Instituto Politécnico Nacional

Escuela Superior de Cómputo

Genetic Algorithms

Reporte Cuarta Práctica:
“Selección de individuos por el método de Torneo”

Grupo 3CM5
Romero Godinez José Enrique

Introducción:

El método de selección por torneo tiene dos variantes, el torneo determinístico y el probabilístico, el funcionamiento de este algoritmo de selección consiste en barajear a los individuos p veces hasta conseguir n seleccionados, en cada barajeo se eligen al azar dos individuos y se comparan en aptitud, es aquí donde los métodos difieren, en el caso del torneo determinístico se escoge siempre al individuo más apto, mientras que en el probabilístico se utiliza un elemento extra al azar, por ejemplo el lanzamiento de una moneada, el cuál dicta si se escoge al más apto o al menos apto.[1]
El método de cruza por punto de cruza consiste en intercambiar determinados alelos desde un punto en común entre los padres para dar paso a su descendencia que tiene estas características.
El método de mutación de cambio de un bit, permite cambiar el valor de un alelo aleatorio en un cierto porcentaje de la población, este cambio puede volver al individuo más o menos apto, según sea el caso.

Objetivo:

Desarrollar un programa en lenguaje c que haga un experimento de selección con 10,30,50 y 100 generaciones de individuos utilizando el método de selección por torneo, cruza por punto de cruza y mutación de cambio de bit al 30% de la población.

Solución:

Dado que el lenguaje c no permite de manera nativa las interfaces gráficas se hará uso del programa gnuplot[2] de linux, el cuál será invocado por un programa escrito en lenguaje c que será el encargado de llevar a cabo la selección, cruza y mutación de los individuos para finalmente graficar a los individuos más y menos aptos de cada generación.

Implementación:

Codigo del programa en c:

```
#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <math.h>

void generaIndividuos(unsigned char*);

void seleccionaPadres(unsigned char*,float*);

void calculaDatos(unsigned char*,float*,float*);

int dimeMaximo(unsigned char*);

int dimeMinimo(unsigned char*);

void cruzar(unsigned char*);

void mutar(unsigned char*);

int main()
```

```
{
	unsigned char* individuos=(unsigned char*)malloc(sizeof(unsigned char)*16);
	float* aptitud=(float*)malloc(sizeof(float)*16);
	float* probabilidad=(float*)malloc(sizeof(float)*16);
	int mayor,menor;
	float mayores[30];
	float menores[30];
	generaIndividuos(individuos);
	FILE * archivoMaximos = fopen("puntosMaximos.txt", "w");
	FILE * archivoMinimos = fopen("puntosMinimos.txt", "w");
	for(int i=0;i<30;i++)
	{
		calculaDatos(individuos,aptitud,probabilidad);
		mayor=dimeMaximo(individuos);
		mayores[i]=probabilidad[mayor];
		menor=dimeMinimo(individuos);
		menores[i]=probabilidad[menor];
		//Aqui se guardan los datos a graficar
		fprintf(archivoMaximos, "%d %f \n",i+1,mayores[i]);
		fprintf(archivoMinimos, "%d %f\n",i+1,menores[i]);
		//Guardar datos a graficar
		seleccionaPadres(individuos,probabilidad);
		cruzar(individuos);
		mutar(individuos);
	}
	char * configGnuplot[] = {"set title \"Histograma\"",
			"set ylabel \"----Aptitud--->\"",
			"set xlabel \"----Generaciones--->\"",
			"set style data histogram",
```

```
                        "set style histogram cluster gap 1",
                            "plot \"puntosMaximos.txt\" using 1:2 with linespoints ,\"puntosMinimos.txt\" using 1:2 with linespoints"
                        };

    FILE * ventanaGnuplot = popen ("gnuplot -persist", "w");
    // Ejecuta los comandos de configGnuPlot 1 por 1
    for (int k=0;k<6;k++){
      fprintf(ventanaGnuplot, "%s \n", configGnuplot[k]);
    }
    fclose(archivoMinimos);
    fclose(archivoMaximos);
    //free(individuos);
    free(aptitud);
    free(probabilidad);
        return 0;
}
void calculaDatos(unsigned char* individuos,float* aptitud,float* probabilidad)
{
        float suma=0;
        double radianes=3.1416/180;
        for(int i=0;i<16;i++)
        {
                aptitud[i]=(float)abs((individuos[i]-5)/(2 + sin((double)individuos[i]*radianes)));
                suma+=(float)aptitud[i];
        }

        for(int i=0;i<16;i++)
        {
```

```
                probabilidad[i]=((float)aptitud[i]/suma);
        }
}
void seleccionaPadres(unsigned char* individuos, float* probabilidad)
{
        unsigned char* seleccion=(unsigned char*)malloc(sizeof(unsigned char)*16);
        long ltime;
        int stime,barajeo1,barajeo2;
        ltime=time(NULL);
        stime=(unsigned)ltime/2;
        srand(stime);
        for(int i=0;i<16;i++)
        {
                barajeo1=rand()%(15);
                barajeo2=rand()%(15);
                if(individuos[barajeo1]<=individuos[barajeo2])
                {
                        seleccion[i]=individuos[barajeo2];
                }else{
                        seleccion[i]=individuos[barajeo1];
                }
        }

        free(individuos);
        individuos=seleccion;
}

void generaIndividuos(unsigned char* individuos)
{
```

```
	long ltime;
	ltime=time(NULL);
	int a,stime;
	stime=(unsigned)ltime/2;
	srand(stime);
	for(int i=0;i<16;i++)
	{
		individuos[i]=(unsigned char)(1+rand()%(32-1));
	}
}
int dimeMaximo(unsigned char* individuos)
{
	int aux=0;
	for(int i=1;i<16;i++)
	{
		if(individuos[aux]<individuos[i])
		{
			aux=i;
		}
	}
	return aux;
}
int dimeMinimo(unsigned char* individuos)
{
	int aux=0;
	for(int i=1;i<16;i++)
	{
		if(individuos[aux]>individuos[i])
		{
```

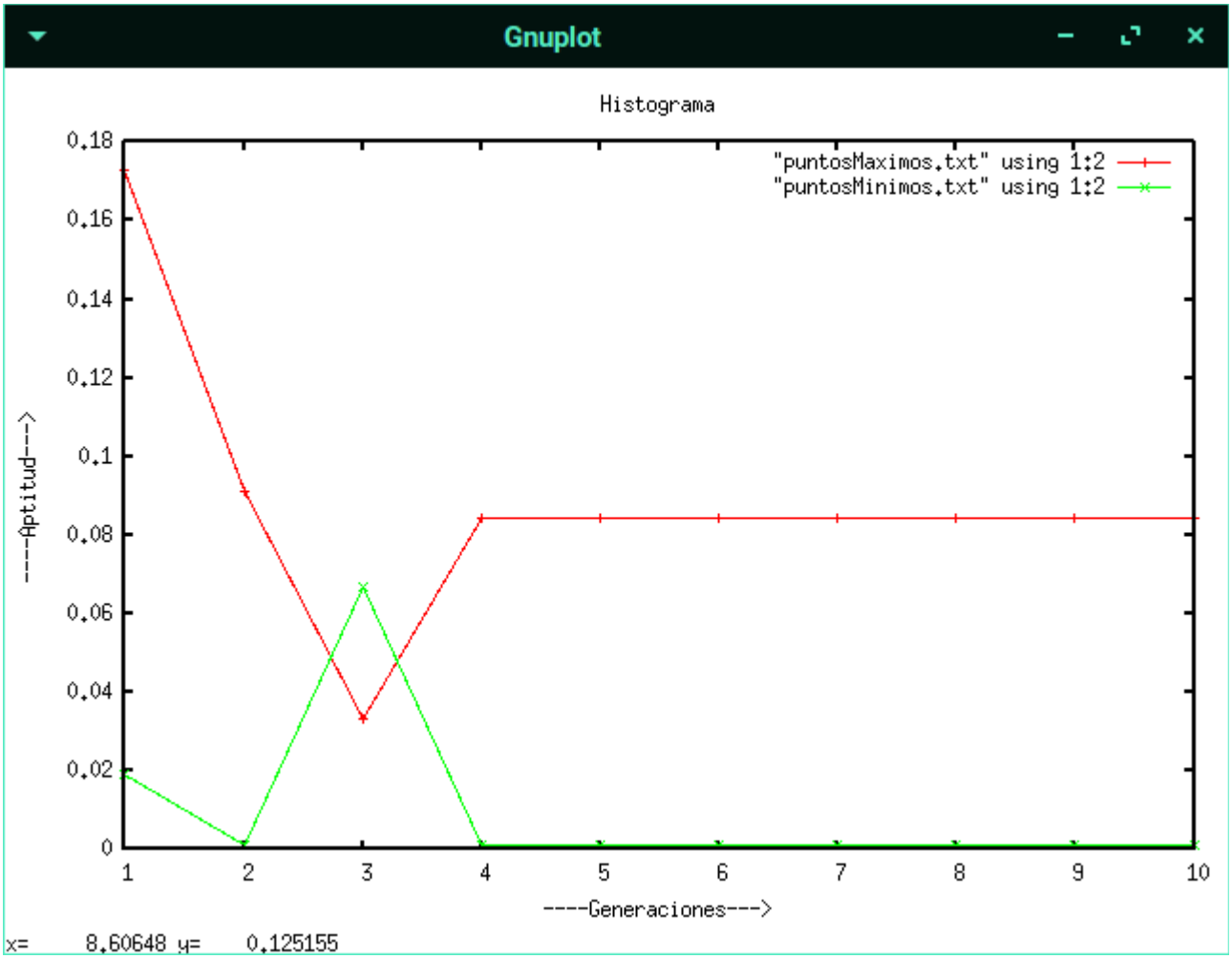
```
                aux=i;
            }
        }
        return aux;
}
void cruzar(unsigned char* individuos)
{
        long ltime;
        ltime=time(NULL);
        int stime;
        stime=(unsigned)ltime/2;
        srand(stime);
        int puntoCruza=0;
        unsigned char* descendencia=(unsigned char*)malloc(sizeof(unsigned char)*16);
        int hijo=0;
        unsigned char bits,valor;
        puntoCruza=rand()%(5);
        for(int i=0;i<16;i++)
        {
            bits=((unsigned char)(pow((double)2,(double)puntoCruza)));
            if(hijo==0)
            {
                valor=bits&individuos[i];
                descendencia[i]=(individuos[i+1]-bits)+valor;
                hijo++;
            }
            if(hijo==1)
            {
                valor=bits&individuos[i];
```

```
                descendencia[i]=(individuos[i-1]-bits)+valor;
                hijo=0;
                puntoCruza=rand()%(5);
            }
        }
        free(individuos);
        individuos=descendencia;
}
void mutar(unsigned char* individuos)
{
        long ltime;
        unsigned char bits,valor;
        int stime;
        int mutados[4],puntoCruza;

        ltime=time(NULL);
        stime=(unsigned)ltime/2;
        srand(stime);
        mutados[0]=rand()%(15);

        mutados[1]=rand()%(15);

        mutados[2]=rand()%(15);
        mutados[3]=rand()%(15);

        puntoCruza=rand()%(5);
        bits=((unsigned char)(pow((double)2,(double)puntoCruza)));
        for(int i=0;i<4;i++)
        {
```

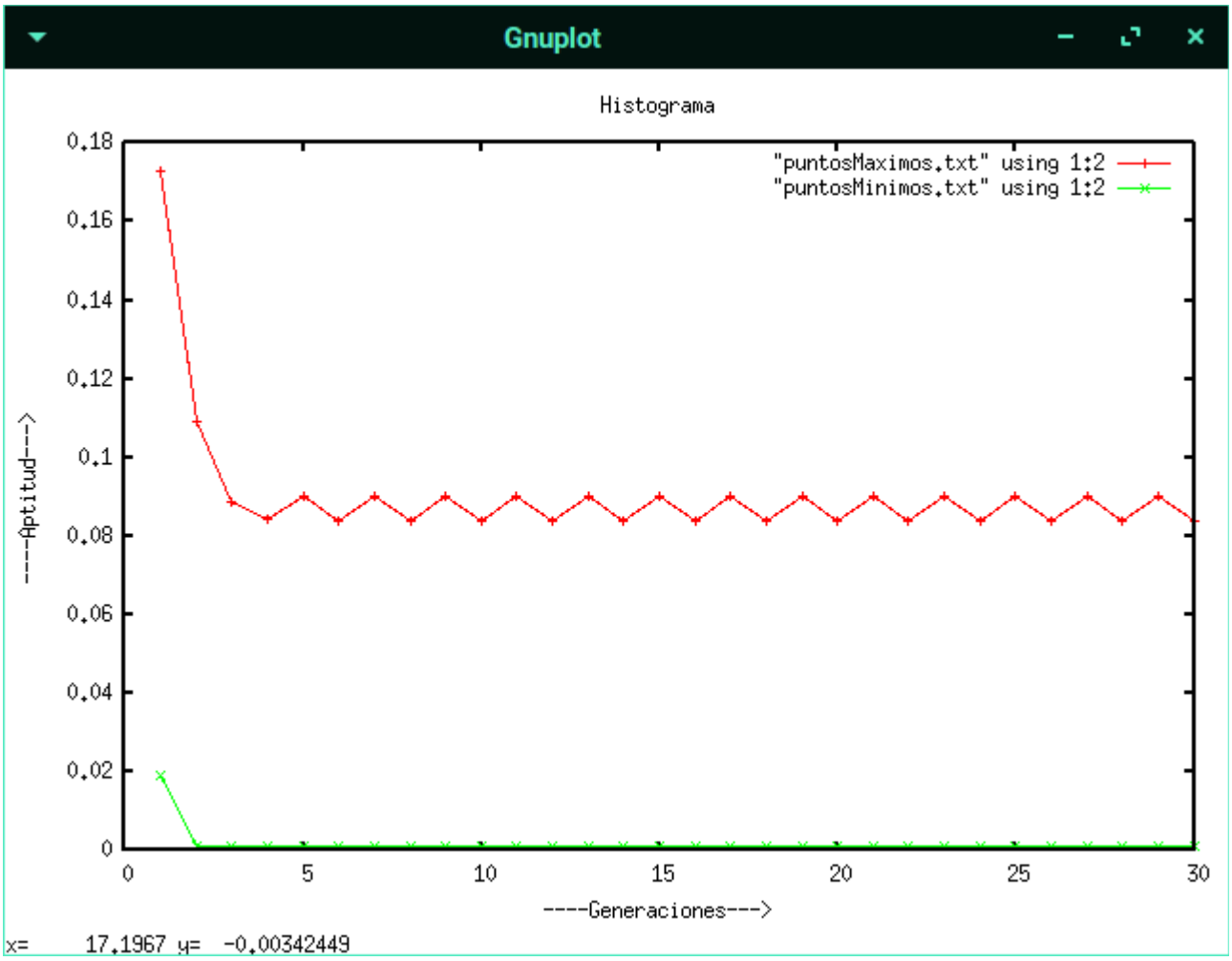
```
            valor=bits&individuos[mutados[i]];
            if(valor==0)
            {
                individuos[mutados[i]]=individuos[mutados[i]]+bits;
            }else{
                individuos[mutados[i]]=individuos[mutados[i]]-bits;
            }
        }
}
```

Ejecucion del programa:

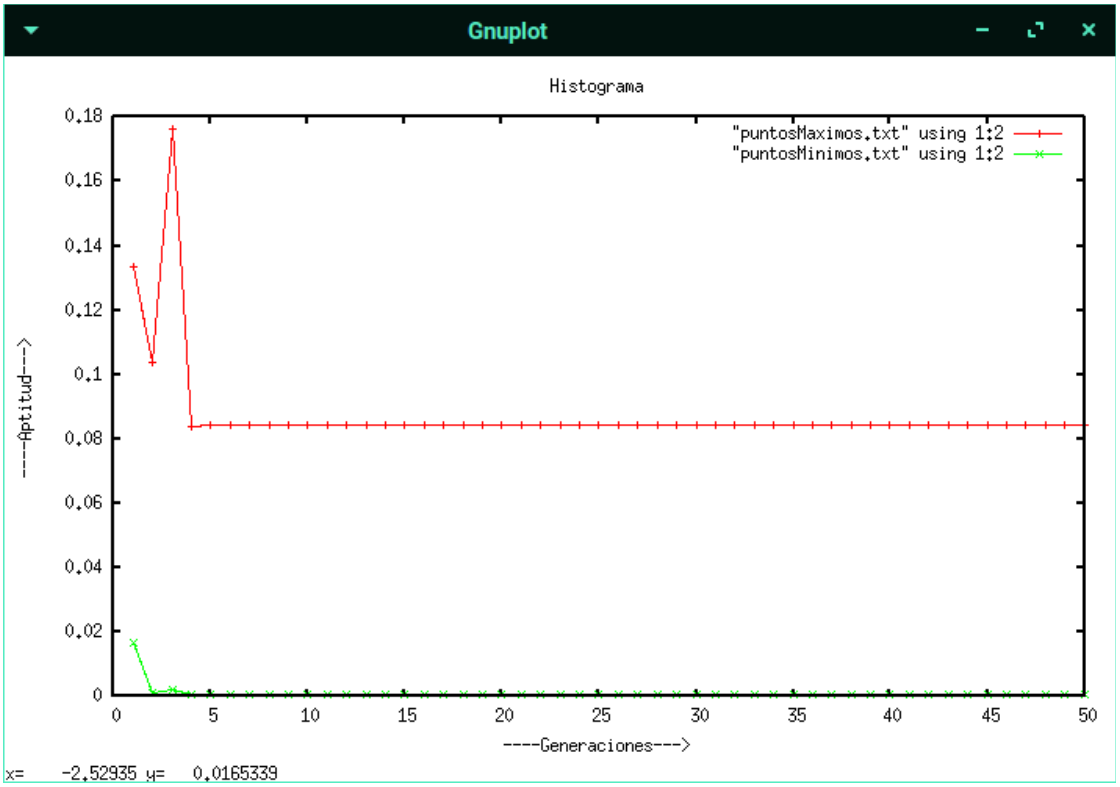
Experimento con 10 generaciones de individuos



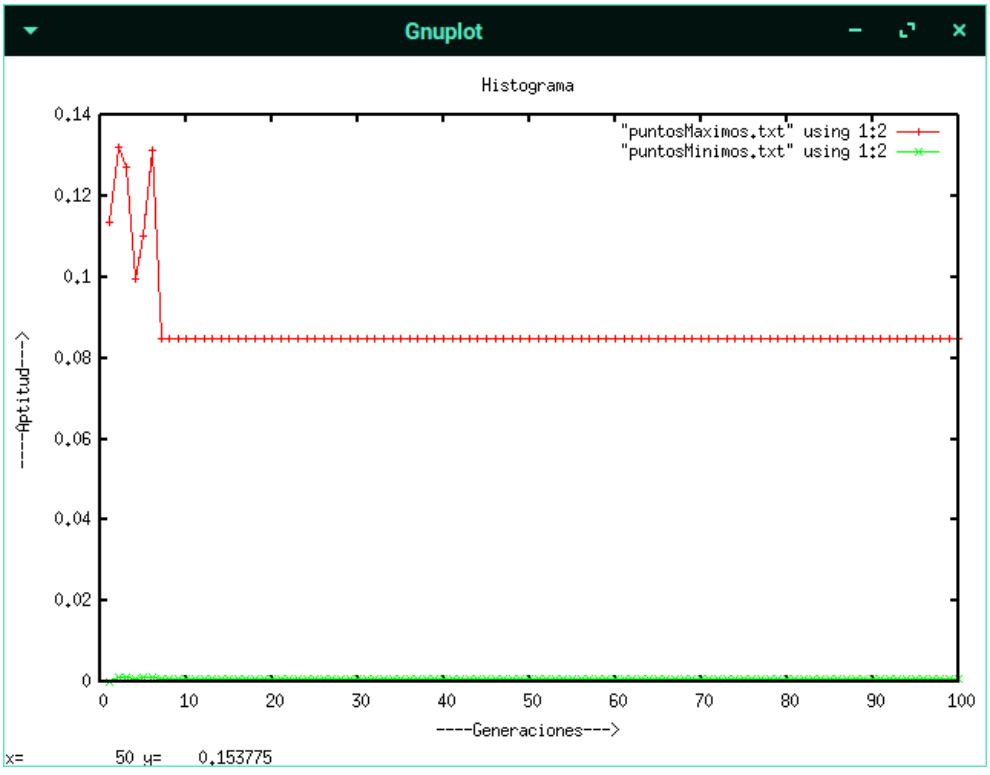
Experimento con 30 generaciones:



Experimento con 50 generaciones:



Experinento con 100 generaciones:



Conclusiones:

El método de selección por torneo es uno de muchos algoritmos de selección que pueden ser aplicado, sólo que esta vez tenemos dos variantes, la variante determinística nos asegura que siempre escogeremos al individuo más apto de cada barajeo, por lo que nuestra población covergerá a su máximo en poco tiempo, caso contrario con el probabilístico ya que la selección puede variar de un momento a otro haciendo la convergencia una tarea un poco más tardada.

Referencias:

[1]http://sabia.tic.udc.es/mgestal/cv/AAGGtutorial/node11.html