**Raport Tema 2**

***Cerinta:***

Sa se implementeze un algoritm genetic care sa minimizeze urmatoarele functii: Rastrigin, Griewangk, Rosenbrock, Six-hump camel back function   
Sa se redacteze un raport care sa contina:

* descrierea algoritmului (reprezentarea, selectia, operatorii utilizati)
* influenta valorilor parametrilor
* comportarea algorimului pe dcele 4 functii de tset in 5, 10 si 30 dimensiuni (cea mai slaba comportare - max; cea mai buna comportare - min; comportamentul mediu - media -> pentru 15 executii ale algoritmului genetic)
* comparatii intre cele 3 metode implementate pentru optimizare numerica (HC, SA, GA); in acest scop se vor realiza 4 grafice corespunzatoare celor 4 functii in 30 dimensiuni pe care reprezentati evolutia celei mai bune solutii in timpul unei rulari pentru fiecare algoritm, in functie de numarul de apeluri a functiei de minimizat.

**Rastrigin:** f6(x)=10·n+sum(x(i)^2-10·cos(2·pi·x(i))),

i=1:n; -5.12<=x(i)<=5.12.

C:\Users\luka\Desktop\fcnindex-13.gif

global minimum: f(x)=0; x(i)=0, i=1:n.

**Griewangk:** f8(x)=sum(x(i)^2/4000)-prod(cos(x(i)/sqrt(i)))+1,

i=1:n, -600<=x(i)<= 600.

http://www.geatbx.com/docu/fcnindex-18.gif

global minimum:

f(x)=0; x(i)=0, i=1:n.

**Rosenbrock’s valley:**

http://www.geatbx.com/docu/fcnindex-10.gif

f2(x)=sum(100·(x(i+1)-x(i)^2)^2+(1-x(i))^2)

i=1:n-1; -2.048<=x(i)<=2.048.

global minimum:

f(x)=0; x(i)=1, i=1:n.

**Six-hump camel back**

fSixh(x1,x2)=(4-2.1·x1^2+x1^4/3)·x1^2+x1·x2+(-4+4·x2^2)·x2^2

-3<=x1<=3, -2<=x2<=2.

C:\Users\luka\Desktop\fcnindex-41 (1).gif

global minimum:

f(x1,x2)=-1.0316; (x1,x2)=(-0.0898,0.7126), (0.0898,-0.7126).

**Pseudocod**

INITIALIZARE **begin**

***t* := 0**

genereaza *P(t)*

evalueaza *P(t)*

ITERARE **while** (not CONDITIE\_OPRIRE)  **do begin**

*t* := *t* + 1

selecteaza *P(t)* din *P(t-1)*

recombina *P(t)*

evalueaza *P(t)*

**end**

### Reprezentarea

### Se va utiliza reprezentarea binara:

* siruri binare: spaţiul de căutare se va disctretiza până la o anumită precizie 10-d. Un interval [*a*, *b*] va fi împărţit în N = (b-a)\*10d subintervale egale.
* Pentru a putea reprezenta cele (b-a)\*10d valori, este nevoie de un număr
* n = parte\_intreaga\_superioara(log2(N)) de biţi.
* Lungimea şirului de biţi care reprezintă o soluţie candidat va fi suma lungimilor reprezentărilor pentru fiecare parametru al funcţiei de optimizat. In momentul evaluarii solutiei (apelul functiei de optimizat) este necesara decodificarea fiecarui parametru reprezentat ca sir de biti in numar real, dupa formula: Xreal = a+decimal(xbiti)\*(b-a)/(2n-1)

**Explicatii pseudocod:**

genereaza *P(t)*

**Genreaza o *populatie* de indivizi random**

CONDITIE\_OPRIRE

**Se va folosi un numar maxim de generatii *gen\_limit***

selecteaza *P(t)* din *P(t-1)*

**Metoda de selectie folosita va fi Roata norocului:**

**Roata norocului**   
Aceasta este schema de selectie introdusa de Holland in algoritmul genetic original. Numarul estimat de copii pe care le primeste un individ este proportional cu fitnessul sau impartit la fitnessul total al populatiei.

EVALUEAZA P **for** i:=0 to POP\_SIZE

eval[i]=f(P(i))

FITNESSUL TOTAL **for** i:=0 to POP\_SIZE

T+=eval[i]

PROB.SEL.INDIVIDUALE **for** i:=0 to POP\_SIZE

p[i]=eval[i]/T

PROB.SEL.CUMULATE q[0] = 0

**for** i:=0 to POP\_SIZE

q[i+1]=q[i]+p[i]

SELECTIA **for** i:=0 to POP\_SIZE

genereaza uniform *r* in (0,1]

selecteza pentru supravieture individul j

pentru care q[j]<r<=q[j+1]

Pentru evaluare va fi folosita functia fitness pentru minimizare. Din moment ce primele 3 functii vor avea numai valori pozitive se va folosit functia fitness ***f(x)=1/g(x)***, iar pentru Six-hump se va folosi functia ***f(x)=g(x)+C***, unde *g(x)* reprezinta rezultatul functiei pe care vrem sa o minimizam aplicate pe individ iar *C* reprezinta o constanta care va garanta ca functia fitness sa fie pozitiva (pentru Six-hump: C=1.0317).

recombina *P(t)*

In urma selectiei se vor lua perechi de cate 2 indivizi din populatia noua si se vor aplica operatorii genetici cu o anumita probabilitate:

**Incrucisarea** are ca scop schimbul de informatie genetica intre doi sau mai multi cromozomi.

* Încrucişarea cu un punct de tăiere, ales aleator.
* **01 001011** \_\ **01** 111100

10 111100 / 10 **001011**

**Mutatia** - modificarea unei gene alese aleatoriu

0010110**0**0 -> 0010110**1**0

*Pentru incrucisare a fost folosita o probabilitate de 80% de a aplica operatorul, iar pentru mutatie fiecarui individ din pereche o probabilitate de 10%.*

evalueaza *P(t)*

Se asigneaza vectorii curenti cu noile date si se repeta pasii pana la conditia de oprire.

**Rezultatul algoritmului genetic va fi reprezentat de cel mai bun individ din ultima generatie**

Rezultate:

|  |  |  |  |
| --- | --- | --- | --- |
| **Rastrigin** | ***global min = 0*** |  |  |
| *test\_no* | *n=5 gens=5000 pop\_size=300* | *n=10 gens=5000 pop\_size=140* | *n=30 gens=5000 pop\_size=100* |
| 1 | 1.42E-14 | 8.427303002 | 80.33386244 |
| 2 | 0.995390173 | 5.548767769 | 97.64382155 |
| 3 | 1.997700347 | 12.57881384 | 108.899024 |
| 4 | 1.002288542 | 12.63664876 | 78.94600077 |
| 5 | 1.42E-14 | 13.47728002 | 98.56718983 |
| 6 | 1.42E-14 | 7.741859979 | 94.53827512 |
| 7 | 1.42E-14 | 12.28369104 | 104.3324186 |
| 8 | 0.997838225 | 8.142475726 | 81.96631964 |
| 9 | 0.995179382 | 10.26629571 | 77.81434438 |
| 10 | 3.490239007 | 10.23989754 | 113.1640816 |
| 11 | 1.994932738 | 12.86301907 | 97.57248183 |
| 12 | 2.003646253 | 10.46549985 | 122.2876984 |
| 13 | 0.995557338 | 11.59101552 | 98.62560086 |
| 14 | 0.996328734 | 12.15567252 | 59.18580687 |
| 15 | 0.996057827 | 7.949413802 | 78.42194804 |
| **min** | **1.42E-14** | **5.548767769** | **59.18580687** |
| **max** | **3.490239007** | **13.47728002** | **122.2876984** |
| **mean** | **1.097677238** | **10.42451028** | **92.81992494** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Griewangk** | ***global min = 0*** |  |  |
| *test\_no* | *n=5 gens=5000 pop\_size=300* | *n=10 gens=5000 pop\_size=140* | *n=30 gens=5000 pop\_size=100* |
| 1 | 1.04E-01 | 0.331872718 | 4.465292978 |
| 2 | 0.116166227 | 0.084339177 | 2.345545357 |
| 3 | 0.394268592 | 0.080240238 | 2.953569276 |
| 4 | 0.295382106 | 1.036520184 | 2.34189349 |
| 5 | 2.76E-01 | 0.946824114 | 2.453844176 |
| 6 | 2.38E-01 | 0.983903462 | 4.014568834 |
| 7 | 1.88E-01 | 1.020420416 | 2.353959968 |
| 8 | 0.330501425 | 0.338798578 | 2.380177596 |
| 9 | 0.086903683 | 1.004409351 | 2.437957858 |
| 10 | 0.141156223 | 0.126106382 | 2.712371274 |
| 11 | 0.031989341 | 0.300727444 | 3.707780314 |
| 12 | 0.070441312 | 0.463374164 | 3.14929427 |
| 13 | 0.697285341 | 0.431648701 | 3.053989215 |
| 14 | 0.353653499 | 0.585537689 | 3.047830466 |
| 15 | 0.253504226 | 0.054892637 | 3.928147263 |
| **min** | **3.20E-02** | **0.054892637** | **2.34189349** |
| **max** | **0.697285341** | **1.036520184** | **4.465292978** |
| **mean** | **0.238506209** | **0.519307684** | **3.023081489** |
| **Rosenbrock** | ***global min = 0*** |  |  |
| test\_no | n=5 gens=5000 pop\_size=300 | n=10 gens=5000 pop\_size=140 | n=30 gens=5000 pop\_size=100 |
| 1 | 1.37E+00 | 78.35135595 | 223.141077 |
| 2 | 0.273168764 | 7.876898029 | 346.1769302 |
| 3 | 3.032923931 | 6.876821992 | 187.0994964 |
| 4 | 1.414106695 | 0.954935053 | 72.16316406 |
| 5 | 2.73E+00 | 6.226021359 | 184.9953192 |
| 6 | 3.15E+00 | 10.44818844 | 210.0451273 |
| 7 | 2.64E+00 | 7.893019629 | 291.6542504 |
| 8 | 1.04921716 | 10.09980657 | 226.1284568 |
| 9 | 2.66914518 | 9.602356517 | 195.4896994 |
| 10 | 0.937008389 | 7.53118727 | 205.2978746 |
| 11 | 2.646367127 | 17.02759243 | 269.5416069 |
| 12 | 0.79068513 | 9.82929864 | 233.3211385 |
| 13 | 2.612260797 | 29.64699697 | 133.8416053 |
| 14 | 2.657162215 | 9.025630797 | 232.9301189 |
| 15 | 3.199469098 | 8.211599759 | 177.2438569 |
| **min** | **2.73E-01** | **0.954935053** | **72.16316406** |
| **max** | **3.199469098** | **78.35135595** | **346.1769302** |
| **mean** | **2.078450247** | **14.64011396** | **212.6046481** |

|  |  |
| --- | --- |
| **Six-hump** | ***global min ≈ -1.0316*** |
| *test\_no* | *n=2 gens=5000 pop\_size=300* |
| 1 | -1.03E+00 |
| 2 | -1.031323853 |
| 3 | -1.031285887 |
| 4 | -1.03153807 |
| 5 | -1.03E+00 |
| 6 | -1.02E+00 |
| 7 | -1.03E+00 |
| 8 | -1.0316262 |
| 9 | -1.031569419 |
| 10 | -1.031537892 |
| 11 | -1.031612961 |
| 12 | -1.03156939 |
| 13 | -1.031623344 |
| 14 | -1.031569422 |
| 15 | -1.031569424 |
| **min** | **-1.03E+00** |
| **max** | **-1.019694505** |
| **mean** | **-1.030729626** |