
PRACTICA 2: REPRESENTACIÓN DE INDIVIDUOS

PRIMER PARCIAL

NOMBRE:

MARTÍNEZ ROMERO JONATHAN YAIR

GRUPO:

3CM5

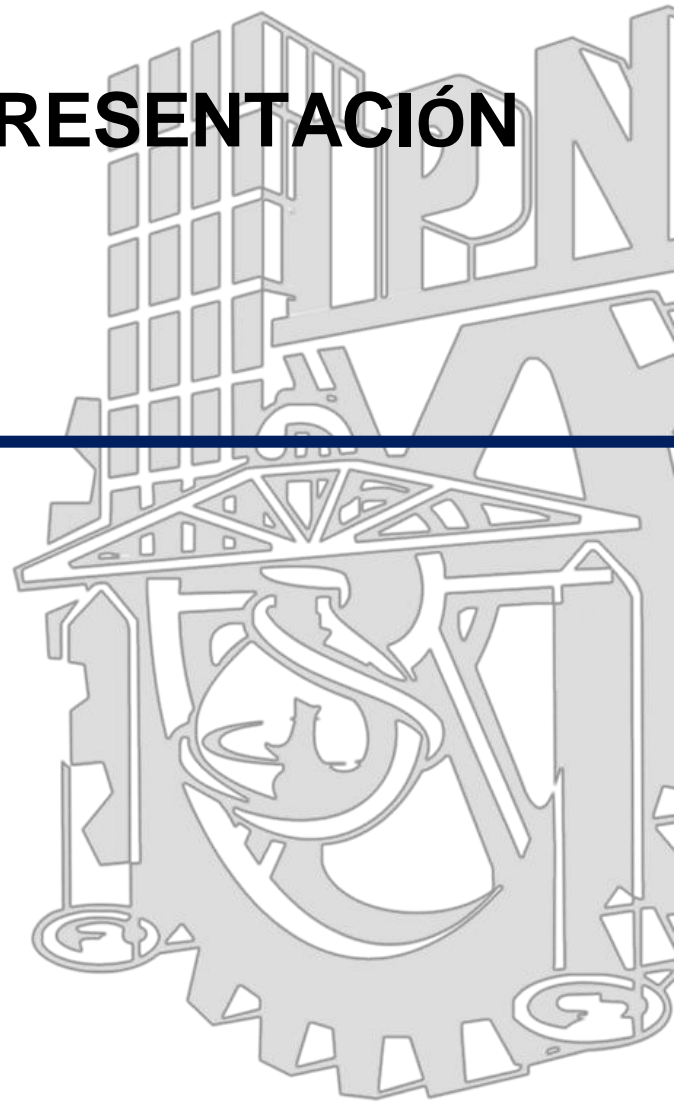
FECHA:

13 DE SEPTIEMBRE DE 2018

UNIDAD DE APRENDIZÁJE:

ALGORITMOS GENETICOS

PRACTICA NO.: 2



INSTITUTO POLITÉCNICO NACIONAL



ESCOM

**INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO**

INTRODUCCIÓN

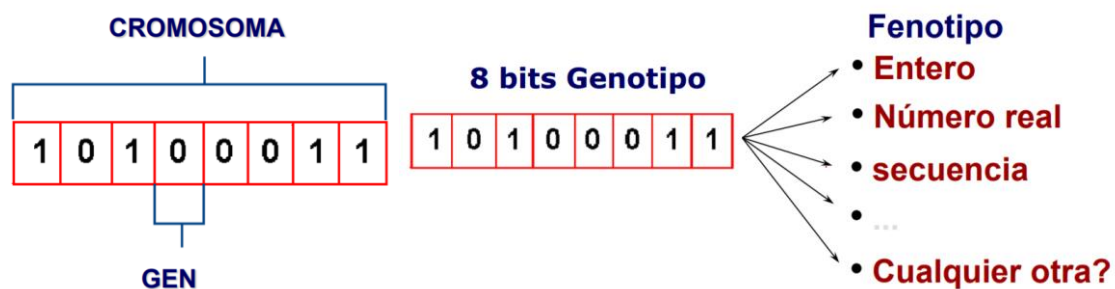
Debemos disponer de un mecanismo para codificar un individuo como un genotipo.

Existen muchas maneras de hacer esto y se ha de elegir la más relevante para el problema en cuestión.

Una vez elegida una representación, hemos de tener en mente como los genotipos (codificación) serán evaluados y qué operadores genéticos hay que utilizar.

- **REPRESENTACIÓN BINARIA**

La representación de un individuo se puede hacer mediante una codificación discreta, y en particular binaria.



- **CÓDIGOS DE GRAY**

La investigación en AGs fue que el uso de la representación binaria no mapea adecuadamente el espacio de búsqueda con el espacio de representación. La codificación de Gray es parte de una familia de representaciones. Podemos convertir cualquier número binario a un código de Gray haciendo XOR a sus bits consecutivos de derecha a izquierda. Por ejemplo, dado el número 101 en binario, haríamos: $1 \oplus 0 = 1$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, produciéndose (el último bit de la izquierda permanece igual) 0111, el cual es el código de Gray equivalente. Algunos investigadores han demostrado empíricamente que el uso de códigos de Gray mejora el desempeño del AG.

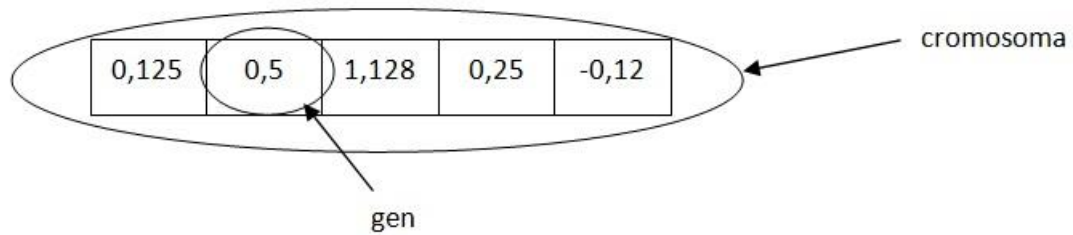
- **REPRESENTACIÓN REAL**

Una forma natural de codificar una solución es utilizando valores reales como genes

Muchas aplicaciones tienen esta forma natural de codificación

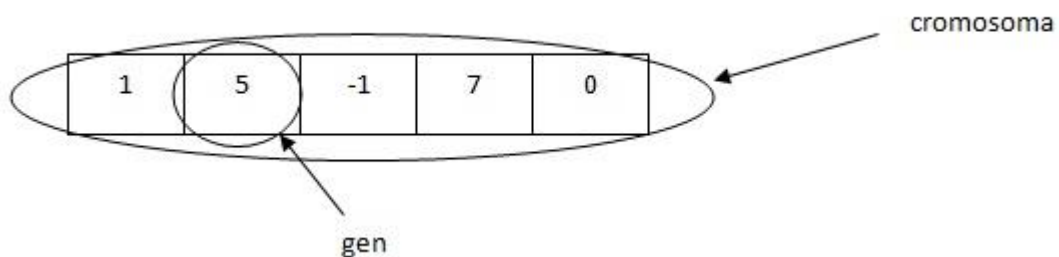
Los individuos se representan como vectores de valores reales:

La función de evaluación asocia a un vector un valor real de evaluación:



• REPRESENTACIÓN ENTERA

En ella se utiliza un vector cuya longitud es la del número de genes de cada individuo y el valor que puede tomar cada elemento es un número entero.



PLANTEAMIENTO DEL PROBLEMA

Instrucciones.

Realizar un programa con 4 tipos de representación de individuos, que se explicaron en clase. En lenguaje C o C++.

2.La representación binaria debe llenar un arreglo o matriz con 10 individuos, cada alelo se crea de manera aleatoria. Y así con cada una de las representaciones.

3.En total deben ser 4 programas o bien uno con un menú por representación.

DESARROLLO

Para la codificación se ocupó el lenguaje de Programación C mediante la utilización del IDE de Dev++.

Para su realización se hizo mediante un menú en el cual era posible elegir el algoritmo que deseábamos implementar de esta manera; para que se realizara correctamente fue necesario el uso y la implementación de números aleatorios que pudieran crear una matriz bidimensional en la que se representaron a 10 individuos es decir 10 cromosomas con 7 alelos cada uno; de esta manera se pudo crear una población con la cual se va a poder trabajar posteriormente conforme avance el curso.

CODIFICACIÓN

```
1. #include <stdio.h>
2. #include <wchar.h>
3. #include <locale.h>
4. #include <stdlib.h>
5. #include <time.h>
6. #include <math.h>
7.
8. void ImprimeCromosoma(int Alelo[][7])
9. {
10.     for(int i=0;i<10;i++)
11.     {
12.         printf("Individuo %d\t",i+1);
13.         for(int j=0;j<7;j++)
14.         {
15.             printf("%d",Alelo[i][j]);
16.         }
17.         printf("\n");
18.     }
19.
20.     printf("\n");
21. }
22.
23. void ImprimeCromosomaGray(int Alelo[][7],int AleloFinal[
24. ][7],int Decimales[])
25. {
26.     for(int i=0;i<10;i++)
27.     {
28.         printf("Individuo
29. %d \t(%d)\t",i+1,Decimales[i]);
30.         for(int j=0;j<7;j++)
31.         {
32.             printf("%d",Alelo[i][j]);
33.         }
34.         printf("=>");
35.         for(int j=0;j<7;j++)
36.         {
37.             printf("%d",AleloFinal[i][j]);
38.         }
39.         printf("\n");
40.     }
41.
42.     printf("\n");
43. }
44.
45. void ImprimeCromosoma(float Alelo[][7])
46. {
47.     for(int i=0;i<10;i++)
48.     {
49.         printf("Individuo %d\t",i+1);
50.         for(int j=0;j<7;j++)
51.         {
52.             printf("%.21f",Alelo[i][j]);
53.         }
```

```

54.         printf("\n");
55.
56.     }
57.
58.     printf("\n");
59. }
60.
61.
62. void binaria()
63. {
64.     srand(time(NULL));
65.
66.     printf("\nRepresentación binaria\n");
67.     int Alelo[10][7];
68.
69.     for(int i=0;i<10;i++)
70.     {
71.         for(int j=0;j<7;j++)
72.         {
73.             Alelo[i][j]=rand() % 1001;
74.             if(Alelo[i][j]%2)
75.             {
76.                 Alelo[i][j] = 1;
77.             }
78.             else
79.             {
80.                 Alelo[i][j] = 0;
81.             }
82.         }
83.     }
84.
85.
86.     ImprimeCromosoma(Alelo);
87. }
88.
89. void gray()
90. {
91.     printf("\nCódigos de Gray\n");
92.
93.     srand(time(NULL));
94.
95.     int Alelo[10][7];
96.     int AleloFinal[10][7];
97.     int Decimal[10];
98.
99.     for(int i=0;i<10;i++)
100.    {
101.        for(int j=0;j<7;j++)
102.        {
103.            Alelo[i][j]=rand() % 1001;
104.            if(Alelo[i][j]%2)
105.            {
106.                Alelo[i][j] = 1;
107.            }
108.            else
109.            {
110.                Alelo[i][j] =0;

```

```

111.         }
112.     }
113. }
114.
115. //ImprimeCromosoma (Alelo);
116.
117. int temp = 0;
118. int td = 0;
119.
120. for(int i=0;i<10;i++)
121. {
122.
123.     td = 0;
124.
125.     for(int k=0;k<7;k++)
126.     {
127.         if(k == 0)
128.         {
129.             AleloFinal[i][k] = Alelo[i][k];
130.         }
131.         else
132.         {
133.             temp = k-1;
134.             AleloFinal[i][k] = Alelo[i][k]^Alelo[i][
temp];
135.         }
136.
137.
138.
139.     }
140.
141.     for(int k=0;k<7;k++)
142.     {
143.         td += Alelo[i][6-k]*pow(2,k);
144.
145.
146.     }
147.
148.     Decimal[i] = td;
149.
150. }
151.
152. ImprimeCromosomaGray(Alelo,AleloFinal,Decimal);
153.
154. }
155.
156. void reales()
157. {
158.     printf("\nCodificación en números reales\n");
159.
160.     srand(time(NULL));
161.
162.     float Alelo[10][7];
163.
164.     for(int i=0;i<10;i++)
165.     {
166.         for(int j=0;j<7;j++)

```

```

167.         {
168.             Alelo[i][j]=(double)rand()/(RAND_MAX+1)*(99
+1;
169.         }
170.     }
171. }
172.
173.
174.     ImprimeCromosoma(Alelo);
175.
176.
177. }
178.
179. void enteros()
180. {
181.     srand(time(NULL));
182.
183.     printf("\nCodificación en números enteros\n");
184.     int Alelo[10][7];
185.
186.     for(int i=0;i<10;i++)
187.     {
188.         for(int j=0;j<7;j++)
189.         {
190.             Alelo[i][j]=rand() % 100;
191.
192.         }
193.     }
194.
195.
196.     ImprimeCromosoma(Alelo);
197.
198.
199. }
200.
201. int main()
202. {
203.     setlocale(LC_ALL, "");
204.     printf("ESCUELA SUPERIOR DE CÓMPUTO\n");
205.     printf("Algoritmos Genéticos\n");
206.     printf("Práctica 2\n\n");
207.
208.     char continua = 's';
209.
210.     while(continua == 's')
211.     {
212.         printf("Seleccione el tipo de representación que
desea realizar\n");
213.         printf("1.- Representación binaria\n");
214.         printf("2.- Códigos de Gray\n");
215.         printf("3.- Codificación en números reales\n");
216.         printf("4.- Codificación en números enteros\n");
217.         int opc = 0;
218.
219.
220.         scanf("%d",&opc);
221.

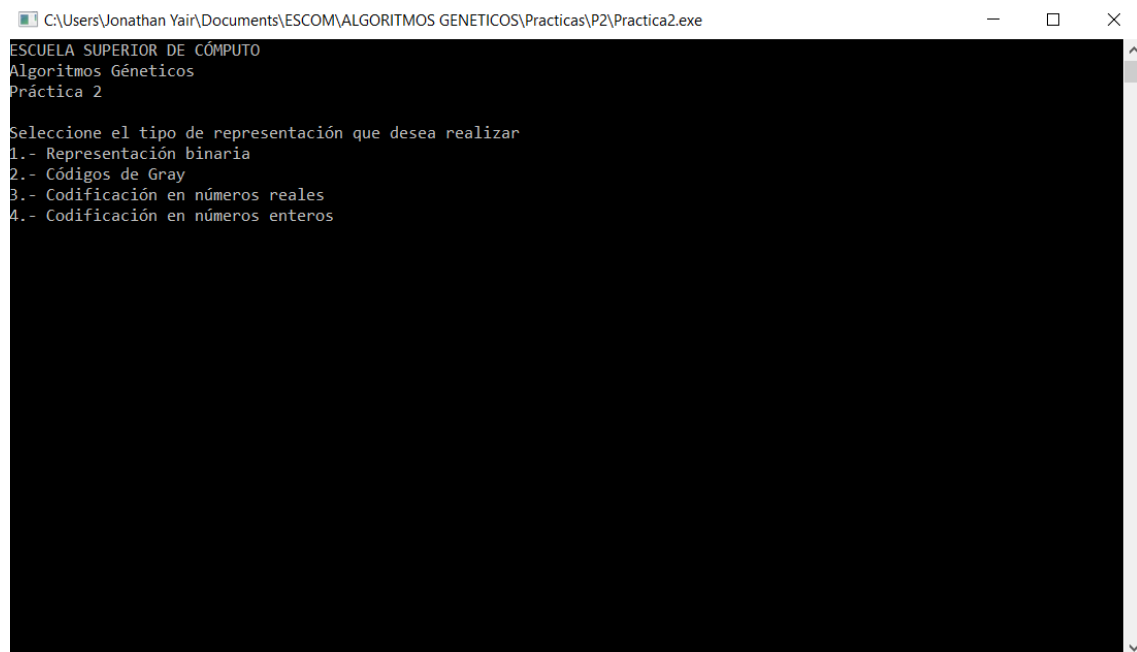
```

```

222.         switch(opc)
223.         {
224.             case 1:
225.                 binaria();
226.             break;
227.             case 2:
228.                 gray();
229.             break;
230.             case 3:
231.                 reales();
232.             break;
233.             case 4:
234.                 enteros();
235.             break;
236.             default:
237.                 printf("No es una opción valida\n");
238.             break;
239.         }
240.
241.         printf("Desea continuar? SI (s) , NO (n)\n");
242.         fflush( stdin );
243.         scanf("%c",&continua);
244.     }
245.     return 0;
246. }

```

PANTALLA



```

C:\Users\Jonathan Yair\Documents\ESCOM\ALGORITMOS GENETICOS\Practicas\P2\Practica2.exe
ESCUELA SUPERIOR DE CÓMPUTO
Algoritmos Genéticos
Práctica 2

Seleccione el tipo de representación que desea realizar
1.- Representación binaria
2.- Códigos de Gray
3.- Codificación en números reales
4.- Codificación en números enteros

```



```
C:\Users\Jonathan Yair\Documents\ESCOM\ALGORITMOS GENETICOS\Practicas\P2\Practica2.exe
ESCUELA SUPERIOR DE CÓMPUTO
Algoritmos Genéticos
Práctica 2

Seleccione el tipo de representación que desea realizar
1.- Representación binaria
2.- Códigos de Gray
3.- Codificación en números reales
4.- Codificación en números enteros
1

Representación binaria
Individuo 1  0,1,0,0,1,1,0,
Individuo 2  0,0,0,1,0,0,1,
Individuo 3  1,0,0,0,0,0,1,
Individuo 4  0,0,0,0,0,1,1,
Individuo 5  1,0,1,1,0,1,1,
Individuo 6  1,1,1,0,0,0,0,
Individuo 7  1,0,0,1,1,1,0,
Individuo 8  1,1,0,0,0,1,1,
Individuo 9  1,0,0,0,0,0,0,
Individuo 10 1,0,0,1,1,1,0,

Desea continuar? SI (s) , NO (n)
```

```
C:\Users\Jonathan Yair\Documents\ESCOM\ALGORITMOS GENETICOS\Practicas\P2\Practica2.exe
Desea continuar? SI (s) , NO (n)
s
Seleccione el tipo de representación que desea realizar
1.- Representación binaria
2.- Códigos de Gray
3.- Codificación en números reales
4.- Codificación en números enteros
2

Códigos de Gray
Individuo 1  (36)  0100100=>0110110
Individuo 2  (5)   0000101=>0000111
Individuo 3  (119) 1110111=>1001100
Individuo 4  (111) 1101111=>1011000
Individuo 5  (62)  0111110=>0100001
Individuo 6  (33)  0100001=>0110001
Individuo 7  (58)  0111010=>0100111
Individuo 8  (19)  0010011=>0011010
Individuo 9  (63)  0111111=>0100000
Individuo 10 (52)  0110100=>0101110

Desea continuar? SI (s) , NO (n)
```

```
C:\Users\Jonathan Yair\Documents\ESCOM\ALGORITMOS GENETICOS\Practicas\P2\Practica2.exe
Codificación en números reales
Individuo 1 29.37,85.50,6.65,1.21,20.10,96.44,85.80,
Individuo 2 36.03,46.55,48.43,51.63,34.44,2.58,23.52,
Individuo 3 20.82,92.57,92.98,56.47,70.09,80.60,59.44,
Individuo 4 43.81,26.52,36.92,89.71,48.30,51.49,61.92,
Individuo 5 59.34,61.37,78.69,86.01,42.20,18.73,76.75,
Individuo 6 34.57,25.93,22.65,87.76,71.18,85.59,80.98,
Individuo 7 32.46,56.06,40.78,9.70,15.20,78.96,47.28,
Individuo 8 95.78,30.50,15.44,59.69,98.27,2.34,64.76,
Individuo 9 11.96,55.42,47.99,20.40,38.38,79.72,16.49,
Individuo 10 23.38,7.95,34.63,74.63,54.29,70.55,36.88,
Desea continuar? SI (s) , NO (n)
```

```
C:\Users\Jonathan Yair\Documents\ESCOM\ALGORITMOS GENETICOS\Practicas\P2\Practica2.exe
Seleccione el tipo de representación que desea realizar
1.- Representación binaria
2.- Códigos de Gray
3.- Codificación en números reales
4.- Codificación en números enteros
4
Codificación en números enteros
Individuo 1 82,45,47,84,17,65,11,
Individuo 2 34,50,93,16,2,88,99,
Individuo 3 47,55,84,79,94,79,99,
Individuo 4 54,79,90,61,90,29,30,
Individuo 5 2,40,39,43,90,9,94,
Individuo 6 96,77,91,56,42,78,68,
Individuo 7 70,89,21,67,90,64,77,
Individuo 8 73,54,36,34,82,87,16,
Individuo 9 82,17,97,92,47,33,34,
Individuo 10 19,2,64,45,37,58,40,
Desea continuar? SI (s) , NO (n)
s
Seleccione el tipo de representación que desea realizar
1.- Representación binaria
2.- Códigos de Gray
3.- Codificación en números reales
4.- Codificación en números enteros
5
No es una opción valida
Desea continuar? SI (s) , NO (n)
```

CONCUSIONES

Una de las cosas que pude observar en la realización de la presenta practica es en la importancia en el uso de múltiples representaciones las cuales pueden ser usadas dependiendo el problema que estemos tratando así como los datos con los cuales se esta trabajando, de esta forma podemos enriquecer nuestro conocimiento en cuanto a la manera correcta de usar las distintas poblaciones.

En mi caso particular considero que la representación decimal y la de Gray son las más útiles ya que de esta forma hay una mayor probabilidad de que los distintos elementos salgan distintos evitando que las operaciones coincidan en el caso de diversos algoritmos utilizados más adelante.