



**INSTITUTO POLITÉCNICO NACIONAL  
ESCUELA SUPERIOR DE CÓMPUTO**



**Algoritmos genéticos**

**“Cruza”**

**Alumno:**

**Navarro Pérez Rafael**

# Índex

## Contenido

Introducción: .....	1
Desarrollo: .....	2
Resultados:.....	3
Conclusiones: .....	5

## Introducción:

En los sistemas biológicos, la cruza es un proceso complejo que ocurre entre parejas de cromosomas. Estos cromosomas se alinean, luego se fraccionan en ciertas partes y posteriormente intercambian fragmentos entre sí.

En computación evolutiva se simula la cruza intercambiando segmentos de cadenas lineales de longitud fija (los cromosomas).

Aunque las técnicas de cruza básicas suelen aplicarse a la representación binaria, éstas son generalizables a alfabetos de cardinalidad mayor, si bien en algunos casos requieren de ciertas modificaciones.

Comenzaremos por revisar las tres técnicas básicas de cruza:

- Cruza de un punto
- Cruza de dos puntos
- Cruza uniforme

### Cruza de un punto

Esta técnica fue propuesta por Holland [127], y fue muy popular durante muchos años. Hoy en día, sin embargo, no suele usarse mucho en la práctica debido a sus inconvenientes. Puede demostrarse, por ejemplo, que hay varios esquemas que no pueden formarse bajo esta técnica de cruza. Considere, por ejemplo, los esquemas siguientes:

$$H1 = 11*****1$$
$$H2 = ****11**$$

Si aplicamos cruza de un punto a estos 2 esquemas, no hay manera de formar una instancia del esquema:

$$H = 11**11*1$$

### Cruza de dos puntos

De Jong fue el primero en implementar una cruza de  $n$  puntos, como una generalización de la cruza de un punto. El valor  $n = 2$  es el que minimiza los efectos disruptivos (o destructivos) de la cruza y de ahí que sea usado con gran frecuencia. No existe consenso en torno al uso de valores para  $n$  que sean mayores o iguales a 3. Los estudios empíricos al respecto proporcionan resultados que no resultan concluyentes respecto a las ventajas o desventajas de usar dichos valores. En general, sin embargo, es aceptado que la cruza de dos puntos es mejor que la cruza de un punto. Asimismo, el incrementar el valor de  $n$  se asocia con un mayor efecto disruptivo de la cruza.

### Cruza uniforme

Esta técnica fue propuesta originalmente por Akey [1], aunque se le suele atribuir a Syswerda .

En este caso, se trata de una cruce de n puntos, pero en la cual el número de puntos de cruce no se fija previamente. La cruce uniforme tiene un mayor efecto disruptivo que cualquiera de las 2 cruces anteriores. A fin de evitar un efecto excesivamente disruptivo, suele usarse con  $P_c = 0.5$ . Algunos investigadores, sin embargo, sugieren usar valores más pequeños de  $P_c$ . Cuando se usa  $P_c = 0.5$ , hay una alta probabilidad de que todo tipo de cadena binaria de longitud L sea generada como máscara de copiado de bits.

### Desarrollo:

Para esta práctica se ocupó el IDE Code Blocks.

Y se realizó en C++, lo cual se pudo ocupar la librería de bitset para crear los individuos y ahorrar recursos con ellos. Y para poder realizar las cruces se ocupó la operación AND para obtener los valores deseados de cada individuo y así poder representar la cruce deseada.

```
1. void IniciarInd(bitset<BIT_IND> *individuo){
2.
3.     *individuo = rand() % MAX_VAL;
4.
5. }
6.
7. bitset<BIT_IND> cruzaunpunto(bitset<BIT_IND> &p1, bitset<BIT_IND> &p2) {
8.     bitset<BIT_IND> aux = p1;
9.     for (int i = 0; i <= PUNTO_CRUZA; i++)
10.    {
11.        aux.set(PUNTO_CRUZA - i, p2[PUNTO_CRUZA - i]);
12.    }
13.    return aux;
14. }
15. bitset<BIT_IND> cruzaDP(bitset<BIT_IND> &individuoA, bitset<BIT_IND> &individuoB) {
16.     bitset<BIT_IND> aux1;
17.     bitset<BIT_IND> aux2;
18.     bitset<BIT_IND> res;
19.     bitset<BIT_IND> and1 (string("1111000011"));
20.     bitset<BIT_IND> and2 (string("0000111100"));
21.     aux1 = individuoA & and1;
22.     aux2 = individuoB & and2;
23.     int total = aux1.to_ulong() + aux2.to_ulong();
24.     res = total;
25.     return res;
26. }
27. bitset<BIT_IND> cruzaUni(bitset<BIT_IND> &individuoA, bitset<BIT_IND> &individuoB) {
28.     bitset<BIT_IND> aux1;
29.     bitset<BIT_IND> aux2;
30.     bitset<BIT_IND> res;
31.     bitset<BIT_IND> and1 (string("1011001011"));
32.     bitset<BIT_IND> and2 (string("0100110100"));
33.     aux1 = individuoA & and1;
34.     aux2 = individuoB & and2;
35.     int total = aux1.to_ulong() + aux2.to_ulong();
36.     res = total;
37.     return res;
```

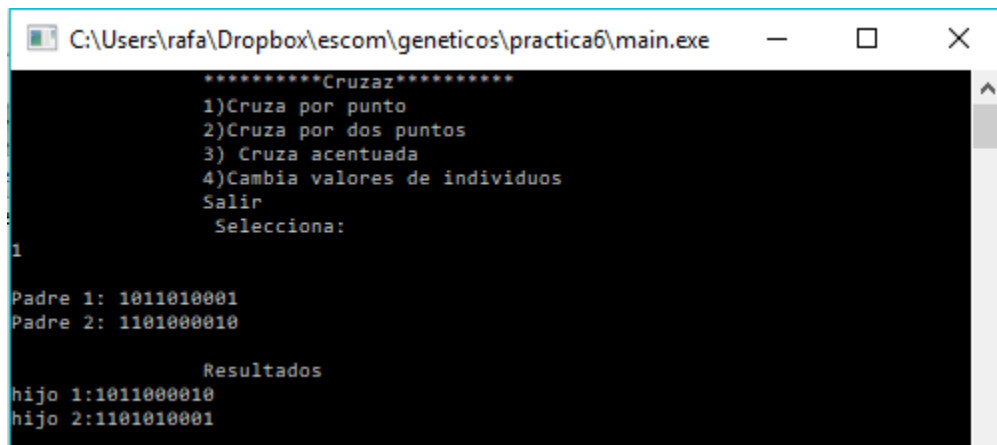
```
38. }
```

### Resultados:

En el programa desarrollado se cuenta con un menú que al ingresar un numero de los 5 posibles se realizara un tipo de crusa distinto:

1. Cruza de un punto
2. Cruza de dos puntos
3. Cruza uniforme
4. Cambiar valores de los individuos
5. Salir

Una vez seleccionada una opción se ejecutará la acción deseada si oprimimos 1 esto debe de ocurrir:

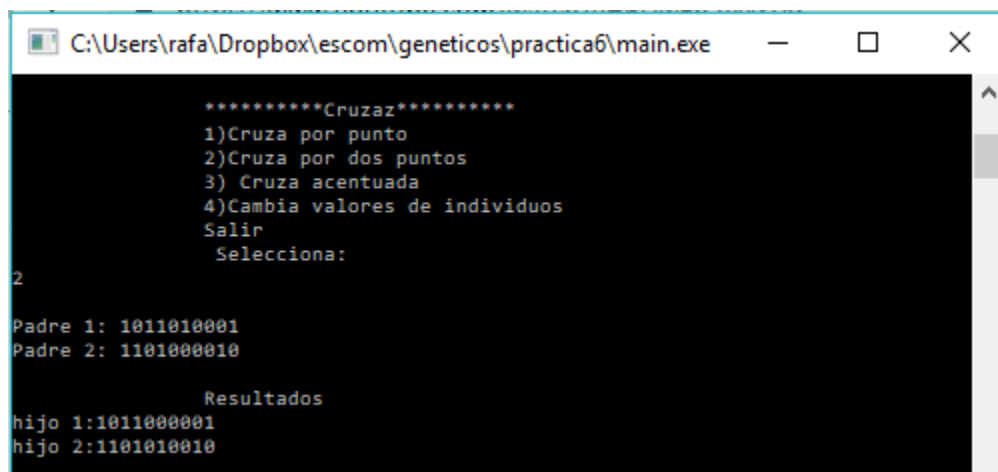


```
C:\Users\rafa\Dropbox\escom\geneticos\practica6\main.exe
*****Cruzaz*****
1)Cruza por punto
2)Cruza por dos puntos
3) Cruza acentuada
4)Cambia valores de individuos
Salir
  Selecciona:
1
Padre 1: 1011010001
Padre 2: 1101000010

      Resultados
hijo 1:1011000010
hijo 2:1101010001
```

Como se ve en la imagen se ejecuta la crusa por un punto que en este caso es 4.

Seleccionando la opción 2: Cruza por dos puntos



```
C:\Users\rafa\Dropbox\escom\geneticos\practica6\main.exe

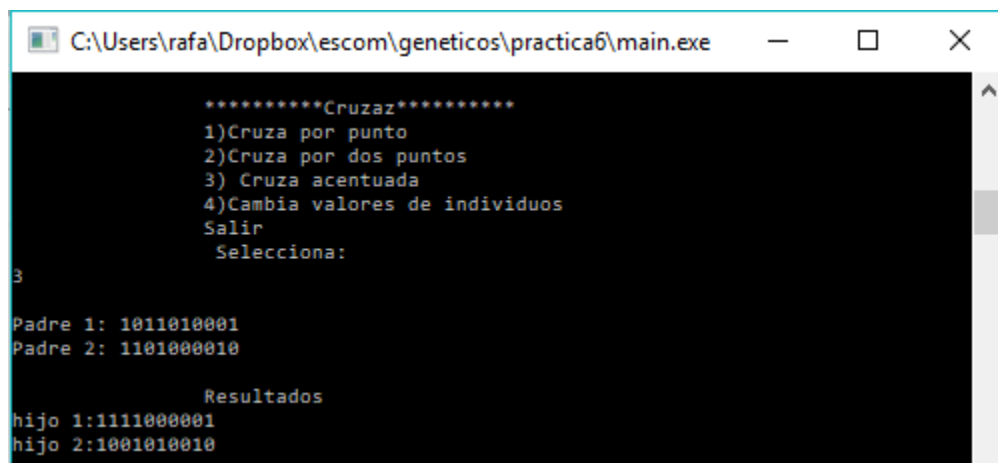
*****Cruzaz*****
1)Cruza por punto
2)Cruza por dos puntos
3) Cruza acentuada
4)Cambia valores de individuos
Salir
  Selecciona:
2

Padre 1: 1011010001
Padre 2: 1101000010

          Resultados
hijo 1:1011000001
hijo 2:1101010010
```

Se observa que son los mismos individuos pero que los hijos son completamente distintos que los de Cruza por un punto, ya que aquí hacemos operaciones AND para poder seleccionar los 2 puntos de corte.

Seleccionando la opción 3: Cruza uniforme



```
C:\Users\rafa\Dropbox\escom\geneticos\practica6\main.exe

*****Cruzaz*****
1)Cruza por punto
2)Cruza por dos puntos
3) Cruza acentuada
4)Cambia valores de individuos
Salir
  Selecciona:
3

Padre 1: 1011010001
Padre 2: 1101000010

          Resultados
hijo 1:1111000001
hijo 2:1001010010
```

De igual manera se puede observar que los hijos son completamente distintos a los de las otras cruza.

### **Conclusiones:**

Es importante la cruce ya que de ahí puede derivar a tener unos mejores individuos para cada caso o el comportamiento que tendrían la interacción de los individuos para que en próximas generaciones los hijos sean los más aptos.

Otro punto importante de la cruce es que en ocasiones el seleccionar un tipo de cruce ayuda a que nuestro programa no converja muy rápido en próximas generaciones y que se lleve un mejor control de estos.

De igual manera se puede crear tu propio método de cruce si es que no te llega a funcionar ninguno de los previos vistos, y que se adapte lo mejor posible a tu problema a resolver.