

Métodos de Inteligencia Artificial

Reporte 2

IF698972

Josefina Esmeralda Arriaga Hernández

26 de enero del 2017 Guadalajara, Jalisco

**Objetivo**

Crear un programa de un algoritmo genético con el propósito de aproximarse a la ecuación establecida usando los diferentes tipos de cruzamiento

**Problema a resolver**

Se tiene la ecuación y= –(x+1.25)2+15 y se va a desarrollar un algoritmo genético a partir de este.

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. Cruzamiento con dos puntos de cruce
6. Cruzamiento uniforme
7. Cruzamiento aritmético
8. Se muta a hijos aleatorios.
9. Se crea una iteración para llegar al resultado deseado.
10. Por último, se promedia el eje y.

**Código desarrollado**

**Cruzamiento con dos puntos de cruce**

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-1.25).^2 + 15;

    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

  %  Dos puntos de cruce p

    for k = 1:(np/4)

        %Hijo1

        unop = randi([2 nbits-4]); %N?meros aleatorios enteros del 2 al 5

        dosp = randi([unop nbits-1]);

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

        %Hijo2

        unop = randi([2 nbits-4]);

        dosp = randi([unop nbits-1]);

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

    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

    %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)

**Cruzamiento uniforme**

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-1.25).^2 + 15;

    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 Uniforme (solo para 8 bits)

    for k = 1:(np/4)

        aleatorio = rand();

        if aleatorio <= 0.5

            %Hijo1

            hijobin(2\*k-1,:) = [padresbin(2\*k-1,1:2) padresbin(2\*k,2:3) padresbin(2\*k-1,3:4) padresbin(2\*k,4:5) padresbin(2\*k-1,5:6) padresbin(2\*k,6:7) padresbin(2\*k-1,7:8)];

            %Hijo2

            hijobin(2\*k,:) = [padresbin(2\*k,1:2) padresbin(2\*k-1,2:3) padresbin(2\*k,3:4) padresbin(2\*k-1,4:5) padresbin(2\*k,5:6) padresbin(2\*k-1,6:7) padresbin(2\*k,7:8)];

        else

            %Hijo1

            hijobin(2\*k-1,:) = [padresbin(2\*k,1:2) padresbin(2\*k-1,2:3) padresbin(2\*k,3:4) padresbin(2\*k-1,4:5) padresbin(2\*k,5:6) padresbin(2\*k-1,6:7) padresbin(2\*k,7:8)];

            %Hijo2

            hijobin(2\*k,:) = [padresbin(2\*k-1,1:2) padresbin(2\*k,2:3) padresbin(2\*k-1,3:4) padresbin(2\*k,4:5) padresbin(2\*k-1,5:6) padresbin(2\*k,6:7) padresbin(2\*k-1,7:8)];

        end

    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

    %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)

**Cruzamiento aritmético**

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-1.25).^2 + 15;

    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];

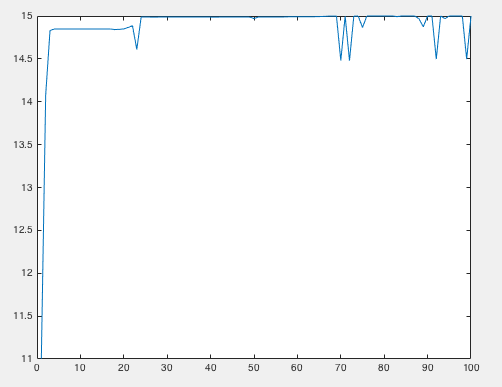
end

plot(yprom)

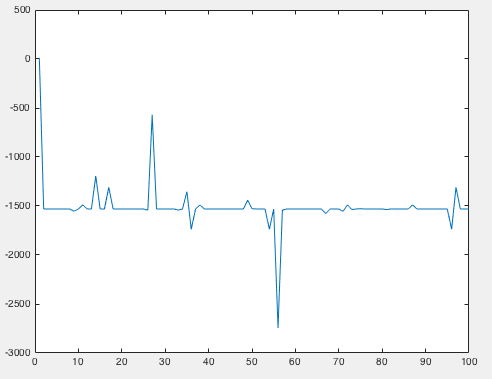
max(y)

**Gráficos**

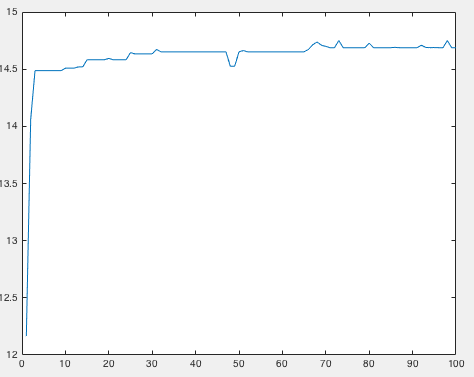
**Cruzamiento con dos puntos de cruce**

****

**Cruzamiento uniforme**

****

**Cruzamiento aritmético**

****

**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 15 varias veces, siendo éste su máximo. Si no se tuviera la mutación podría salir super individuos y los hijos saldrían igual haciendo que se estanque el promedio. Esto sucede con los tres tipos de cruzamiento siendo el primero de dos puntos de cruce, cruzamiento uniforme y cruzamiento aritmético.

**Resultados**

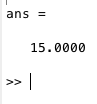
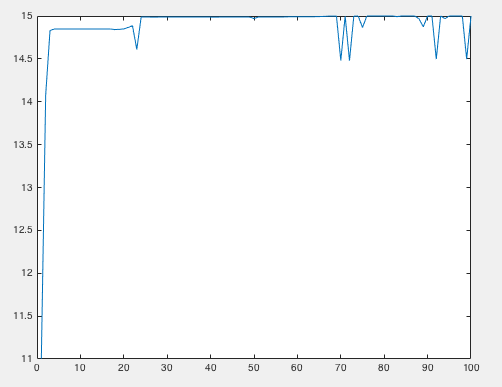
El resultado es 15, siendo este el promedio del eje Y de la función y= –(x+1.25)2+15 y el máximo de la función, se logra obtener ese resultado ya que se agregó la mutación (cambiando el número binario de un poblador al azar) haciendo que el algoritmo no tenga super individuos. Son 8 pobladores que van generando generaciones 1000 veces a un paso de 0.01.

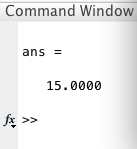
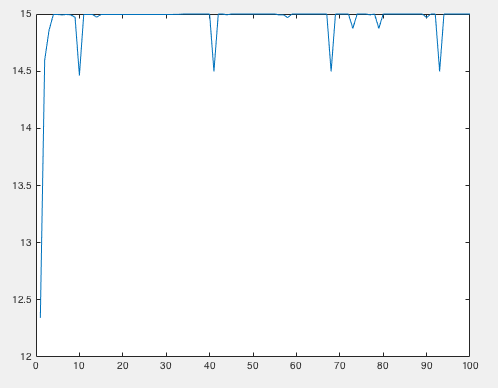
**Conclusiones**

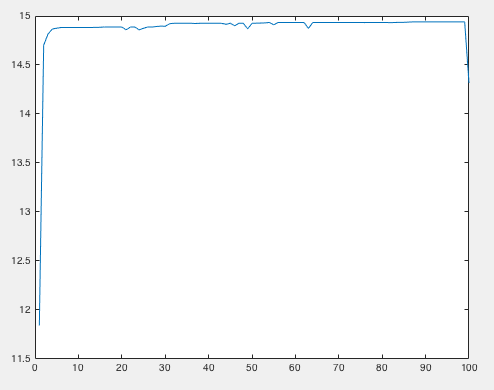
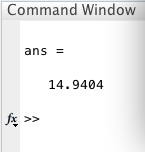
En conclusión, se puede decir que el programa es exitoso en los tres tipos de cruzamiento ya que se llega al resultado esperado sin crear super individuos, en pocas iteraciones. El algoritmo genético es heurístico siendo esto que se obtuvo una buena solución, facilitando llegar al punto máximo 15.

**Funcionamiento**

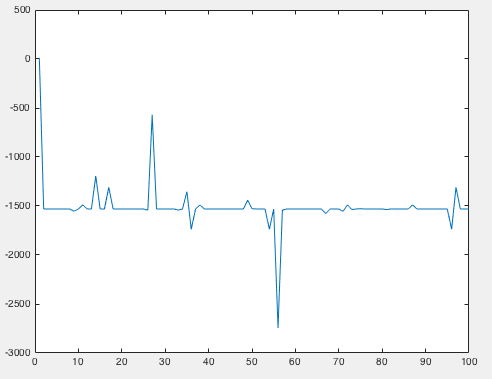
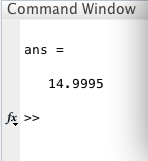
**Cruzamiento con dos puntos de cruce**

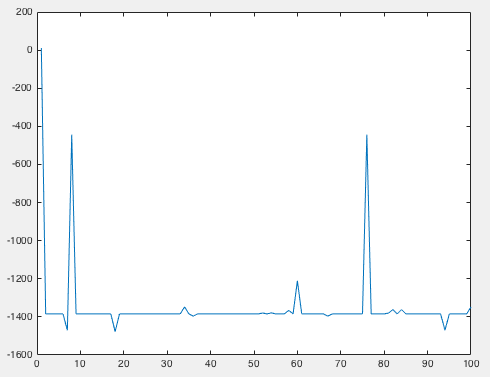
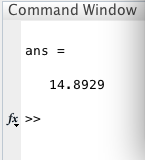


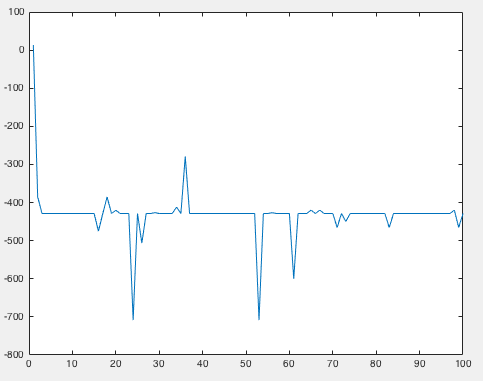
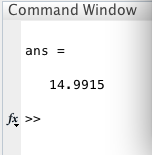




**Cruzamiento uniforme**







**Cruzamiento aritmético**

