

Métodos de Inteligencia Artificial

Reporte 3

IF698972

Josefina Esmeralda Arriaga Hernández

30 de enero del 2017 Guadalajara, Jalisco

**Objetivo**

Crear un programa de un algoritmo genético con el propósito de aproximarse a las ecuaciones establecidas.

**Problema a resolver**

Se tiene las ecuaciones y = x2sen2(10πx)+1 dentro del intervalo [0, 2] que se va a maximizar, se minimizará y= 10+ x2- 10cos(2πx) dentro del intervalo -5.12 ≤ x ≤ 5.12 y se maximizará la función y = f(x1, x2) = (1.5-x1+x1x2)2+(2.25-x1-x1x22)2+(2.625-x1-x1x23)2 dentro del intervalo -4.5 ≤ x ≤ 4.5, se va a desarrollar un algoritmo genético a partir de estos.

Los pasos para crear el algoritmo son:

1. Generar la población.
2. Se evalúa los pobladores.
3. Se selecciona los progenitores.
4. Se realiza el cruzamiento para obtener hijos.
5. Se muta a hijos aleatorios.
6. Se crea una iteración para llegar al resultado deseado.
7. Por último, se promedia el eje y.

**Código desarrollado**

**Primera función**

clear all;

close all;

clc

np = 8; %N?mero de pobladores

tpaso = 0.01; %Tama?o de paso

xmin = 0;

xmax = 2;

elementos = (xmax - xmin)/ tpaso; %k

nbits = ceil(log2(elementos)); %ceil redondea hacia arriba los decimales

%Generamos la poblaci?n inicial

x1 = randi([0,2^nbits-1],np,1); %Genera np n?meros aleatorios enteros en una columna

x1real = ((xmax - xmin)/(2^nbits-1))\*x1 + xmin; %Convierte enteros a decimales

%(funci?n objetivo)

for i = 1:100 %N?mero de generaciones

    y = x1real.\*(sin(10\*pi\*x1real)).^2+1;

    yprom(i) = mean(y); %Media del vector y

    cromosoma = [y x1 x1real];

    cromosomaOrd = sortrows(cromosoma,1); %Ordena en funci?n de la columna 1

    %Selecci?n

    padres = cromosomaOrd(np/2+1:np,2);

    padresbin = dec2bin(padres,nbits);

    %Cruzamiento

    %Cruce aritm?tico

 for k = 1:(np/4)

        padres1(1) = padres(1)+padres(2);

        padres1(2) = padres(3)+padres(4);

        padres1(3) = padres(1)+padres(4);

        padres1(4) = padres(2)+padres(3);

        %Hijo1

        hijo(2\*k-1,:) = xor(padresbin(2\*k-1),padresbin(2\*k));

        %Hijo2

        hijo(2\*k,:) = xor(padresbin(2\*k),padresbin(2\*k-1));

 end

    hijobin=dec2bin(hijo,nbits);

    %Mutaci?n

    m = rand();

    if m >= 0.8

        nhijo = randi(np/4);

        bit = randi([1,nbits]);

        if hijobin(nhijo,bit) == '1'

            hijobin(nhijo,bit) = '0';

        else

            hijobin(nhijo,bit) = '1';

        end

    end

    %Convertir a n?meros reales en base a la funci?n objetivo

    hijodec = bin2dec(hijobin);

    hijoreal = ((xmax - xmin)/(2^nbits-1))\*hijodec + xmin;

    %Juntamos padres e hijos en el vector de x1

    %Aqu? se eliminan los m?s d?biles

    x1real = [cromosomaOrd(np/2+1:np,3);hijoreal];

    x1 = [cromosomaOrd(np/2+1:np,2);hijodec];

end

plot(yprom)

max(y)

**Segunda función**

clear all;

close all;

clc

np = 32; %N?mero de pobladores

tpaso = 0.01; %Tama?o de paso

xmin = -5.12;

xmax = 5.12;

elementos = (xmax - xmin)/ tpaso; %k

nbits = ceil(log2(elementos)); %ceil redondea hacia arriba los decimales

%Generamos la poblaci?n inicial

x1 = randi([0,2^nbits-1],np,1); %Genera np n?meros aleatorios enteros en una columna

x1real = ((xmax - xmin)/(2^nbits-1))\*x1 + xmin; %Convierte enteros a decimales

%(funci?n objetivo)

for i = 1:1000 %N?mero de generaciones

    y =  -(10+x1real.^2-10\*(cos(2\*pi\*x1real)));

    yprom(i) = mean(y); %Media del vector y

    cromosoma = [y x1 x1real];

    cromosomaOrd = sortrows(cromosoma,1); %Ordena en funci?n de la columna 1

    %Selecci?n

    padres = cromosomaOrd(np/2+1:np,2);

    padresbin = dec2bin(padres,nbits);

    %Cruzamiento

    %Cruce aritm?tico

 for k = 1:(np/4)

        padres1(1) = padres(1)+padres(2);

        padres1(2) = padres(3)+padres(4);

        padres1(3) = padres(1)+padres(4);

        padres1(4) = padres(2)+padres(3);

        %Hijo1

        hijo(2\*k-1,:) = xor(padresbin(2\*k-1),padresbin(2\*k));

        %Hijo2

        hijo(2\*k,:) = xor(padresbin(2\*k),padresbin(2\*k-1));

 end

    hijobin=dec2bin(hijo,nbits);

    %Mutaci?n

    m = rand();

    if m >= 0.8

        nhijo = randi(np/4);

        bit = randi([1,nbits]);

        if hijobin(nhijo,bit) == '1'

            hijobin(nhijo,bit) = '0';

        else

            hijobin(nhijo,bit) = '1';

        end

    end

    %Convertir a n?meros reales en base a la funci?n objetivo

    hijodec = bin2dec(hijobin);

    hijoreal = ((xmax - xmin)/(2^nbits-1))\*hijodec + xmin;

    %Juntamos padres e hijos en el vector de x1

    %Aqu? se eliminan los m?s d?biles

    x1real = [cromosomaOrd(np/2+1:np,3);hijoreal];

    x1 = [cromosomaOrd(np/2+1:np,2);hijodec];

end

plot(yprom)

max(y)

**Tercera función**

clear all;

close all;

clc

np = 32; %N?mero de pobladores

tpaso = .01; %Tama?o de paso

xmin = -4.5;

xmax = 4.5;

elementos = (xmax - xmin)/ tpaso; %k

nbits = ceil(log2(elementos)); %ceil redondea hacia arriba los decimales

%Para segunda x

tpaso2 = .01; %Tama?o de paso

xmin2 = -4.5;

xmax2 = 4.5;

elementos2 = (xmax2 - xmin2)/ tpaso2; %k

nbits2 = ceil(log2(elementos2)); %ceil redondea hacia arriba los decimales

%Generamos la poblaci?n inicial

x1 = randi([0,2^nbits-1],np,1); %Genera np n?meros aleatorios enteros en una columna

x1real = ((xmax - xmin)/(2^nbits-1))\*x1 + xmin; %Convierte enteros a decimales

%Generamos la poblaci?n inicial

x2 = randi([0,2^nbits2-1],np,1); %Genera np n?meros aleatorios enteros en una columna

x2real = ((xmax2 - xmin2)/(2^nbits2-1))\*x2 + xmin2; %Convierte enteros a decimales

%(funci?n objetivo)

for i = 1:100 %N?mero de generaciones

    %y = -((x1real+2\*x2real-7).^2+(2\*x1real+x2real-5).^2);

    y = -((1.5-x1real+x1real.\*x2real).^2+(2.25-x1real-x1real.\*(x2real).^2).^2+(2.625-x1real-x1real.\*(x2real).^3).^2);

    yprom(i) = mean(y); %Media del vector y

    cromosoma = [y x1 x1real x2 x2real];

    cromosomaOrd = sortrows(cromosoma,1); %Ordena en funci?n de la columna 1

    %Selecci?n

    padres = cromosomaOrd(np/2+1:np,2);

    padresbin = dec2bin(padres,nbits);

    padres2 = cromosomaOrd(np/2+1:np,4);

    padresbin2 = dec2bin(padres2,nbits2);

    %Cruzamiento

    %Un punto de cruce p

    for k = 1:(np/4)

        %Hijo1

        p = randi([2 nbits-1]); %Aleatorios enteros del 2 a nbits-1

        hijobin(2\*k-1,:) = [padresbin(2\*k-1,1:p) padresbin(2\*k,p+1:nbits)];

        %Hijo2

        p = randi([2 nbits-1]);

        hijobin(2\*k,:) = [padresbin(2\*k,1:p) padresbin(2\*k-1,p+1:nbits)];

    end

     for k = 1:(np/4)

        %Hijo1

        p = randi([2 nbits2-1]); %Aleatorios enteros del 2 a nbits-1

        hijobin2(2\*k-1,:) = [padresbin2(2\*k-1,1:p) padresbin2(2\*k,p+1:nbits2)];

        %Hijo2

        p = randi([2 nbits-1]);

        hijobin2(2\*k,:) = [padresbin2(2\*k,1:p) padresbin2(2\*k-1,p+1:nbits2)];

    end

    %Mutaci?n

    m = rand();

    if m >= 0.8

        nhijo = randi(np/4);

        bit = randi([1,nbits]);

        if hijobin(nhijo,bit) == '1'

            hijobin(nhijo,bit) = '0';

        else

            hijobin(nhijo,bit) = '1';

        end

    end

      m = rand();

    if m >= 0.8

        nhijo = randi(np/4);

        bit = randi([1,nbits2]);

        if hijobin2(nhijo,bit) == '1'

            hijobin2(nhijo,bit) = '0';

        else

            hijobin2(nhijo,bit) = '1';

        end

    end

    %Convertir a n?meros reales en base a la funci?n objetivo

    hijodec = bin2dec(hijobin);

    hijoreal = ((xmax - xmin)/(2^nbits-1))\*hijodec + xmin;

    %Convertir a n?meros reales en base a la funci?n objetivo

    hijodec2 = bin2dec(hijobin2);

    hijoreal2 = ((xmax2 - xmin2)/(2^nbits2-1))\*hijodec2 + xmin2;

    %Juntamos padres e hijos en el vector de x1

    %Aqu? se eliminan los m?s d?biles

    x1real = [cromosomaOrd(np/2+1:np,3);hijoreal];

    x1 = [cromosomaOrd(np/2+1:np,2);hijodec];

    x2real = [cromosomaOrd(np/2+1:np,5);hijoreal2];

    x2 = [cromosomaOrd(np/2+1:np,4);hijodec2];

end

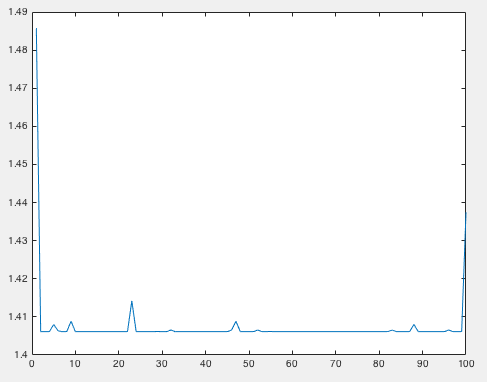
plot(yprom)

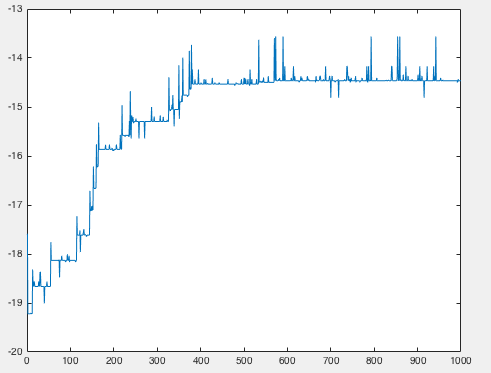
[val,ind] = max(y)

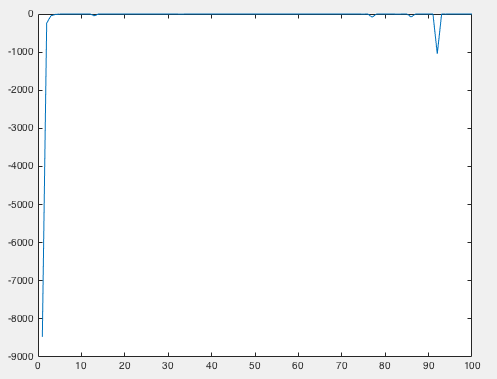
disp(['Resultado: x1= ' num2str(cromosoma(ind,3))...

    ' x2=' num2str(cromosoma(ind,5)) ', Desempe?o = ' num2str(val)])

**Gráficos**

**Primera función**

**Segunda función**

**Tercera función**

**Interpretación de gráficos**

En las primeras iteraciones se llega al resultado deseado y con ayuda de las mutaciones que se realizan de manera aleatoria se observa que se llega a 10 varias veces, entonces el promedio del eje y da como resultado el máximo o mínimo de la función dependiendo de que función se está hablando. Si no se tuviera la mutación podría salir super individuos y los hijos saldrían igual haciendo que se estanque el promedio.

**Resultados**

El resultado es entre 2-3, siendo este el promedio del eje Y de la función y = x2sen2(10πx)+1, se logra obtener ese resultado ya que se agregó la mutación (cambiando el número binario de un poblador al azar) haciendo que el algoritmo no tenga super individuos. Son 8 pobladores que van generando generaciones 1000 veces.

El de la segunda función es y= 10+ x2- 10cos(2πx) en un intervalo de -5.12 ≤ x ≤ 5.12 teniendo como resultado 0 ya que éste es su mínimo. Son 32 pobladores que van generando generaciones 1000 veces. Todos los algoritmos genéticos se realizaron con el cruzamiento de un punto.

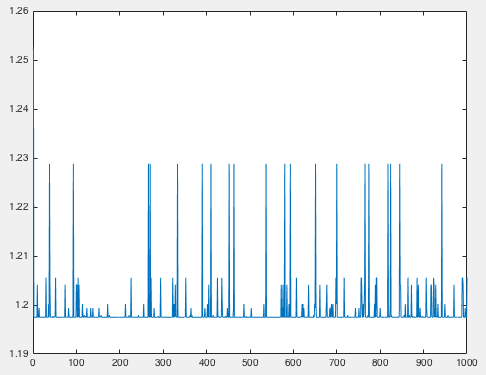
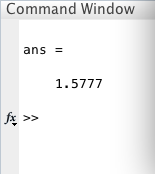
Y para terminar la tercera función y = f(x1, x2) = (1.5-x1+x1x2)2+(2.25-x1-x1x22)2+(2.625-x1-x1x23)2 dentro del intervalo -4.5 ≤ x ≤ 4.5 tiene como resultado en x1 de 3 y x2 de 0.5. En el algoritmo genético se dice también en cual iteración se llegó al resultado esperado al igual que los resultados de ambas incógnitas.

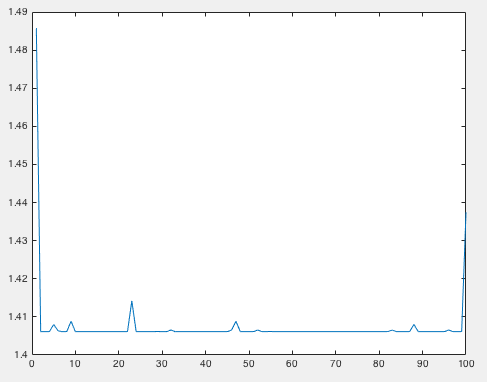
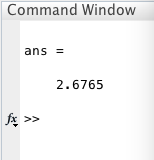
**Conclusiones**

En conclusión, se puede decir que el programa es exitoso ya que se llega al resultado esperado sin crear super individuos, en pocas iteraciones. El algoritmo genético es heurístico siendo esto que se obtuvo una buena solución, para las distintas funciones se obtuvo el resultado deseado y se aprendió a crear algoritmos genéticos con dos variables.

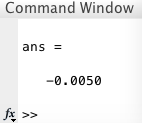
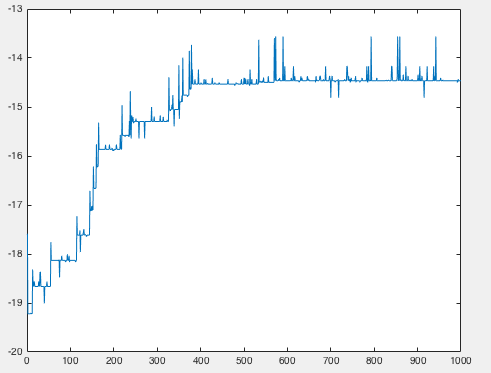
**Funcionamiento**

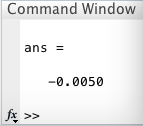
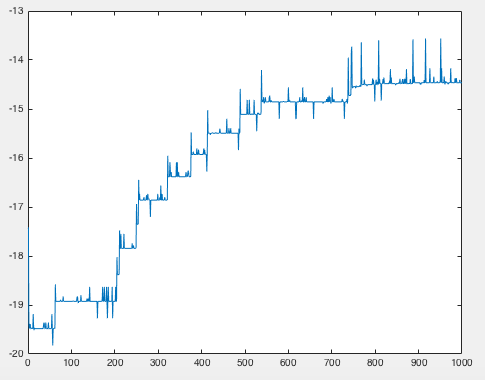
**Primera función**





**Segunda función**

****

****

**Tercera función**

