

Instituto Politécnico Nacional
Escuela Superior de Cómputo



Practica 4:

Selección por Torneo

Genethic Algorithms

Álvarez González Oscar

Prof. Morales Guitaron Sandra Luz

Grupo: 3CM5

Índex

Contenido

Introducción:

Desarrollo:

Resultados:

Conclusiones:

Introducción:

Los métodos de selección proporcional antes descritos requieren de dos pasos a través de toda la población en cada generación:

1. Calcular la aptitud media (y, si se usa escalamiento sigma, la desviación estándar).
2. Calcular el valor esperado de cada individuo.

El uso de jerarquías requiere que se ordene toda la población (una operación cuyo costo puede volverse significativo en poblaciones grandes).

La selección mediante torneo es similar a la de jerarquías en términos de la presión de selección, pero es computacionalmente más adecuada para implementarse en paralelo.

Esta técnica fue propuesta por Wetzel y estudiada en la tesis doctoral de Brindle.

La idea básica del método es seleccionar con base en comparaciones directas de los individuos.

Hay 2 versiones de la selección mediante torneo:

- Determinística
- Probabilística

El algoritmo de la versión determinística es el siguiente:

- Barajar los individuos de la población.
- Escoger un número p de individuos (típicamente 2).
- Compararlos con base en su aptitud.
- El ganador del “torneo” es el individuo más apto.
- Debe barajarse la población un total de p veces para seleccionar N padres (donde N es el tamaño de la población).

Orden	Aptitud	Barajar	Ganadores
(1)	254	(2)	
(2)	47	(6)	(6)
(3)	457	(1)	
(4)	194	(3)	(3)
(5)	85	(5)	
(6)	310	(4)	(4)

Barajar	Ganadores
(4)	
(1)	(1)
(6)	
(5)	(6)
(2)	
(3)	(3)

Padres:

(6) y (1), (3) y (6), (4) y (3)

Desarrollo:

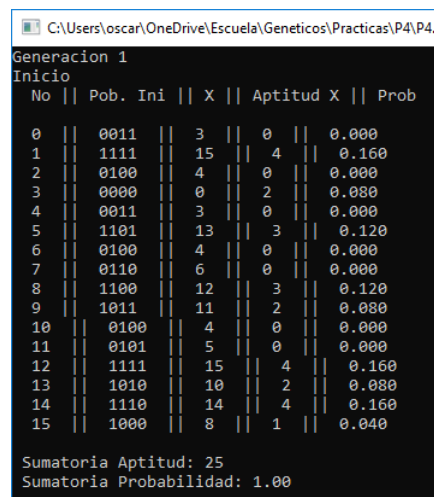
Para el desarrollo de esta práctica se utiliza lo mismo métodos de la practica anterior, cruza de un punto y mutación por bit de 30% y ahora se cambia el tipo de selección que en este caso es la selección por torneo.

Las funciones que utilice en este proyecto son:

```
void binario(int gen);
void inicializacion(int gen,unsigned char arrg[16][4]);
void seleccionTorneo(int gen, unsigned char arrg[16][4],float *Prob,int *dec,int *apt);
void cruza(int gen,unsigned char sel[16][4]);
void mutacion(int gen,unsigned char cruza[16][4]);
void grafica(float *Max, float *Min);
```

Como se ve son las mismas funciones de la practica anterior quitando por supuesto la función de seleccionTorneo.

En esta práctica ahora el son solo 16 individuos con 4 bits cada uno por lo que el máximo número que se puede formar es 15 y el mínimo es 0.



No	Pob. Ini	X	Aptitud X	Prob
0	0011	3	0	0.000
1	1111	15	4	0.160
2	0100	4	0	0.000
3	0000	0	2	0.080
4	0011	3	0	0.000
5	1101	13	3	0.120
6	0100	4	0	0.000
7	0110	6	0	0.000
8	1100	12	3	0.120
9	1011	11	2	0.080
10	0100	4	0	0.000
11	0101	5	0	0.000
12	1111	15	4	0.160
13	1010	10	2	0.080
14	1110	14	4	0.160
15	1000	8	1	0.040

Sumatoria Aptitud: 25
Sumatoria Probabilidad: 1.00

También en esta practica la $f(x)$ de la aptitud cambio en:

$$f(x) = ABS \left| \frac{x - 5}{2 + Sen(x)} \right|$$

Bueno ahora explicare la función de seleccionTorneo lo que hará es tener un numero "r" aleatorio de rango [0.0 a 1.0]eso lo que me dice es si sale un numero de 0.1 a 0.7 significa que se tomara el individuo mas apto sin en cambio sale un numero de 0.8 a 0.9 seleccionara al individuo menos apto, los individuos también son escogidos al azar y pelean para ver quien sigue en ejecución.

```

r=0.2
x=15
y=0
1000

```

Ejemplo 1

```

r=0.9
x=8
y=5
1100

```

Ejemplo 2

En el primer ejemplo se ve que cumple con la condición ya que al estar en el rango de (.1 a .7) se toma el individuo más apto, de los escogidos y después se escoge dos individuos al azar en este caso se escogió el individuo 15(1000) y 0(0011) y al ser el individuo 15 mas apto es el que se escoge.

En el segundo ejemplo se ve que no cumple la condición por lo cual se tomara el individuo menos apto, los individuos escogidos al azar fueron el 8(1100) y el 5(1101), al ser menos apto el individuo 8 ese es el que se escoge.

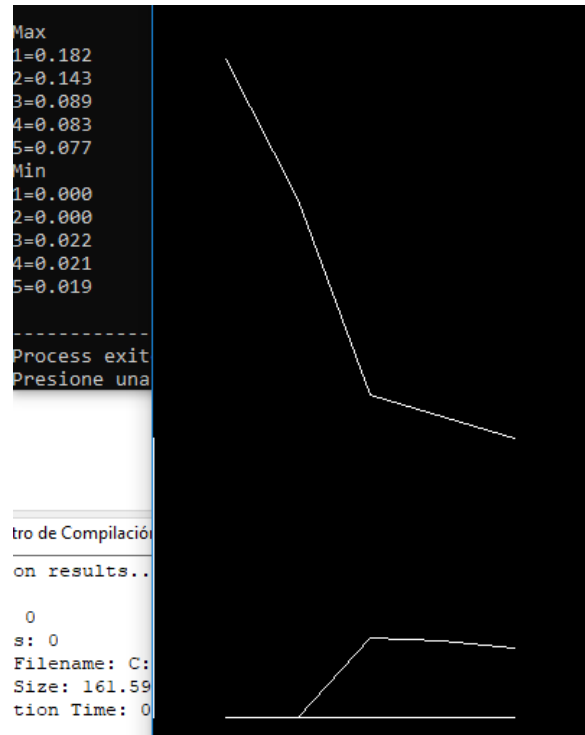
La mutación es la misma de la practica anterior solo que ahora se pide el 30% sobre los individuos por lo cual se hacen 5 cambios de bits.

Mutacion						
No	Desc	Mutacion	X	Aptitud X	Prob	
0	0101	0101	5	0	0.000	
1	0101	1101	13	3	0.081	
2	0001	0001	1	1	0.027	
3	1101	1101	13	3	0.081	
4	0011	0011	3	0	0.000	
5	0111	0111	7	0	0.000	
6	1101	1101	13	3	0.081	
7	1101	1101	13	3	0.081	
8	1101	1101	13	3	0.081	
9	1111	1111	15	4	0.108	
10	1101	1111	15	4	0.108	
11	1101	1101	13	3	0.081	
12	1011	1011	11	2	0.054	
13	1110	1110	14	4	0.108	
14	0001	0111	7	0	0.000	
15	0111	1111	15	4	0.108	

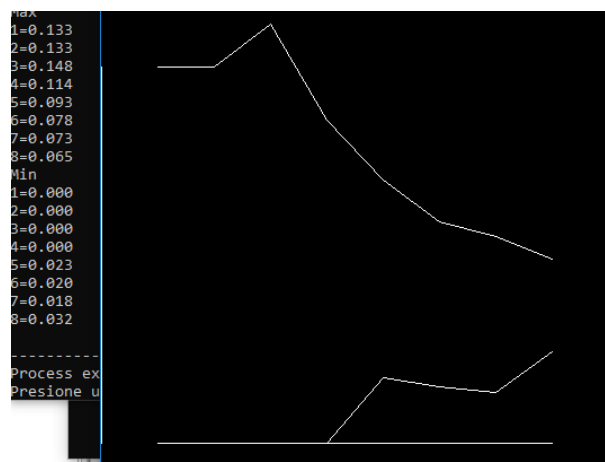
Resultados:

En esta practica se nos pide que hagamos corrimientos del programa con 5, 10, 15 y 30 generaciones, se graficara los Máximos y los mínimos de cada generación con base en lo sacado en la probabilidad.

Para 5 generaciones:



Para 8 generaciones:



No se que paso puede que sea mi computadora ya que es bastante vieja pero al principio incluso ayer pude correr hasta 40-50 generaciones ahora estoy teniendo problemas para que corra mínimo 10 y lo máximo que pudo hacer fueron 8 no se que pudo haber pasado.

Conclusión:

Esta práctica me costó un poco más que la anterior en cuanto a la codificación del algoritmo de selección, me frustró un poco que no se que halla pasado que no me deja hacer mínimo 10 generaciones no lo comprendo, hablando de los resultados se ve que en este algoritmo tarda mas es subir el mínimo pero cuando sube se ve que convergiría aún más rápido que la selección por ruleta.