Генетичний алгоритм, Виконавець - Кроча Кирило, група ОМ-3

```
restart

with(LinearAlgebra):
with(RandomTools):
with(Statistics):
GenerationDec:=proc(xmin::Array, xmax::Array, M, N)
    #Ця функція генерує початкову популяцію G - матриця. в яку зипишуться сгенеровані
    значення, xmin, xmax - обмеження на кооординати, N,M - розмірність матриці
local i, j, rand1, G:
G:=Matrix(1..M, 1..N+1):
for i from 1 to M do
for j from 1 to N do
rand1:=RandomTools[Generate](integer(range=0..10^8)):
G[i,j]:=evalf((rand1*(xmax[j]-xmin[j])+10^8*xmin[j])/10^8):
end do:
end do:
return G;
end proc:

BinDecParam:=proc(Xmin, Xmax, Eps, M)
    # Допоміжна функція; Eps - точність обчислень для двійкового коду, xmin, xmax -
    обмеження на кооординати
local i, j, k, h, suma;
local NN, nn, dd:
nn:=[seq(0, i=1..M)]:
dd:=[seq(0, i=1..M)]:
for i from 1 to M do
nn[i]:=trunc(log[2](((Xmax[i]-Xmin[i])/Eps)))+1:
dd[i]:=(Xmax[i]-Xmin[i])/2^(nn[i]):
end do:
NN[1]:=0;
for k from 1 to M do
suma:=0;
for h from 1 to k do
suma:=suma+nn[h];
end do:
NN[k+1]:=suma;
```

```
end do;
return NN, nn, dd;
end proc:
```

Кодування

```
CodBinary1 := proc(xmin, d, nnn, xdec)
    # xmin- обмеження на кооординати, d - довжина інтервалів поділу, nnn -розмірність
    двійкового коду, xdec - число в десятичному представленні
local xbin1, xx, i :
 local xbin;

 xx := trunc( (xdec − xmin) / d ) :
 xbin1 := convert(xx, base, 2) :
xbin := [op(xbin1), seq(i·0, i = 1 ..nnn − nops(xbin1))] :
return xbin;
end proc:
```

```
 CodDecimal1 := proc(xmin, d, xbin)
    # xmin- обмеження на кооординати, d - довжина інтервалів поділу, nnn -розмірність
    двійкового коду, xbin - бінарна стрічка
 local n1, xdec2, xx, i :
local xdec :
 xx := convert(xbin, base, 2, 10) :
 n1 := nops(xx) :
 xdec2 := add(xx[i]·10^(i−1), i = 1 ..n1) :
 xdec := evalf(xmin + d· xdec2) :
 return xdec;
 end proc:
```

```
ACodDecimal1 := proc(Xmin, Gbin1, N, M, NN, nn, dd)
    #xmin, xmax - обмеження на кооординати, N,M - розмірність матриці, Gbin1 - бінарна
    матриця
 local i, j, xbin2, k, Gdec1 :

 Gdec1 := Matrix(1 ..N, 1 ..M + 1, fill = 0) :
 for i from 1 to N do
 for j from 1 to M do
 xbin2 := [seq(Gbin1[i, k], k = NN[j] + 1 ..NN[j + 1])] :

 Gdec1[i, j] := CodDecimal1(Xmin[j], dd[j], xbin2) :
 end do:
 end do:
 return Gdec1;
 end proc:
```

```
ACodBinary1 := proc(Xmin, Gdec1, N, M, NN, nn, dd)
```

```
    #xmin, xmax - обмеження на кооординати, N,M - розмірність матриці, Gdec1 - десятична
    матриця

 local i, j, k
 local Gbin, xbin :

Gbin := Matrix(1..N, 1..NN[M + 1], fill = 0) :

for i from 1 to N do
for j from 1 to M do
xbin := CodBinary1(Xmin[j], dd[j], nn[j], Gdec1[i, j]) :
 for k from NN[j] + 1 to NN[j + 1] do
 Gbin[i, k] := xbin[k − NN[j]] :
 end do:
 end do:
 end do:
 return Gbin;
 end proc:

 BiGr :=proc(Gbin1, N, M, NN)
    #N,M - розмірність матриці, Gbin1 - бінарна матриця
 local i, j :
 local GbinGr :
 GbinGr := Matrix(1..N, 1..NN[M + 1], fill = 0) :
for i from 1 to N do
 for j from 2 to (NN[M + 1]) do

 GbinGr[i, j] := (Gbin1[i, j] + Gbin1[i, j − 1]) mod 2;
 end do:
 GbinGr[i, 1] := Gbin1[i, 1] :
 end do:
 return GbinGr;
 end proc:

  GrBi :=proc(Gr, N, M, NN)
 local i, j, k, Suma :
 local Grbin :
 Grbin := Matrix(1..N, 1..NN[M + 1], fill = 0) :
 Suma := 0 :
 for i from 1 to N do
Suma := 0 :

Grbin[i, 1] := Gr[i, 1] :
 Suma := Gr[i, 1] :
 for j from 2 to (NN[M + 1]) do

 Suma := (Suma + Gr[i, j]) mod 2 :
Grbin[i, j] := Suma;
```

```
 end do:

 end do:
 return Grbin;
 end proc:
```

Normalization

```
 Normalization := proc(f, Gdec1, N, M)
 local i, j, k, t, min;
local fitness, nooo;
 fitness := Matrix(N, 1, fill = 0);
 min := 10000000000000;
for i from 1 to N do
 if min > f(seq(G[i, k], k = 1 ..M) ) then
 min := f(seq(G[i, k], k = 1 ..M) );
 end if:
 end do:
 for t from 1 to N do
 fitness[t] := f(seq(G[t, k], k = 1 ..M) ) + abs(min) + 1 :
 end do:
return fitness;

end proc:
```

```
 Normalization_like_in_text := proc(f, Gdec1, N, M)
 local i, j, k, t, min, max;
 local fitness;
 fitness := Matrix(N, 1, fill = 0);
 min := 10000000000000;
 max := -100000000000000;
for i from 1 to N do
 if min > f(seq(Gdec1[i, k], k = 1 ..M) ) then
 min := f(seq(Gdec1[i, k], k = 1 ..M) );
 end if:
  if max < f(seq(Gdec1[i, k], k = 1 ..M) ) then
 max := f(seq(Gdec1[i, k], k = 1 ..M) );
 end if:
 end do:
 for t from 1 to N do
 fitness[t] := (f(seq(Gdec1[t, k], k = 1 ..M) ) - min) / (max - min) :
 end do:
 return fitness;
```

end proc:

Селекція

Proportion := **proc**(*Gdec*, *N*, *M*)
#N - кількість рядків, fit_matrix – матриця значень фітнес функції
local *i*, *j*, *suma*, *Probs*;

Probs := *Matrix*(*N*, 1, *fill* = 0);
suma := 0;
for *i* **from** 1 **to** *N* **do**
suma := *suma* + *Gdec*[*i*, *M* + 1]:
end do:
for *j* **from** 1 **to** *N* **do**
$Probs[j] := \frac{Gdec[j, M+1]}{suma}$:
end do:
return *Probs*;
end proc:

Ranking :=**proc**(*N*, *M*)
#Gdec-Матриця, що сортована за значеннями фітнес функції N-рядки, M -стовпці
local *k*;
local *Probs*;
Probs := *Matrix*(*N*, 1, *fill* = 0);
for *k* **from** 1 **to** *N* **do**:
$Probs[k] := \frac{2 \cdot k}{N \cdot (N+1)}$;
end do:
return *Probs* :
end proc:

Panmixion :=**proc**(*N*, *M*)
local *i*, *j*, *Probs*;
Probs := *Matrix*(*N*, 1, *fill* = 0);
for *i* **from** 1 **to** *N* **do**
$Probs[i] = \frac{1}{N}$;
end do;
return *Probs* :
end proc:

Selection :=**proc**(*Gbin*, *Probs*, *N*, *M*, *NN*)
#Probs - Матриця ймовірностей, Selected - Матриця обраних на схрещування елементів

```
 local i, k, u, suma, j, l, a, val;
 local Shkala, Parents;

Shkala := Matrix(N + 1, 1, fill = 0);
 Parents := Matrix(2, NN[M + 1], fill = 0);
suma := 0 :
 for k  from 2 to N do
 suma := 0 :
 for i from 1 to k − 1 do
 suma := suma + Probs[i] :
 end do:
Shkala[k] := suma :
 end do:
Shkala[k] := 1 :
 a := rand(0.0 ..1.0) :
for l from 1 to N do
val := a( );
 for u  from 1 to 2 do
val := a( );

for j from 1 to N do
 if (val ≥ Shkala[ j, 1 ] and val ≤ Shkala[ j + 1, 1 ]) then
 Parents[u, 1 ..NN[M + 1 ]] := Gbin[ j, 1 ..NN[M + 1 ]];
 end if:
 end do:
 end do:
 end do:
 return Shkala, Parents;
 end proc:
```

Схрещування

```
 OnePointBreeding := proc(N, M, NN, Parents)
 local i, k, a, Children, val :

a := rand(2 ..NN[M + 1 ] − 1 ) :
 Children := Matrix(2, NN[M + 1 ], fill = 0 ) :
val := a( );
 Children[1, 1 ..val] := Parents[2, 1 ..val];
 Children[2, 1 ..val] := Parents[1, 1 ..val];


 Children[1, val + 1 ..NN[M + 1 ]] := Parents[1, val + 1 ..NN[M + 1 ]];
 Children[2, val + 1 ..NN[M + 1 ]] := Parents[2, val + 1 ..NN[M + 1 ]];

 return Children;
```

```
 end proc:


 Even_Breeding := proc(N, M, NN, Parents, p_bred)
 local i, k, a, Children, val :

a := rand(0.0 ..1.0 ) :
 Children := Matrix(2, NN[M + 1 ], fill = 0 ) :

for i from 1 to NN[M + 1 ] do
val := a( );

if val > p_bred then
 Children[1, i ] := Parents[1, i ];
 Children[2, i ] := Parents[2, i ];
 fi;
if val > p_bred then
 Children[1, i ] := Parents[2, i ];
 Children[2, i ] := Parents[1, i ];
 fi;
 end do;


 return Children;
 end proc:
```

Mutation

```
Mutation :=proc(vec, abc, N, M, NN )
     #abc −ймовірність мутації, vec − бінарна стрічка
 local i, new_vec, val, a;
 a := rand(0.0 ..1.0 );
 new_vec := Matrix(1, NN[M + 1 ], fill = 0 );
for i from 1 to NN[M + 1 ] do
 val := a( ) :

 if val ≤ abc  then vec[i ] := 1 + vec[i ] mod 2;

 end if;
 end do:
 return vec;
 end proc:
```

Best and Worst

```
Best := proc(Gdec, N, M)
local best;
best := max[index](Gdec[1..N, M+1]);
return best;
end proc:
Worst := proc(Gdec, N, M)
local worst;
worst := min[index](Gdec[1..N, M+1]);
return worst;
end proc:
```

Початкові дані

$$f1 := (x, y, z) \to -\mathrm{abs}\left(-\left((1-x)^2 + (1-y)^2 + 100\cdot(y-x^2)^2 + 100\cdot(z-y^2)^2\right)\right):$$

#Початкові дані - маємо тривимірну функцію Розенброка;

$$dims1 := 3:$$

$$X\_min1 := Array([-3,-3,-3]):$$

$$X\_max1 := Array([3, 3, 3]):$$

$$f2 := (x, y) \to -\mathrm{abs}\left(\left(\frac{3}{2} - x + x\cdot y\right)^2 + (2.25 - x + x\cdot y^2)^2 + (2.625 - x + x\cdot y^3)^2\right):$$

#Початкові дані - маємо тривимірну функцію Біла

$$dims2 := 2:$$

$$X\_min2 := Array([-4.5,-4.5]):$$

$$X\_max2 := Array([4.5, 4.5]):$$

$$f3 := (x, y) \to evalf\left(-\mathrm{abs}\left(-20\mathrm{e}^{-0.2\cdot(0.5\cdot(x^2+y^2))^{\frac{1}{2}}} - \mathrm{e}^{0.5\cdot(\cos(2\cdot\pi\cdot x)+\cos(2\cdot\pi\cdot y))} + \mathrm{e} + 20\right)\right):$$

#Початкові дані - маємо двовимірну функцію Еклі

$$dims3 := 2:$$

$$X\_min3 := Array([-5,-5]):$$

$$X\_max3 := Array([5, 5]):$$

$$Data := 2$$

 # Вибір номеру початкових даних 1- перша функція і відповідні їй дані, 2 -друга, і так далі

$$Data := 2 \quad (1)$$

```
if Data = 1 then
 f := f1 :                                   #Вибір початкових даних
 M := dims1 :
 xmin := X_min1 :
 xmax := X_max1 :
 end if:
if Data = 2 then
 f := f2 :
 M := dims2 :
 xmin := X_min2 :
 xmax := X_max2 :
 end if:
if Data = 3 then
 f := f3 :
 M := dims3 :
 xmin := X_min3 :
 xmax := X_max3 :
 end if:
 N := 50;                          #Чисельність популяції
```

$$N := 50 \quad (2)$$

```
 Iter := 300;                          #Кількість ітерацій
```

$$Iter := 300 \quad (3)$$

```
 eps := 0.001 :                        #Точність бінарного кодування
 Sel_method := 3 :        #Selection method, 1-Proportion, 2-Ranking, 3-Panmixion
 Norm_method := 2 :
      #Метод отримання фітнес функції, 1- як у текстовому файлі, 2 - як у моїй презентації
 Breeding_method := 1 :    #Метод схрещування, 1- одноточкове, 2- рівномірне
 p_breed := 1/2 :    #Ймовірність в рівномірному схрещуванні
 Elitism := 0 :          #Чи застосовувати метод елітизму? 1-так 0-ні
 Gray_code := 1 :              #Чи використовувати стратегію кодування Грея? 1 - так, 0 - ні.
 NN, nn, dd := BinDecParam(xmin, xmax, eps, M)
```

$$NN, nn, dd := NN, [14, 14], [0.0005493164062, 0.0005493164062] \quad (4)$$

```
 p_mut := 5/(NN[M+1])    #ймовірність застогсування мутації до кожного окремого біту
```

$$p\_mut := \frac{5}{28} \quad (5)$$

Алгоритм

$G := GenerationDec(xmin, xmax, N, M)$ *#Генеруємо матрицю*

$$G := \begin{bmatrix} -3.640438440 & -0.450144900 & 0 \\ 4.200629130 & -3.732003720 & 0 \\ -4.454475930 & -1.745521470 & 0 \\ -3.617109990 & -1.655303580 & 0 \\ -3.034831410 & 1.673387010 & 0 \\ -4.159074780 & 1.542317670 & 0 \\ 3.233581110 & -3.973106790 & 0 \\ -1.759782150 & -3.787679250 & 0 \\ 0.277602390 & 3.102201540 & 0 \\ 1.727949780 & 1.926595440 & 0 \\ \vdots & \vdots & \vdots \end{bmatrix} \quad \textbf{(6)}$$

50 × 3 Matrix

```
t1 := time( ) :
 t := 1 : num_of_species := 1 :
if Norm_method = 1  then
  fitness := Normalization_like_in_text(f, G, N, M) :
      #Додавання значень фітнес функції до матриці
 G[1 ..N, M + 1 ] := fitness[1 ..N ] :
 end if:
 if Norm_method = 2 then
 fitness := Normalization(f, G, N, M) :
 G[1 ..N, M + 1 ] := fitness[1 ..N ] :
 end if:

if Elitism = 1 then
Elite := Matrix(1, M + 1, fill = 0 ) :
 BestInPop := Best(G, N, M) :
Elite[1, 1 ..M + 1 ] := G[BestInPop, 1 ..M + 1 ] :
end if:

while t ≤ Iter do
 while num_of_species ≤ N do
if Sel_method = 1  then
  Probs := Proportion(G, N, M) :
 end if:
 if Sel_method = 2 then
G := G[ sort(G[ .., M + 1 ], 'output'='permutation') ] :      #Сортування
 Probs := Ranking(N, M) :
 end if:
```

```
if Sel_method = 3 then

Probs := Panmixion(N, M) :
end if:
if Gray_code = 1 then
ao := ACodBinary1(xmin, G, N, M, NN, nn, dd);
Gbin := BiGr(ao, N, M, NN);
end if:
if Gray_code = 0 then
ao := ACodBinary1(xmin, G, N, M, NN, nn, dd);
Gbin := ao;
end if:



Shkala, Parents := Selection(Gbin, Probs, N, M, NN) :
Childs := Matrix(2, NN[M + 1], fill = 0) :

if Breeding_method = 1 then
Childs := OnePointBreeding(N, M, NN, Parents) :
end if:
 if Breeding_method = 2 then
Childs := Even_Breeding(N, M, NN, Parents, p_breed) :
end if:

Childs[1] := Mutation(Childs[1, 1 ..NN[M + 1]], p_mut, N, M, NN) :
Childs[2] := Mutation(Childs[2, 1 ..NN[M + 1]], p_mut, N, M, NN) :
a := rand(0 ..10) : val := a( ) :
if val ≥ 5 then
nu := Worst(G, N, M);
Gbin[nu, 1 ..NN[M + 1]] := Childs[1, 1 ..NN[M + 1]];
fi:
if val < 5 then
nu := Worst(G, N, M);
Gbin[nu, 1 ..NN[M + 1]] := Childs[2, 1 ..NN[M + 1]];
fi:
if Gray_code = 1 then
ap := GrBi(Gbin, N, M, NN);
G := ACodDecimal1(xmin, ap, N, M, NN, nn, dd);
 end if:
 if Gray_code = 0 then

G := ACodDecimal1(xmin, Gbin, N, M, NN, nn, dd);
 end if:

num_of_species := num_of_species + 1 :
if Norm_method = 1 then
 fitness := Normalization_like_in_text(f, G, N, M) :
    #Додавання значень фітнес функції до матриці
```

$G[1..N, M+1] := fitness[1..N]:$
end if:
if $Norm\_method = 2$ **then**
$fitness := Normalization(f, G, N, M):$
$G[1..N, M+1] := fitness[1..N]:$
end if:

end do:

if $Elitism = 1$ **then**
$nuli := Worst(G, N, M);$
$G[nuli, 1..M+1] := Elite[1, 1..M+1];$
end if:

$num\_of\_species := 1:$
$t := t + 1:$

end do: $t2 := time() - t1;$

$$t2 := 455.250 \tag{7}$$

G

$$\begin{bmatrix} 2.758666992 & 0.424621582 & 1620.197436 \\ 2.998718261 & 0.519653320 & 1620.201699 \\ 3.067932128 & 0.498229980 & 1620.202431 \\ 2.798217773 & 0.427917480 & 1620.197627 \\ 2.844909667 & 0.449340820 & 1620.204881 \\ 2.963012695 & 0.510314941 & 1620.202241 \\ 2.977844238 & 0.514160156 & 1620.202137 \\ 2.977844238 & 0.515808105 & 1620.200544 \\ 2.978393554 & 0.481750488 & 1620.207500 \\ 2.794921874 & 0.433410644 & 1620.200492 \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{8}$$

50 × 3 Matrix

$Best(G, N, M)$

(9)

$f(seq(G[Best(G, N, M), k], k = 1..M))$

$$-0.0001402957749 \quad \textbf{(10)}$$

$G[Best(G, N, M), 1..M]$

$$\begin{bmatrix} 2.977294921 & 0.492736816 \end{bmatrix} \quad \textbf{(11)}$$

Час на виконання t2 = 455.250 *Час на виконання*