i broja bita potrebnih za kodovanje data je sljedećom formulom:

Pri čemu su:

* n – broj bita za kodovanje
* Gg – gornja granica intervala
* Gd – donja granica intervala
* p – tražena preciznost

Pri testiranju korišćene su sljedeće vrijednosti:

Pa se dobija da je:

U Tabeli 1. je dato nekoliko primjera kodovanja decimalnih brojeva iz intervala *I*.

Tabela 1 - Nekoliko kodova za brojeve iz intervala I

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Za kodovanje proizvoljnog broja iz intervala *I* je korišćena funkcija *encode* koja vraća cijeli broj po formuli:

Gdje je *x* vrijednost realnog broja koji se koduje.

Za dekodovanje dobijenih cijelih brojeva u realne brojeve koristi se funkcija *decode* koja vraća realni broj iz cijelog broja po sljedećoj formuli:

Gdje je *d* – vrijednost cijelog broja koji se dekoduje.

Nakon definisanja kodovanja potrebno je preći na izbor početne pretpostavke, tj. izbor početne populacije za genetički algoritam. Prilikom generisanja vrijednosti za početnu populaciju koristi se funkcija koja daje broj iz intervala [0, 1], te je neophodno preslikati brojeve iz ovog intervala u traženi interval *I*. Ovo se postiže po sljedećoj formuli:

Gdje je *r* – slučajno generisana vrijednost na intervalu [0, 1].

Budući da je tražena funkcija *z* funkcija od dvije promjenljive, potrebno je generisati populaciju za vrijednosti *x* i *y* koje su predstavljene sa dva niza: *population\_x, population\_y*.

Izgled početne populacije dat je u Tabeli 2.

Ciklus genetičkog algoritma započinje ocjenom kvaliteta rješenja. Za ocjenu kvaliteta koristi se funkcija koja ima najmanju vrijednost za najlošije rješenje, a najveću vrijednost za najbolje najbolje rješenje. Ova funkcija se naziva *fitnes funkcija*.

Za potrebe traženja minimuma funkcije *z* koristi se sljedeća fitnes funkcija:

Dok se za potrebe traženja maksimuma funkcije *z* koristi sljedeća fitnes funkcija:

Za izbor fintes funkcije za minimum ili maksimum koristi se parametar *minOrMax.* Ukoliko je *minOrMax =* 1 biće izabrana fitnes funkcija za minimum i algoritam će tražiti minimum funkcije *z*, dok za bilo koju drugu vrijednost biće izabrana fitnes funkcija za maksimum i algoritam će tražiti maksimum funkcije *z*.

U Tabeli 3. dat je prikaz populacije sa vrijednostima funkcije *z* i odgovarajućim vrijednostima fitnes funkcije za traženje maksimuma.

Takođe je moguće dati ocjenu cijele populacije:

Potrebno je napomenuti da ocjena populacije ne znači nužno i bolju prilagođenost.

Nakon ocjenjivanja populacije, potrebno je odrediti roditelje za sljedeću generaciju genetičkog algoritma. Ovaj proces se naziva selekcija. Najbolje prilagođena jedinka ima najveću šansu da postane roditelj ili jedinka u sljedećoj generaciji. Za izbor jedinke korišćena je ruletska selekcija. Ruletska selekcija realizovana je na sljedeći način:

* svaka jedinka posjeduje vjerovatnoću izbora jedinke koja se računa po formuli:

,

* svaka jedinka „zauzima“ određeni interval na ruletu koji definiše kumulativna vjerovatnoća data sa:

Okretanje točka ruleta se simulira generisanjem pseudoslučajnog broja r iz intervala [0,1]. Tada važi ili ili . U prvom slučaju bira se hromozom rednog broja 0, dok se u drugom slučaju bira hromozom rednog broja *i*. Proces selekcije se završava kada se izabere populacija odgovarajuće veličine.

Potrebno je napomenuti da je proces selekcije realizovan u dva načina rada. Prvi način je neelitistička selekcija što podrazumijeva da će se sve jedinke sljedeće generacije birati na slučajan način. Drugi način je elitistička selekcija, tj. dvije najbolje jedinke iz prethodne generacije direktno prelaze u sljedeću, dok se ostale biraju na slučajan način.

Prilikom testiranja algoritma uočeno je da se u slučaju elitističke selekcije dobijaju značajno bolji rezultati i algoritam u manje generacija daje traženo rješenje. Ovo je posljedica toga da najbolje jedinke direktno prelaze u sljedeću generaciju, ali isto tako imaju i veću vjerovatnoću prelaska te se u narednoj generaciji dobija veći broj bolje prilagodjenih jedinki nego u slučaju neelitističke selekcije.

Sljedeća faza algoritma je rekombinacija. U ovoj fazi neki od hromozoma će ostati nepromijenjeni, dok će se neki ukrštati. Ukrštanje je realizovano

Naročito obratiti pažnju na eventualne inovacije u algoritmu i detaljno ih obrazložiti. Voditi računa o načinu izbora pojedinih parametara. Dati objašnjenje izbora odgovarajućih parametara, funkcija i sl.

Uočiti kako varijacije pojedinih parametara utiču na kvalitet rješenja. Pokušati doći do optimalnih parametara koji za najkraće vrijeme daju zadovoljavajuće rezultate.

Potrebno je izvršiti minimalno 10 eksperimenata. Rezultate prikazati na pogodan način: tabelarno, grafički i sl.

# Primjeri formatiranja

U ovom Poglavlju je dat način referenciranja tabela, slika, listinga koda, matematičkih formula, kao i sami prikazi istih. Numeracija je formata *BrojPoglavlja.RedniBrojPrikazaUPoglavlju*.

Primjer tabelarnog prikaza dat je u Tabeli 4.1.

Tabela 4.1 *Primjer tabelarnog prikaza*

|  |  |  |
| --- | --- | --- |
|  | **Atribut 1** | **Atribut 2** |
| **Varijanta 1** | Rezultat 1.1 | Rezultat 1.2 |
| **Varijanta 2** | Rezultat 2.1 | Rezultat 2.2 |

Dozvoljene su varijacije izgleda, ali je neophodno da izgled tabela bude unificiran u cijelom izvještaju.

Primjer prikaza slikom dat je na Slici 4.1.



Slika 4.1 *Primjer prikaza slikom*

Primjer listinga sa kodom dat je u Listingu 4.1.

Listing 4.1 *Primjer prikaza koda*

|  |
| --- |
| #include <stdio.h>  int main(int argc, char \*argv[])  {  printf(“Hello, world!\n”);  return 0;  } |

Primjer matematičkog zapisa formule dat je sa (4.1).

(4.1)

Zadržati font, veličinu, formatiranje paragrafa i formatiranje papira.

Ovo poglavlje služi isključivo za ilustraciju prikaza slika, tabela i jednačina te ne treba kao takvo da se nalazi u izvještaju projektnog zadatka.

# Zaključak

Unutar zaključka potrebno je dati vlastito mišljenje o konkretnom algoritmu, prednostima i nedostacima istog koji su uočeni tokom izrade projektnog zadatka. Za koje parametre algoritam daje najbolje rješenje i zašto? Da li postoji način da se algoritam unaprijedi i poboljša? Koje su praktične koristi analiziranog algoritma i sl.?