

# PRACTICA III: MÉTODO RULETA

ALGORITMOS GENÉTICOS

3CM5

*Colín Varela, Alejandro*

## Contents

<b>1</b>	<b>Introducción</b>	<b>2</b>
<b>2</b>	<b>Desarrollo</b>	<b>2</b>
2.1	Código . . . . .	3
<b>3</b>	<b>Pruebas</b>	<b>7</b>
<b>4</b>	<b>Conclusiones</b>	<b>11</b>

## 1 Introducción

En esta practica programamos un Algoritmo genético basado en la selección por ruleta y gratificamos los por cientos de los máximos y mínimos para ver como se balancean entre si.

## 2 Desarrollo

En esta práctica se opto por utilizar la librería 'plot-sdl' la cual toma provecho de las funciones de graficación de sdl para darle al programador una manera intuitiva y sencilla de poblar una gráfica con valores que posteriormente se puedan graficar. Simplemente le dimos los valores de máximos y mínimos que obteníamos de generación en generación.

## 2.1 Código

Iniciamos con un menu pidiendo los valores de los individuos y de las generaciones. Posteriormente procedemos a generar dicho numero de individuos y rellenarlos con bits aleatorios. Al final hacemos un for con el numero de generaciones donde llamaremos a la función generar e iremos guardando los minimos y los máximos en la lista de coordenadas de la gráfica.

```
152     int gen,ind;
153     printf("\n\nPractica 3: Seleccion por ruleta\n");
154     printf("Alumno: Alejandro Colin Varela\n");
155     do
156     {
157         printf("\nIndique el numero de individuos (Numero par) [2-32] : ");
158         scanf("%d",&ind);
159         if(ind%2!=0)
160         {
161             printf("\nFavor de usar un numero par!\n");
162             ind=0;
163         }
164     }while(ind>32||ind<2);
165     do
166     {
167         printf("\nIndique un numero de generaciones [1-15] : ");
168         scanf("%d",&gen);
169     }while(gen>15||gen<1);
170     srand(time(NULL));
171     int **individuos,**descendencia;
172     int i,i2;
173     individuos = (int**) malloc(ind*sizeof(int*));
174     descendencia = (int**) malloc(ind*sizeof(int*));
175     for(i=0;i<ind;i++)
176     {
177         individuos[i] = (int*) malloc(5*sizeof(int));
178         descendencia[i] = (int*) malloc(5*sizeof(int));
179         for (int i2 = 0; i2 < 5; ++i2)
180         {
181             individuos[i][i2] = rand()%2;
182             descendencia[i][i2] = 0;
183         }
184     }
185     coordlist coordinate_list=NULL;
186     double min,max;
187     double max2=0;
188     for(i=0;i<gen;i++)
189     {
190         min=1024;
191         max=0;
192         printf("Generacion n°%d : \n", i+1);
193         generar(individuos,descendencia,ind,&min,&max);
194         coordinate_list=push_back_coord(coordinate_list,0,i,max);
195         coordinate_list=push_back_coord(coordinate_list,1,i,min);
196         if(max>max2)max2=max;
197     }
198     free(individuos);
199     free(descendencia);
200
```

Aquí tenemos la funciones que posteriormente conformaran a la función de generar

```
15 int decimal(int *ind)
16 {
17     int i;
18     int cont = 0;
19     for (i = 0; i < 5; ++i)
20     {
21         cont = cont + (ind[i]*pow(2,4-i));
22     }
23     return cont;
24 }
25 void imprimir_ind(int **ind,int x)
26 {
27     int i,i2;
28     for (i = 0; i < x; ++i)
29     {
30         printf("\nI[%d] = ", i+1);
31         for (i2 = 0; i2 < 5; ++i2)
32         {
33             printf("%d ",ind[i][i2]);
34         }
35     }
36     printf("\n\n");
37 }
38 void copiar_ind(int *ind, int *des)
39 {
40     int i;
41     for (int i = 0; i < 5; ++i)
42     {
43         des[i] = ind[i];
44     }
45 }
46 void mutar(int *des)
47 {
48     int i;
49     int s;
50     for (int i = 0; i < 5; ++i)
51     {
52         s = rand()%100;
53         if(s<=5) des[i] = abs(des[i]-1);
54     }
55 }
56 void cruzar(int *ind1,int *ind2,int punto)
57 {
58     int i,val;
59     for (i = punto; i < 5; ++i)
60     {
61         val = ind1[i];
62         ind1[i] = ind2[i];
63         ind2[i] = val;
64     }
65 }
```

Lo siguiente es la función de generar que hara todas las operaciones necesarias para conseguir seleccionar individuos por metodo de la ruleta.

```

58 int i,cnt,cnt2;
59 double *valores = (double*) malloc(in*sizeof(double));
60 double *sumaprom = (double*) malloc(in*sizeof(double));
61 double suma=0;
62 double dec=0;
63 double random;
64 imprimir_ind(ind,in);
65 for(cnt=0;cnt<in;cnt++)
66 {
67     valores[cnt] = pow(decimal(ind[cnt]),2);
68     if(valores[cnt]>*max)
69     {
70         *max=valores[cnt];
71     }
72     if(valores[cnt]<*min)
73     {
74         *min=valores[cnt];
75     }
76     suma = suma + valores[cnt];
77 }
78 *max = *max / suma;
79 *min = *min / suma;
80 for(cnt=0;cnt<in;cnt++)
81 {
82     valores[cnt] = valores[cnt] / suma;
83     if(cnt!=0)
84     sumaprom[cnt] = sumaprom[cnt-1] + valores[cnt];
85     else
86     sumaprom[cnt] = valores[cnt];
87 }
88 for (cnt = 0; cnt < in; ++cnt)
89 {
90     random = rand()%100;
91     dec = random / 100;
92     for(cnt2=0;cnt2<in;cnt2++)
93     {
94         if(cnt2==0&&dec<=sumaprom[cnt2])
95         {
96             copiar_ind(ind[cnt2],des[cnt]);
97             break;
98         }
99         else if(dec>sumaprom[cnt2-1]&&dec<=sumaprom[cnt2])
100         {
101             //Copiar individuo
102             copiar_ind(ind[cnt2],des[cnt]);
103             break;
104         }
105         else if(dec>1)
106         {
107             copiar_ind(ind[in-1],des[cnt]);
108             break;
109         }
110     }
111 }
112 }

```

Finalmente se hace la cruza, la mutación y la descendencia se copia a los individuos para la siguiente generación.

```
122 //Cruza
123 for ( cnt = 0; cnt < in; cnt+=2)
124 {
125     cruzar(des[cnt],des[cnt+1],(rand()%4) + 1);
126 }
127 //Muta
128 for (cnt = 0; cnt < in; ++cnt)
129 {
130     mutar(des[cnt]);
131 }
132 //Copia la descendencia en los nuevo individuos
133 for (cnt = 0; cnt < in; ++cnt)
134 {
135     copiar_ind(des[cnt],ind[cnt]);
136 }
137 free(valores);
138 free(sumaprom);
139 }
```

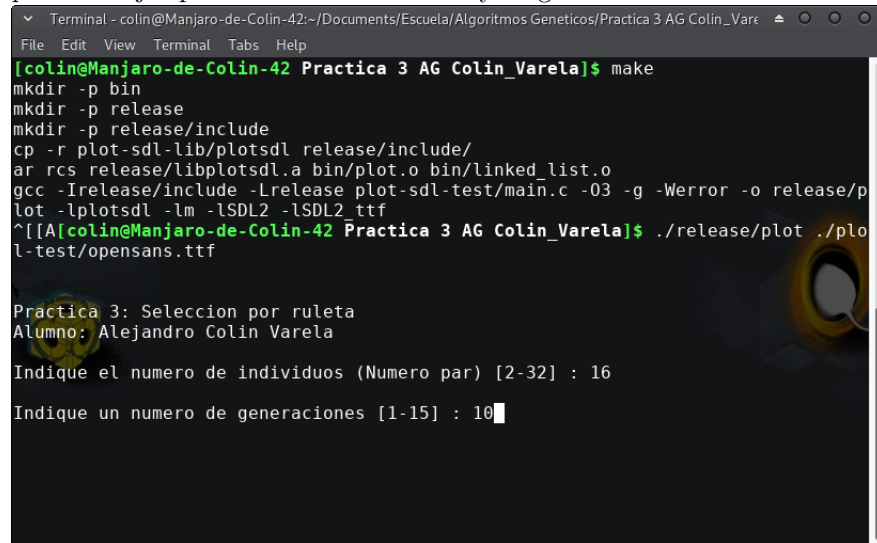
En esta parte le damos los últimos valores a la gráfica para que tenga un formato mas presentable y le pasamos la lista de coordenadas que fue entregada antes con los minimos y los maximos..

```
202 plot_params params;
203
204 params.screen width=800;
205 params.screen height=640;
206 params.plot_window_title="Numeros Aleatorios";
207 params.font_text_path=argv[1];
208 params.font_text size=18;
209 params.caption_text x="";
210 params.caption_text y="";
211 params.caption_list = caption_list;
212 params.coordinate_list = coordinate_list;
213 params.scale_x = 5;
214 params.scale_y = 1;
215 params.max_x = gen;
216 params.max_y = max2;
217
218 int ret = plot_graph(&params);
219
220 if (ret == EXIT_FAILURE)
221 {
222     printf("plot_graph return with status %d\n",ret);
223     return EXIT_FAILURE;
224 }
225 return EXIT_SUCCESS;
226 }
```

Nota: Favor de leer el README.txt para mas información sobre como compilar y correr la practica

### 3 Pruebas

Primero abrimos una consola en la carpeta de la practica y compilamos con 'make', nos debe mostrar esta salida para una compilación exitosa. Posteriormente corremos con el comando descrito en el README.txt y nos sale el menu, para este ejemplo se usan 16 individuos y 10 generaciones.



```
Terminal - colin@Manjaro-de-Colin-42:~/Documents/Escuela/Algoritmos Geneticos/Practica 3 AG Colin_Varela
File Edit View Terminal Tabs Help

[colin@Manjaro-de-Colin-42 Practica 3 AG Colin_Varela]$ make
mkdir -p bin
mkdir -p release
mkdir -p release/include
cp -r plot-sdl-lib/plotsdl release/include/
ar rcs release/libplotsdl.a bin/plot.o bin/linked_list.o
gcc -Irelease/include -Irelease plot-sdl-test/main.c -O3 -g -Werror -o release/p
lot -lplotsdl -lm -lSDL2 -lSDL2_ttf
^[[A[colin@Manjaro-de-Colin-42 Practica 3 AG Colin_Varela]$ ./release/plot ./plo
t-test/opensans.ttf

Practica 3: Seleccion por ruleta
Alumno: Alejandro Colin Varela

Indique el numero de individuos (Numero par) [2-32] : 16

Indique un numero de generaciones [1-15] : 10
```



Aquí vamos las dos ultimas generaciones

```
Terminal - colin@Manjaro-de-Colin-42:~/Documents/Escuela/Algoritmos Geneticos/Practica 3 AG Colin_Vare
File Edit View Terminal Tabs Help

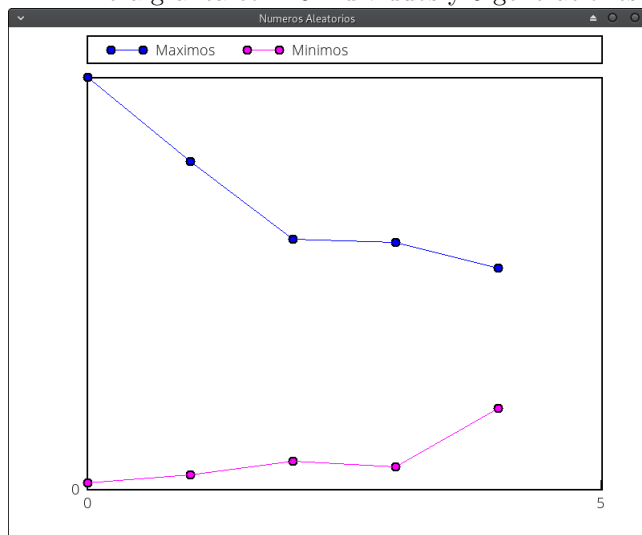
Generacion n°9 :

I[1] = 1 1 0 1 1
I[2] = 1 1 1 0 1
I[3] = 1 1 0 1 1
I[4] = 1 0 1 1 1
I[5] = 1 1 0 1 1
I[6] = 1 1 1 1 1
I[7] = 1 1 0 1 1
I[8] = 1 1 0 1 1
I[9] = 1 1 1 0 1
I[10] = 1 1 1 0 1
I[11] = 1 0 0 1 0
I[12] = 1 1 1 1 1
I[13] = 1 1 0 1 1
I[14] = 1 0 0 1 1
I[15] = 1 0 0 1 1
I[16] = 1 1 1 1 1

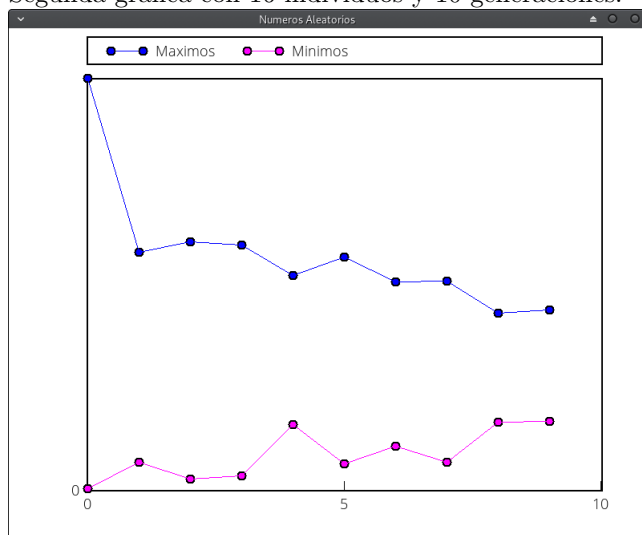
Generacion n°10 :

I[1] = 1 1 1 1 1
I[2] = 1 1 1 0 1
I[3] = 1 1 1 1 1
I[4] = 1 1 0 1 1
I[5] = 1 0 0 1 1
I[6] = 1 1 0 1 1
I[7] = 1 0 1 1 1
I[8] = 1 1 0 1 1
I[9] = 1 1 1 1 1
I[10] = 1 1 1 0 1
I[11] = 1 0 0 1 1
I[12] = 1 1 1 1 1
I[13] = 1 1 0 1 1
```

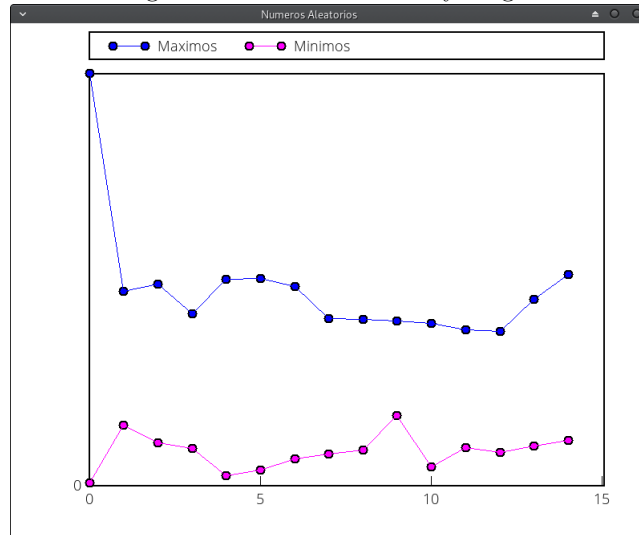
Primera gráfica con 16 individuos y 5 generaciones:



Segunda gráfica con 16 individuos y 10 generaciones:



Tercera gráfica con 16 individuos y 15 generaciones:



## 4 Conclusiones

La practica nos permitió experimentar por primera vez con Algoritmos Genéticos siendo esta ocasión siguiendo el metodo de la Ruleta, dejandonos con ganas de descubrir más formas de seleccónar los individuos.