Reporte “Algoritmo Genético Binario”

Alma Eliza Guerrero Sánchez.

1. Introducción

En la inteligencia Artificial siempre ha sido caracterizada por la resolución de problemas, usando el comportamiento de sistemas neurológicos, la naturaleza, los genes. Esto para tener encontrar la manera mas optima para la solución de alguna situación. Esta técnica se basa en los mecanismos de selección que utiliza la naturaleza, de acuerdo a los cuales los individuos más aptos de una población son los que sobreviven, al adaptarse más fácilmente a los cambios que se producen en su entorno. Hoy en día se sabe que estos cambios se efectúan en los genes de un individuo (unidad básica de codificación de cada uno de los atributos de un ser vivo), y que sus atributos más deseables (i.e., los que le permiten adaptarse mejor a su entorno) se transmiten a sus descendientes cuando éste se reproduce sexualmente.

Un investigador de la Universidad de Michigan llamado John Holland era consciente de la importancia de la selección natural, y a fines de los 60s desarrolló una técnica que permitió incorporarla a un programa. Su objetivo era lograr que las computadoras aprendieran por sí mismas. A la técnica que inventó Holland se le llamó originalmente "planes reproductivos", pero se hizo popular bajo el nombre "algoritmo genético" tras la publicación de su libro en 1975.

Una definición bastante completa de un algoritmo genético es la propuesta por John Koza:

"Es un algoritmo matemático altamente paralelo que transforma un conjunto de objetos matemáticos individuales con respecto al tiempo usando operaciones modeladas de acuerdo al principio Darwiniano de reproducción y supervivencia del más apto, y tras haberse presentado de forma natural una serie de operaciones genéticas de entre las que destaca la recombinación sexual. Cada uno de estos objetos matemáticos suele ser una cadena de caracteres (letras o números) de longitud fija que se ajusta al modelo de las cadenas de cromosomas, y se les asocia con una cierta función matemática que refleja su aptitud. "

2. Marco teórico

Los AG simulan la supervivencia de los más aptos entre individuos durante generaciones consecutivas. Cada generación consiste en una población de cadenas de caracteres que son análogas al cromosoma que vemos en nuestro ADN. Cada individuo representa un punto en un espacio de búsqueda y una posible solución. A los individuos de la población se les hace pasar por un proceso de evolución.

El algoritmo genético se puede aplicar para solucionar problemas que no se adaptan bien a los algoritmos de optimización estándar, incluidos aquellos problemas en los que la función objetivo es discontinua, no diferenciable, estocástica o altamente no lineal.

2.2 Población Inicial

El proceso comienza con un conjunto de individuos que se llama Población. Cada individuo es una solución al problema que quiere resolver.

Un individuo se caracteriza por un conjunto de parámetros (variables) conocidos como genes. Los genes se unen en una cadena para formar un cromosoma (solución).

En un algoritmo genético, el conjunto de genes de un individuo se representa mediante una cadena, en términos de un alfabeto. Usualmente, se usan valores binarios (cadena de 1s y 0s). Decimos que codificamos los genes en un cromosoma.

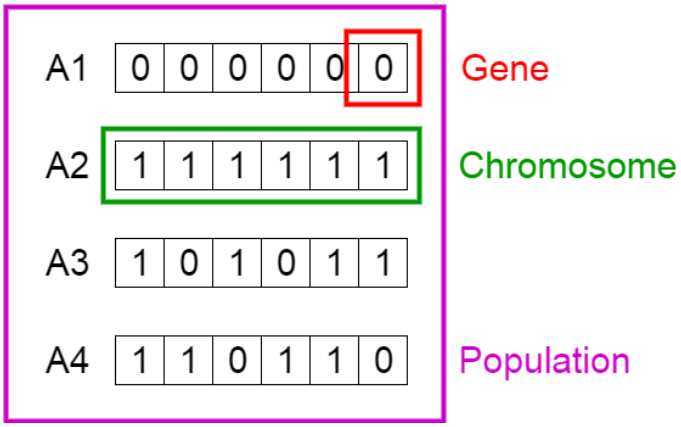

Figura 1. Población Inicial

2.3 Espacio de Búsqueda

Cuando se resuelve un problema, se busca la mejor solución entre un conjunto de posibles soluciones. Al conjunto de todas las posibles soluciones a un problema concreto se llama espacio de búsqueda. Cada punto en el espacio de búsqueda representa una posible solución. Cada posible solución se le puede asociar un fitness o un valor que indicará cómo de buena es la solución para el problema. Un algoritmo genético (AG) devolverá la mejor solución de entre todas las posibles que tenga en un momento dado.

Entonces parece que buscar una solución se reduce a buscar un valor extremo (mínimo o máximo) en el espacio de búsqueda. A veces el espacio de búsqueda puede ser bien definido, pero en la mayoría de las ocasiones sólo se conocen algunos puntos en el espacio de búsqueda. Cuando se usa un AG las posibles soluciones generan otras a medida que el genético evoluciona.

2.4 Función Fitness

La función de aptitud o función Fitness determina qué tan adecuada es una persona (la capacidad de una persona para competir con otras personas). Da una puntuación de aptitud física a cada individuo. La probabilidad de que un individuo sea seleccionado para la reproducción se basa en su puntaje de condición física.

2.5 Métodos de Selección

La idea de la fase de selección es seleccionar a los individuos más aptos y dejar que pasen sus genes a la siguiente generación.

Se seleccionan dos pares de individuos (padres) en función de sus calificaciones de aptitud física. Las personas con un buen estado físico tienen más posibilidades de ser seleccionadas para su reproducción.

2.6 Cruza

El cruce es la fase más significativa en un algoritmo genético. Para cada par de padres que se deben aparear, se elige un punto de cruce al azar dentro de los genes.

Por ejemplo, considere que el punto de cruce es 3 como se muestra a continuación.

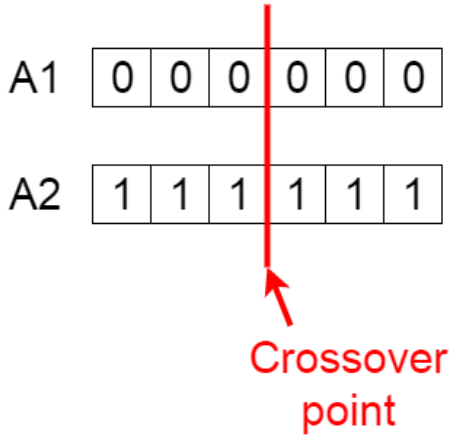

Figura 2. Cortes

Los descendientes se crean intercambiando los genes de los padres entre sí hasta que se alcanza el punto de cruce.

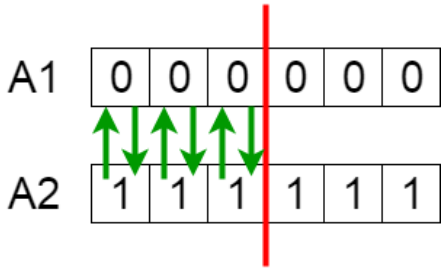

Figura 3.Cruza

Los nuevos descendientes se suman a la población.

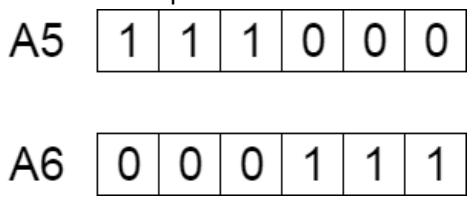

Figura 4. Descendientes

2.7 Etilismo

Este concepto expresa la idea de que el mejor individuo de la actual generación pase sin modificar a la siguiente generación. De esta forma no se perderá el mejor cromosoma. Al resto de la población se le aplica la reproducción normalmente.

Por otra parte existen algoritmos genéticos llamados elitistas debido a que convergen muy rápidamente a la solución.

3. Metodología

```
w; %%numero de generaciones que correrá
k; %%contador de generaciones
n; %%numero de población
A=round(rand(n,6)) %genero  población inicial
A1=bi2de(A) %%convierto de decimal
Normalizado; Normalizo la población en un rango de [-1,1]
Mientras k>contador de generaciones
 {

    Mientras i>size(A){
Aquí realizo la selección y la cruza a dos cortes
      Pareja1(pares)
      Pareja2(impares)
     Corte1=pareja1(1:2)
     Corte2= pareja2(3:4)
     Corte3= pareja1(5:6)
     Hijo=corte1+corte2+corte3;
    HijosSeccion1=hijos;
  }

    Mientras i>size(A){
Aquí realizo la selección y la cruza a dos cortes
      Pareja1(pares)
      Pareja2(impares)
     Corte1=pareja2(1:2)
     Corte2= pareja1(3:4)
     Corte3= pareja2(5:6)
     Hijo=corte1+corte2+corte3;
    HijosSeccion2=hijos;
  }
A=HijosSeccion1+HijosSeccion2
Grafico la nueva generación
 }
```

Cuadro 1.Seudocódigo Sin Etilismo selección aleatoria

```
w; %%numero de generaciones que correrá
k; %%contador de generaciones
n; %%numero de población
A=round(rand(n,6)) %genero  población inicial
A1=bi2de(A) %%convierto de decimal
```

```
Normalizado; Normalizo la población en un rango de [-1,1]
Next_gener=[]
Mientras k>contador de generaciones
 {

    Mientras i>size(A){
Aquí realizo la selección y la cruza a dos cortes
//Aquí realizo la selección y la cruza a dos cortes
      Para la selección de parejas
   Genero un vector con números aleatorios
     Corte1=pareja1(1:3)
     Corte2= pareja2(4:6)
     Hijo=corte1+corte2
    Corte1=pareja2(1:3)
     Corte2= pareja1(4:6)
     Hijo2=corte1+corte2
      Normalizo las parejas y los hijos resultantes
     nPareja1= normalizo (pareja1)
     nPareja2= normalizo (pareja2)
     nhijo=normalizo(Hijo)
    nhijo2= normalizo(Hijo2)
    ordeno de mayor a menor las 4 opciones
    tomo los 2 mas altos
    y se depositan en la siguiente generación
     finales
   next_gener=(finales)
  }
 }
```

Cuadro 2.Seudocódigo Con Etilismo selección Pseudoaleatoria

```
w; %%numero de generaciones que correrá
k; %%contador de generaciones
n; %%numero de población
A=round(rand(n,6)) %genero  población inicial
A1=bi2de(A) %%convierto de decimal
Normalizado; Normalizo la población en un rango de [-1,1]
Mientras k>contador de generaciones
 {

    Mientras i>size(A){
Aquí realizo la selección y la cruza a dos cortes
      Pareja1(pares)
      Pareja2(impares)
     Corte1=pareja1(1:2)
     Corte2= pareja2(3:4)
     Corte3= pareja1(5:6)
     Hijo=corte1+corte2+corte3;
    HijosSeccion1=hijos;
```

```
Aquí realizo la selección y la cruza a dos cortes
      Corte1=pareja2(1:2)
      Corte2= pareja1(3:4)
      Corte3= pareja2(5:6)
      Hijo=corte1+corte2+corte3;
     HijosSeccion2=hijos;
     Total=(hijoSeccion2+hijoSeccion1+pareja1+Pareja2)
   Convierto a decimal y después normalizo
  Comparo quien tiene mayor valor fitness y se guardan en mi arreglo de generacion
   Generacion=(ganadores)
   }
Grafico la nueva generación
 }
```

Cuadro 3.Seudocódigo Con Etilismo selección Aleatoria

```
w; %%numero de generaciones que correrá
k; %%contador de generaciones
n; %%numero de población
A=round(rand(n,6)) %genero  población inicial
A1=bi2de(A) %%convierto de decimal
Normalizado; Normalizo la población en un rango de [-1,1]
Next_gener=[]
Mientras k>contador de generaciones
 {

    Mientras i>size(A){
Aquí realizo la selección y la cruza a dos cortes
//Aquí realizo la selección y la cruza a dos cortes
      Para la selección de parejas
   Genero un vector con números aleatorios
     Corte1=pareja1(1:3)
     Corte2= pareja2(4:6)
     Hijo=corte1+corte2
    Corte1=pareja2(1:3)
     Corte2= pareja1(4:6)
     Hijo2=corte1+corte2
     HijosFinales=Hijo+Hijo2
    next_gener=(HijosFinales)
  }
 }
```

Cuadro 4.Seudocódigo Sin Etilismo selección Pseudoaleatoria

4. Código Documentado

```
c=1  %%variable para contar numero de generaciones
n=100 %%numero de poblacion
A=round(rand(n,6)) %%genero poblacion
A1=bi2de(A)%%convierto de binario a decimal
normalizado=funciones(A1) %%normalizo la poblacion en el intervalo de -
1 a 1 mando llamar la funcion de normalizacion
%%%%%%%%%%%%%%%%%%%%%%%%
x = linspace(-1,1,100);
y =(x.^2)
plot(x,y) %%Ploteo la funcion fitness
grid on
hold on
x1=normalizado
y1=normalizado.^2
plot(x1,y1,'+r')%%ploteo mi poblacion para ver su nivel fitness de
manera grafica
hold on
do = true;
while do
c=c+1
 if(c==200)
 do=false
 A1=bi2de(A)
%%%%%%%%%55
normalizado=funciones(A1)
%%%%%%%%%%%%%%seleccion aleatoria y cruza con dos cortes%%%%%%%%%%%%%
par=seleccion(A)%%Poblacion seleccionada
impar=Iseleccion(A)%%Poblacion seleccionada
hijos=[] %%hijos que tendre
for i=1:size(par)%%For para que pase por todo mi arreglo par
    pareja1=par(i,:) %%elijo la pareja 1
    pareja2=impar(i,:)%%elijo la pareja 2
    corte1=pareja1(1:2)%%[1][0][-][-][-][-]%%selecciono los cortes
    corte2=pareja2(3:4)%%[-][-][1][0][-][-]
    corte3=pareja1(4:5)%%[-][-][-][-][1][0]
    hijos=[hijos;corte1,corte2,corte3]%%junto los cortes y los guardo
en un arreglo
end
hijosp=[] %%hijos que tendre
for i=1:size(par)
    pareja1=par(i,:)
    pareja2=impar(i,:)
    corte11=pareja2(1:2)
    corte22=pareja1(3:4)
    corte33=pareja2(4:5)
    hijosp=[hijosp;corte11,corte22,corte33]
end
generacionx=[]
generacionx=[generacionx;hijosp;hijos]
%%%%%%%%%%%Ploteo la nueva generacion
reese=bi2de(generacionx)
normalizado=funciones(reese)
x2 = linspace(-1,1,100);
title(['generacion',num2str(c)])
grid on
hold on
x3=normalizado
```

```
y3=normalizado.^2
plot(x3,y3,'o')
    end
end
```

Cuadro 5.Código selección Aleatoria sin Etilismo

```
%%%%ALMA ELIZA GUERRERO SANCHEZ
c=1  %%variable para contar numero de generaciones
n=6 %%numero de poblacion
A=round(rand(n,6)) %%genero poblacion
A1=bi2de(A)%%convierto de binario a decimal
normalizado=funciones(A1) %%normalizo la poblacion en el intervalo de -
1 a 1 mando llamar la funcion de normalizacion
%%%%%%%%%%%%%%%%%%%%%%%%
x = linspace(-1,1,100);
y =(x.^2)
plot(x,y) %%Ploteo la funcion fitness
grid on
hold on
x1=normalizado
y1=normalizado.^2
plot(x1,y1,'+r')%%ploteo mi poblacion para ver su nivel fitness de
manera grafica
hold on
do = true;
while do
c=c+1 %%contador de generaciones
 if(c==100)
do=false
A1=bi2de(A) %%convierto a decimal
%%%%%%%%%55
normalizado=funciones(A1)  %%normalizo con la funcion
%%%%%%%%%%%%%%SELECCION PSEUDOALEATORIA CON UN CORTE%%%%%%%%%%%%%
arr1 = randi(n,1,n)
arr2= randi(n,1,n)
elegidosO=[]
elegidosT=[]
for(i=1:size(A))
 x=arr1(i)
 y=arr2(i)
 pareja11=A(x,:)
 pareja12=A(y,:)
 elegidosO=[elegidosO;pareja11]
 elegidosT=[elegidosT;pareja12]
end


 for(i=1:size(elegidosO))
     hijos=[]
     hijos2=[]
    pareja1=elegidosO(i,:)
    pareja2=elegidosT(i,:)
    corte1=pareja1(1:3)
    corte2=pareja2(4:6)
    corte11=pareja1(4:6)
```

```
    corte22=pareja2(1:3)
    hijos=[hijos;corte1,corte2]
    hijos2=[hijos2;corte11,corte22]
    %%%%%%%%%%%%%%%%%%%
    parejaBin=bi2de(elegidosO(i,:))
    parejaBin2=bi2de(elegidosT(i,:))
    hijo1Bin=bi2de(hijos)
    hijo2Bin=bi2de(hijos2)
    %%%%%%%%%%%%%%%%%%%
    ANorm=[]
    ANorm=[ANorm;parejaBin;parejaBin2;hijo1Bin;hijo2Bin]
    etil=desnormalizar(ANorm)
    hijo1=etil(1,:)
    hijo2=etil(2,:)
    generacion_x=[]
    generacion_x=[generacion_x;hijo1;hijo2]
    titanes=dec2bin(generacion_x)
    A=[]
    A=titanes
    %%Ploteo
    %alex=str2double(titanes);
    normalizado=funciones(etil)
    x2 = linspace(-1,1,100);
    title(['generacion',num2str(c)])
    grid on
    hold on
    x3=normalizado
    y3=normalizado.^2
    plot(x3,y3,'o')
 end
    end
end
```

Cuadro 6.Código selección Pseudoaleatoria con Etilismo

```
c=1  %%variable para contar numero de generaciones
n=6 %%numero de poblacion
A=round(rand(n,6)) %%genero poblacion
A1=bi2de(A)%%convierto de binario a decimal
normalizado=funciones(A1) %%normalizo la poblacion en el intervalo de -
1 a 1 mando llamar la funcion de normalizacion
%%%%%%%%%%%%%%%%%%%%%%%%
x = linspace(-1,1,100);
y =(x.^2)
plot(x,y) %%Ploteo la funcion fitness
grid on
hold on
x1=normalizado
y1=normalizado.^2
plot(x1,y1,'+r')%%ploteo mi poblacion para ver su nivel fitness de
manera grafica
hold on
do = true;
while do
c=c+1
 if(c==1500)
```

```
 do=false
 A1=bi2de(A)
%%%%%%%%%%55
normalizado=funciones(A1)
%%%%%%%%%%%%%%seleccion aleatoria y cruza con dos cortes%%%%%%%%%%%%%
par=seleccion(A)%%Poblacion seleccionada
impar=Iseleccion(A)%%Poblacion seleccionada
generacionx=[]
%%%%%%%%%%%%%
for i=1:size(par)%%For para que pase por todo mi arreglo par
    hijos=[]
    pareja1=par(i,:) %%elijo la pareja 1
    pareja2=impar(i,:)%%elijo la pareja 2
    corte1=pareja1(1:2)%%[1][0][-][-][-][-]%%selecciono los cortes
    corte2=pareja2(3:4)%%[-][-][1][0][-][-]
    corte3=pareja1(4:5)%%[-][-][-][-][1][0]
    hijos=[hijos;corte1,corte2,corte3]%%junto los cortes y los guardo
en un arreglo
     %%hijos que tendre
    hijosp=[]
    pareja1=par(i,:)
    pareja2=impar(i,:)
    corte11=pareja2(1:2)
    corte22=pareja1(3:4)
    corte33=pareja2(4:5)
    hijosp=[hijosp;corte11,corte22,corte33]
    %%%%%%%%%%%%%%%%%%
    parejaBin=bi2de(pareja1)
    parejaBin2=bi2de(pareja2)
    hijo1Bin=bi2de(hijos)
    hijo2Bin=bi2de(hijosp)
    %%%%%%%%%%%%%%%%%%
    ANorm=[]
    ANorm=[ANorm;parejaBin;parejaBin2;hijo1Bin;hijo2Bin]
    etil=desnormalizar(ANorm)
    hijo1=etil(1,:)
    hijo2=etil(2,:)
    generacion_x=[]
    generacion_x=[generacion_x;hijo1;hijo2]
    titanes=dec2bin(generacion_x)
    A=[]
    A=titanes
    %%Ploteo
    %alex=str2double(titanes);
    normalizado=funciones(generacion_x)
    x2 = linspace(-1,1,100);
    title(['generacion',num2str(c)])
    grid on
    hold on
    x3=normalizado
    y3=normalizado.^2
    plot(x3,y3,'o')
end
%%%%%%%%%%%Ploteo la nueva generacion

    end
end
```

Cuadro 7 . Selección Aleatoria Con Etilismo

```
%%%%ALMA ELIZA GUERRERO SANCHEZ
c=1  %%variable para contar numero de generaciones
n=100 %%numero de poblacion
A=round(rand(n,6)) %%genero poblacion
A1=bi2de(A)%%convierto de binario a decimal
normalizado=funciones(A1) %%normalizo la poblacion en el intervalo de -
1 a 1 mando llamar la funcion de normalizacion
%%%%%%%%%%%%%%%%%%%%%%%%
x = linspace(-1,1,100);
y =(x.^2)
plot(x,y) %%Ploteo la funcion fitness
grid on
hold on
x1=normalizado
y1=normalizado.^2
plot(x1,y1,'+r')%%ploteo mi poblacion para ver su nivel fitness de
manera grafica
hold on
do = true;
while do
c=c+1 %%contador de generaciones
 if(c==5000)
do=false
A1=bi2de(A) %%convierto a decimal
%%%%%%%%%55
normalizado=funciones(A1)  %%normalizo con la funcion
%%%%%%%%%%%%%%SELECCION PSEUDOALEATORIA CON UN CORTE%%%%%%%%%%%%%%
arr1 = randi(n,1,n)
arr2= randi(n,1,n)
elegidosO=[]
elegidosT=[]
for(i=1:size(A))
 x=arr1(i)
 y=arr2(i)
 pareja11=A(x,:)
 pareja12=A(y,:)
 elegidosO=[elegidosO;pareja11]
 elegidosT=[elegidosT;pareja12]
end

hijos=[]
 for(i=1:size(elegidosO))
    pareja1=elegidosO(i,:)
    pareja2=elegidosT(i,:)
    corte1=pareja1(1:3)
    corte2=pareja2(4:6)
    hijos=[hijos;corte1,corte2]
 end

%%%%%%%%%%% PLOTEO MI ULTIMA GENERACION PARA COMPROBAR CON MI
GENERACION
```

```
%%%%%%%%%%% INICIAL
hijosB=bi2de(hijos)
normalizado=funciones(hijosB)
x2 = linspace(-1,1,100);
title(['generacion',num2str(c)])
grid on
hold on
x3=normalizado
y3=normalizado.^2
plot(x3,y3,'o')

    end
end
```

Cuadro 8 . Selección Pseudoaleatoria sin Etilismo

5. Resultados

Selección Aleatoria Sin Etilismo.

Esta fue la que tuvo resultados menos óptimos de los 4 algoritmos.

En este se uso 2 cortes para la cruza y se uso selección aleatoria juntando parejas 1 eran pares y parejas 2 su posición en el arreglo debería ser Impar, se realizo la cruza a 2 cortes y no se le agrego etilismo.

El criterio de paro fue por numero de generación, esto no tanto para evaluar que los puntos fueran óptimos si no para evaluar la evolución de los puntos en los distintos algoritmos aplicados.

Se hicieron para este 5 corridas la primera fue con 300 Generaciones, la segunda con 600 Generaciones, la tercera 1200 Generaciones, 18000 Generaciones y la ultima fueron 10,000.

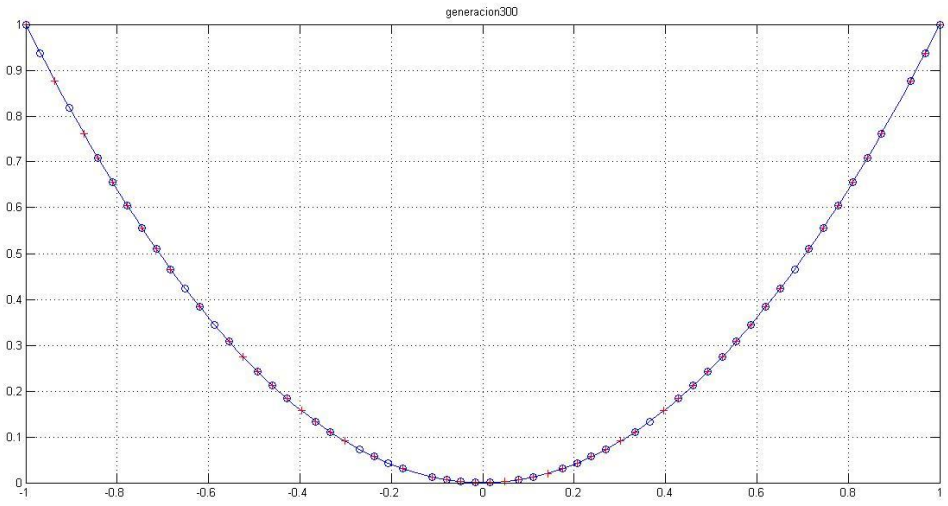

Figura5. Aleatorio Sin Etil A 2 cortes con 300 Generaciones

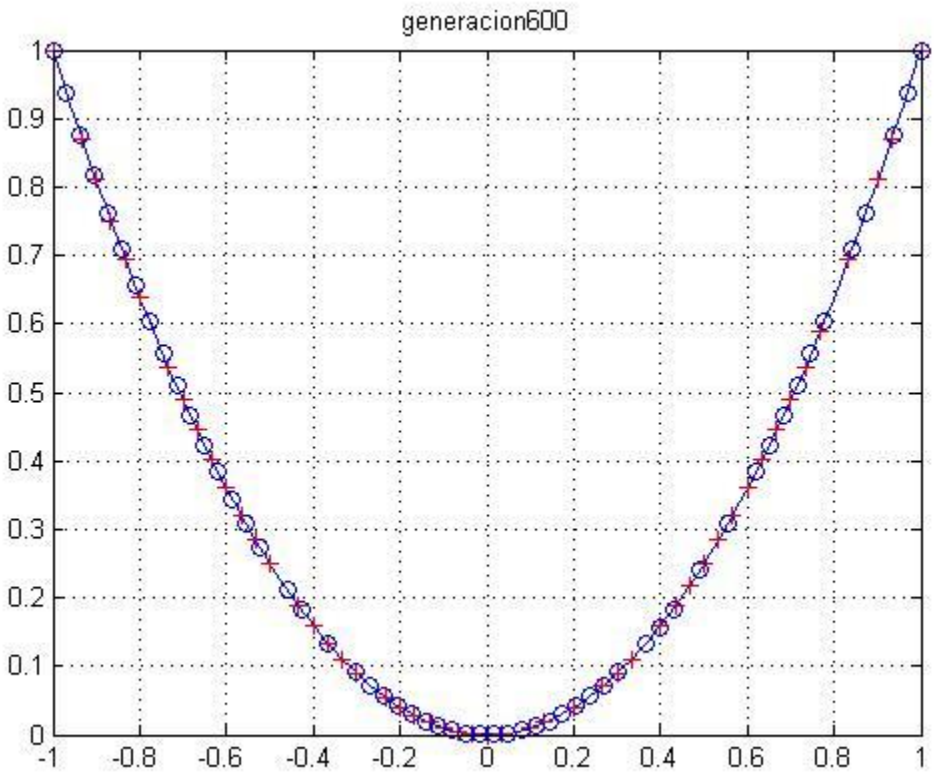

Figura 6. Aleatorio Sin Etilismo A 2 cortes con 600 Generaciones.

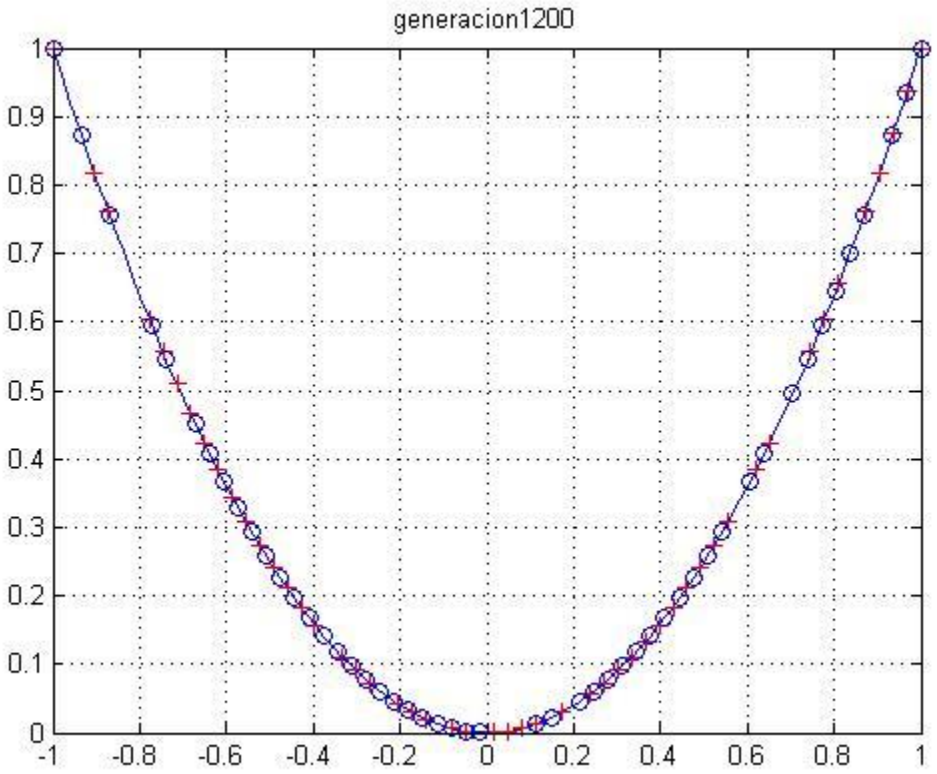

Figura 7. Aleatorio Sin Etilismo A 2 cortes con 1200 Generaciones.

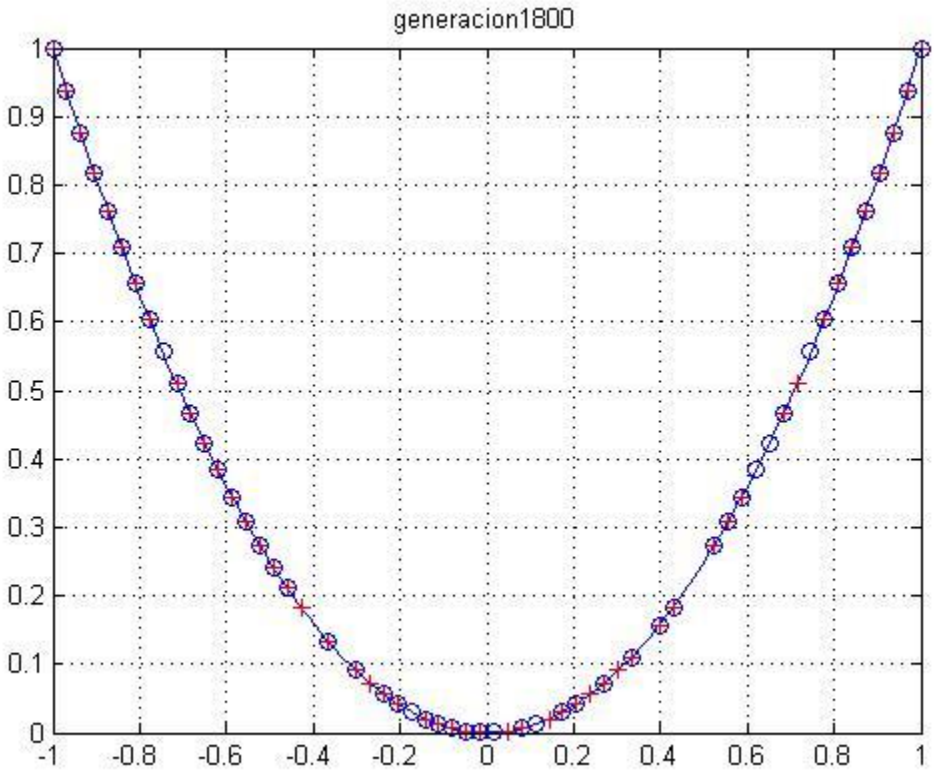

Figura 8. Aleatorio Sin Etilismo A 2 cortes con 1800 Generaciones.

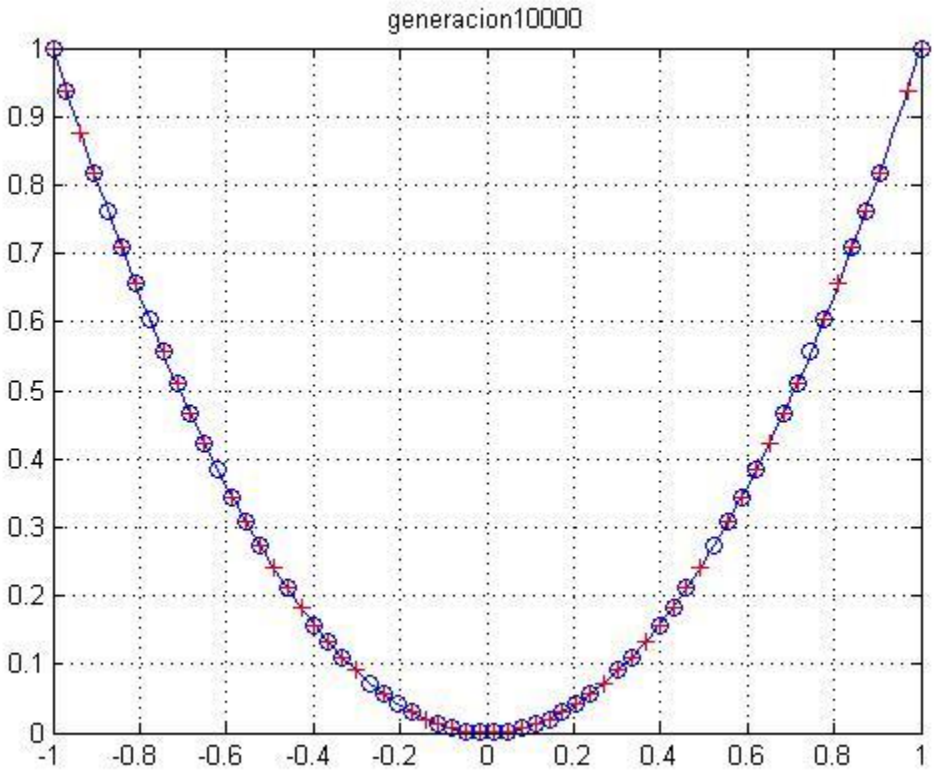

Figura 9. Aleatorio Sin Etilismo A 2 cortes con 1800 Generaciones.

Selección Aleatoria Con Etilismo A Dos Cortes

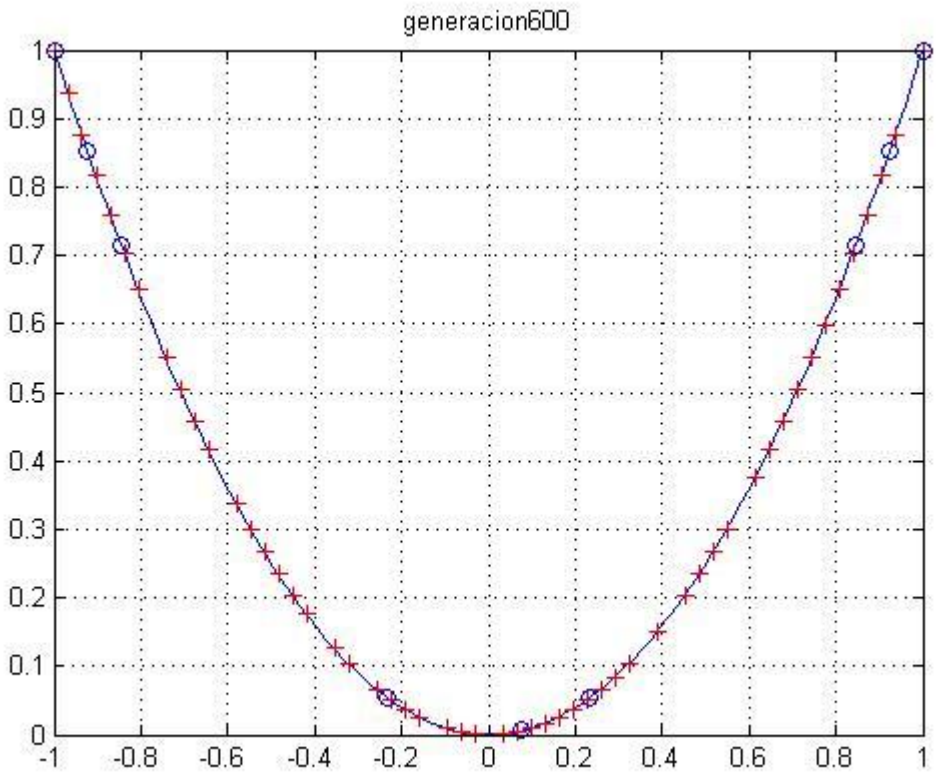

Figura 10. Aleatorio Con Etilismo A 2 cortes con 600 Generaciones.

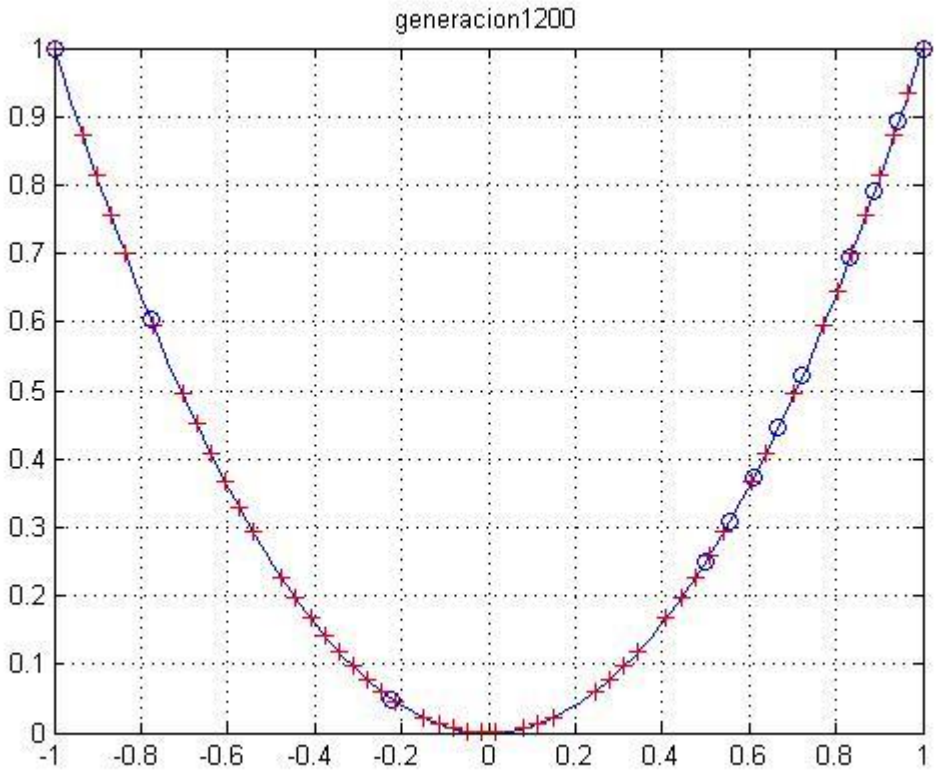

Figura 11. Aleatorio Con Etilismo A 2 cortes con 1200 Generaciones.

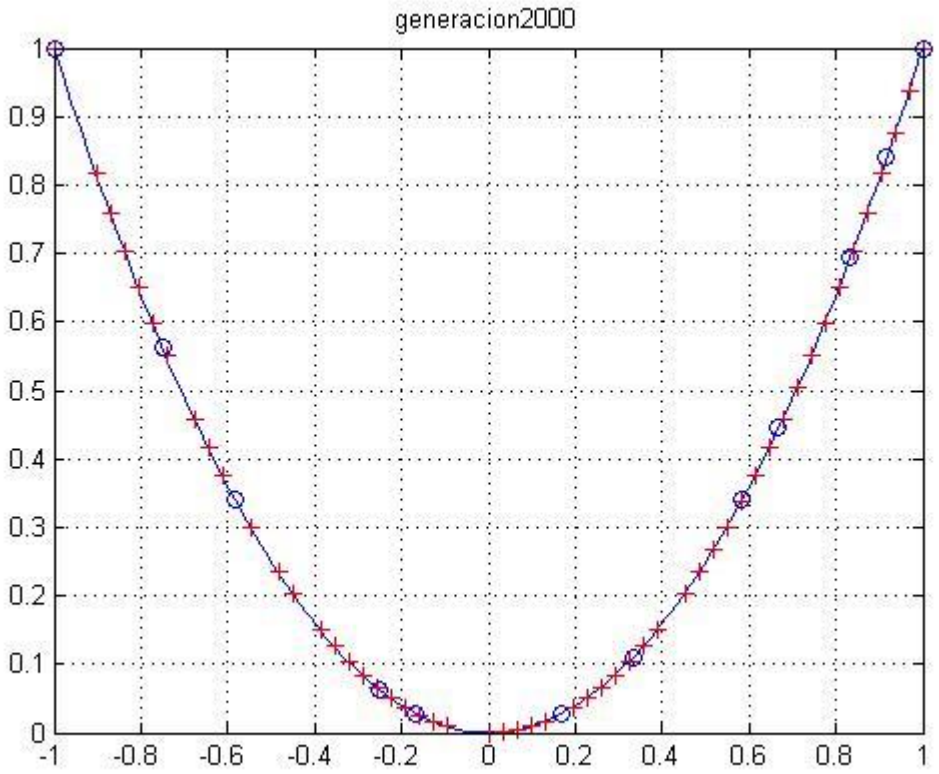

Figura 12. Aleatorio Con Etilismo A 2 cortes con 2000 Generaciones.

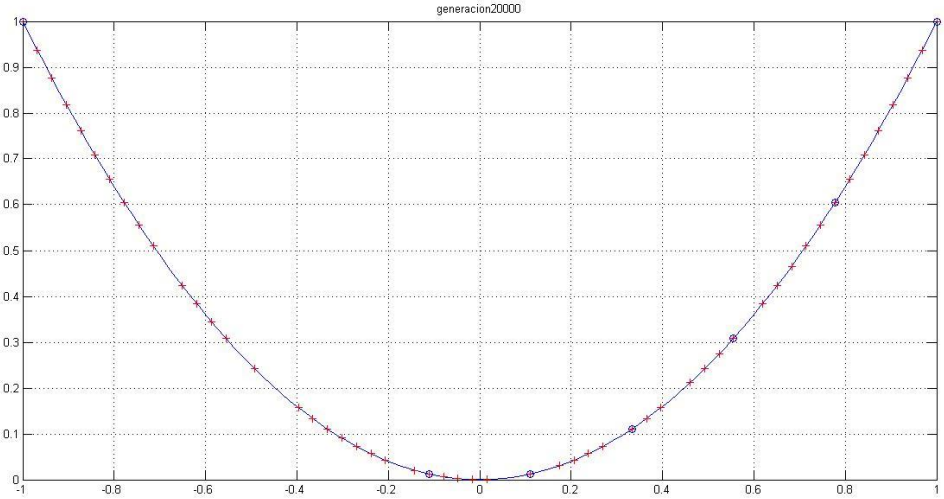

Figura 13. Aleatorio Con Etilismo A 2 cortes con 20000 Generaciones.

Selección Pseudoaleatoria Sin Etilismo A Un Corte

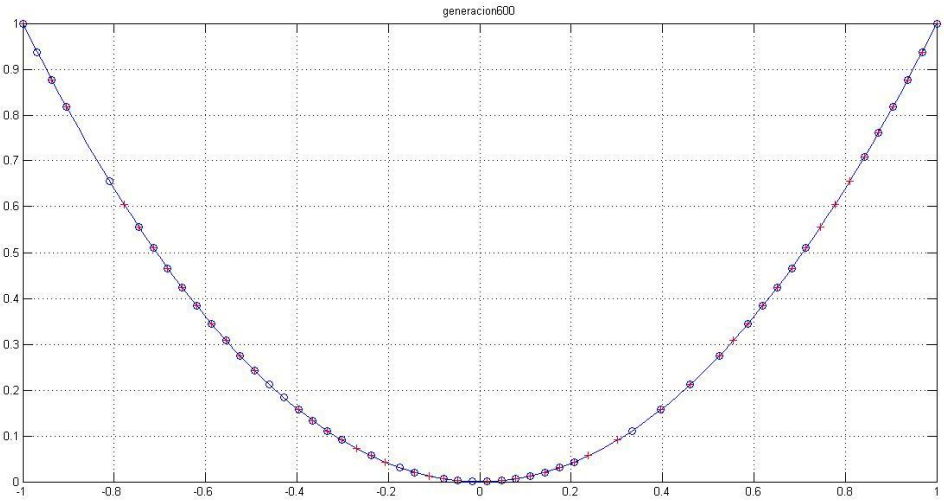

Figura 14. Pseudoaleatorio Sin Etilismo a 1 corte con 600 Generaciones.

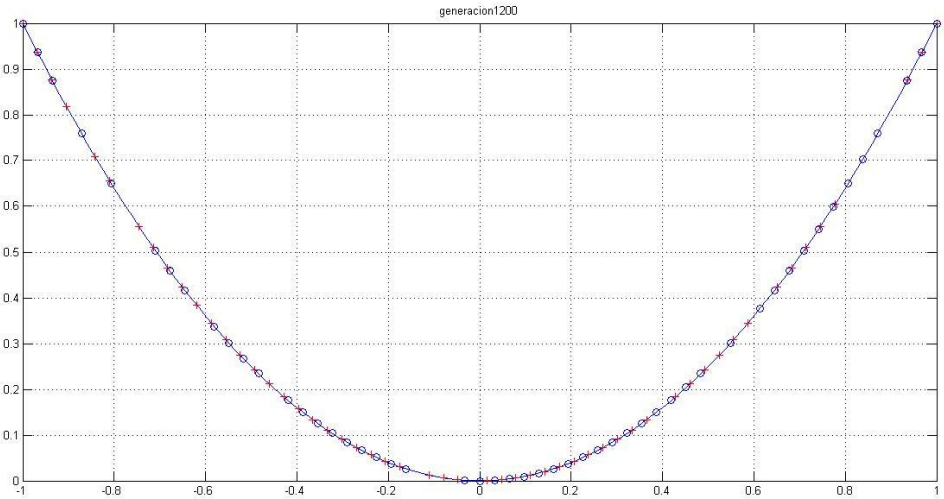

Figura 15. Pseudoaleatorio Sin Etilismo a 1 corte con 1200 Generaciones.

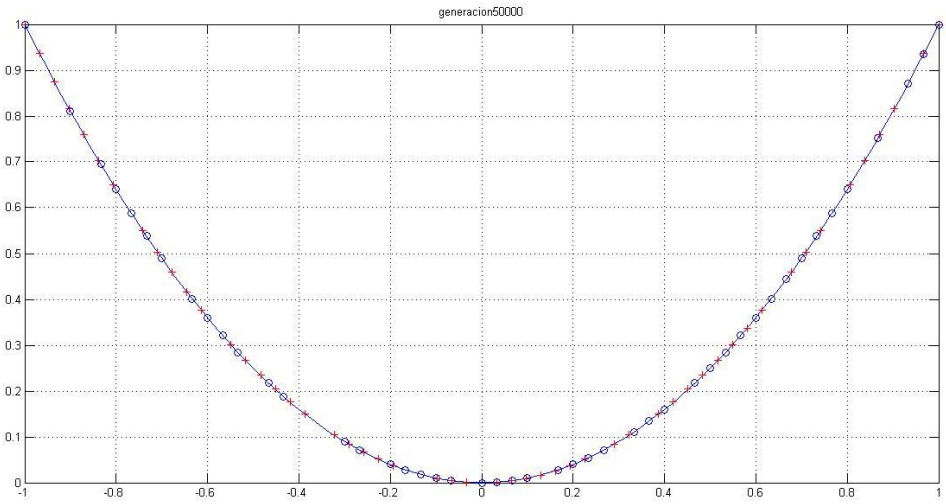

Figura 16. Pseudoaleatorio Sin Etilismo a 1 corte con 60000 Generaciones.

Selección Pseudoaleatoria Con Etilismo A Un Corte

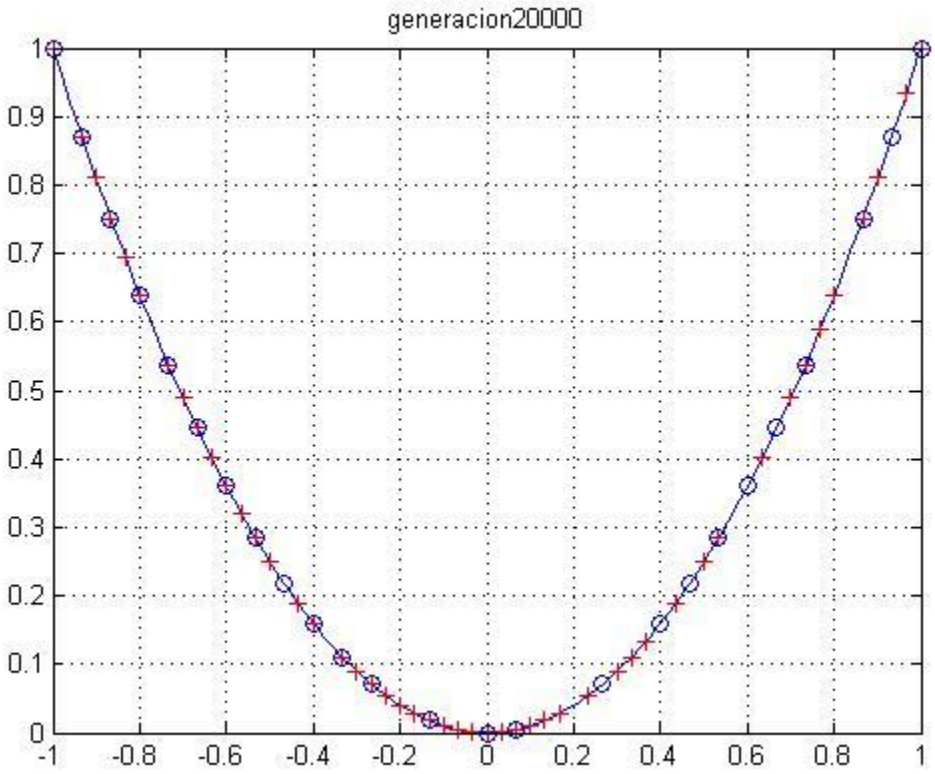

Figura 17. Pseudoaleatorio con Etilismo a 1 corte con 20000 Generaciones.

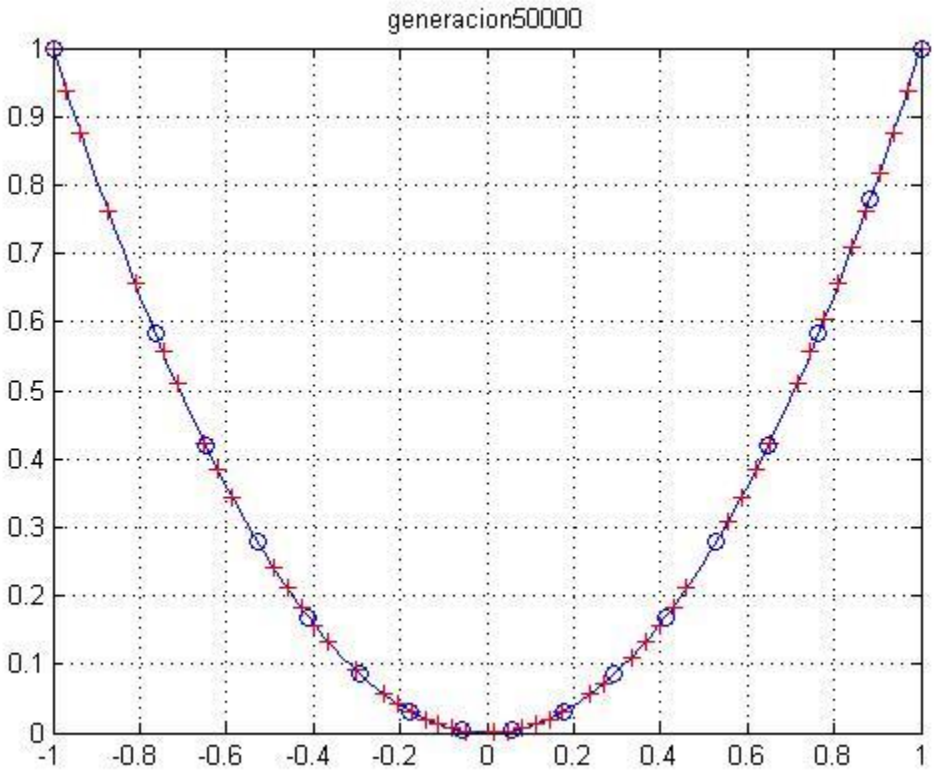

Figura 18. Pseudoaleatorio con Etilismo a 1 corte con 50000 Generaciones

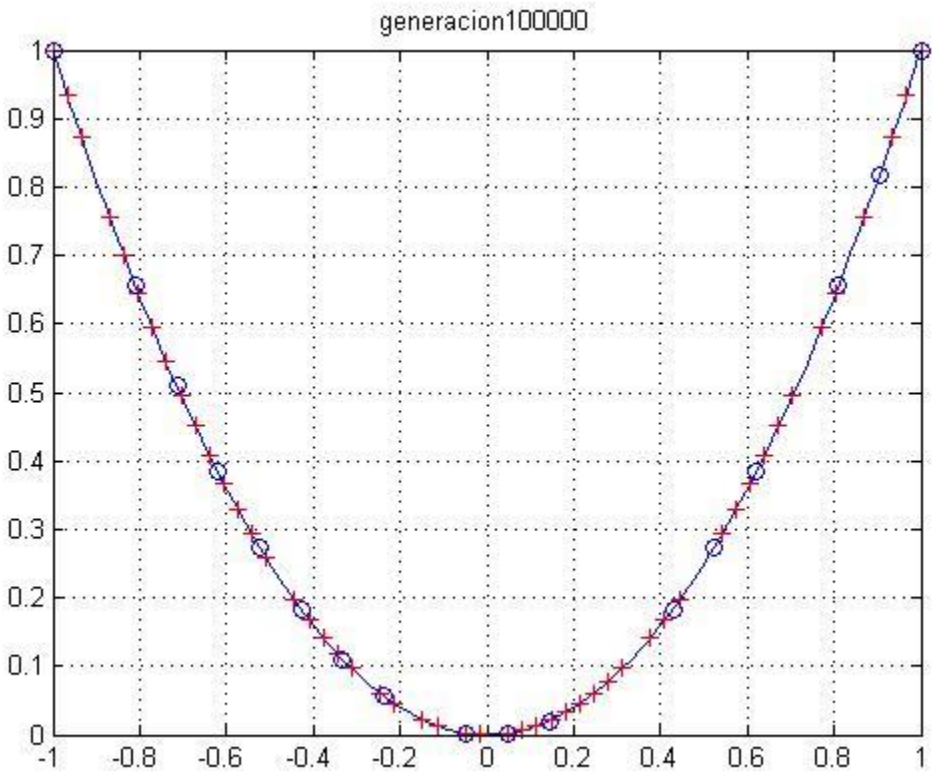

Figura 19. Pseudoaleatorio con Etilismo a 1 corte con 100000 Generaciones

Descripción de Resultados	
Aleatorio sin Etilismo a dos corte	Este algoritmo fue de los que no dio un resultado optimo, ni con el aumento de numero de generaciones.
Aleatorio con Etilismo a dos corte	El aplicarle Etilismo al algoritmo anterior mejoro de manera notable los resultados y dio valores buenos, y cuando se aumentaba el numero de generaciones mejoraba de manera notable el resultado final.
Pseudoaleatorio sin etilismo a un cortes	Este algoritmo en lo particular no dio resultados tan poco óptimos, pero no eran los esperados, conforme se aumentaba el numero de iteraciones se mejoraban muy poco la optimización de los datos.
Pseudoaleatorio con etilismo a un cortes	Este ultimo algoritmo, se esperaba mas de este algoritmo pero de alguna forma no se tuvieron los resultados previstos a pesar de que tenia aplicado el elitismo

De los 4 algoritmos aplicados el mas efectivo fue el de selección aleatoria con etilismo a dos cortes, el segundo fue Pseudoaleatorio con etilismo a un cortes, el tercero es Pseudoaleatorio sin etilismo a un cortes y el ultimo es Pseudoaleatorio con etilismo a un cortes.

Si nos vamos por tiempo de ejecución el que es mejor es el de Pseudoaleatorio con etilismo a un cortes.

6. Discusión

Los resultados anteriores expuestos, como se pudo observar se pueden mejorar aplicando otras técnicas de selección adecuadas y aplicando técnicas como de mutación para realizar una mejor en los datos y ser mas óptimos.

El proceso de cruce y como se realiza la mezcla de los datos, es de las partes que mas interesaron no estoy segura de que otros tipos de corte se puede haber, y aquí la pregunta es si con los cortes realizados ayudan a mejorar la calidad de los datos.

7. Conclusiones

Desde el inicio del ejercicio, uno de mis propósitos no era tanto obtener los resultados mas óptimos, me interesaba mas el desarrollo y ver le proceso de la cruza de los genes así como la aplicación del etilismo y como afectaba en las siguientes generaciones y por ende a los datos finales.

El proceso de selección es la parte más marcante en un algoritmo de selección en definitiva es la selección. Dependiendo de qué tipo de selección se usa marca la pauta para los resultados que se tendrán en el futuro.

De igual manera el proceso de cortes es de gran influencia para realizar la combinación de genes.

El etilismo juego un papel importante en este ejercicio porque se encargo tomar solo los valores mas fuertes para la siguiente generación.

8. Biografía

Marcos; Rivero Gestal (Daniel; Rabuñal, Juan Ramón; Dorado, Julián; Pazos, Alejandro), & Gestal, M. (2010). *Introducción a los algoritmos genéticos y la programación genética*. Universidade da Coruña.

Coello, C. A. C., & Zacatenco, C. S. P. (2004). Introducción a la computación evolutiva. *México*.

Cao, Y. J., & Wu, Q. H. (1999). Teaching genetic algorithm using MATLAB. *International journal of electrical engineering education*, *36*(2), 139-153