INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Genetic Algorithms

Practica 6:

Profa.: MORALES GUITRON SANDRA LUZ

Grupo: 3CM5

Alumno:

Salcedo Barrón Ruben Osmair.

MEXICO, D.F. a 25 de noviembre del 2018

Introducción

Cruza por un Punto

Esta técnica fue propuesta por Holland, 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.

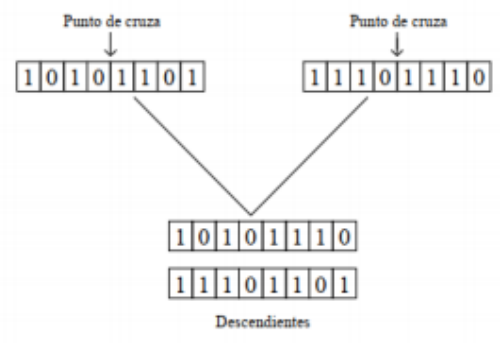


Figura 1: Cruza por un punto.

Cruza por dos Puntos

DeJong 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.

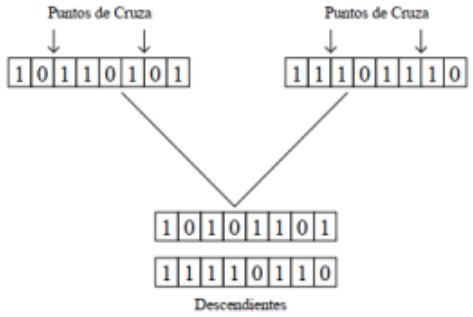
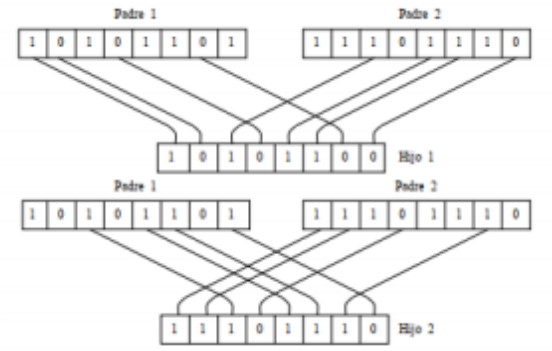


Figura 2: Cruza por dos puntos.

Cruza Uniforme

Esta técnica fue propuesta originalmente por Ackley, aunque se le suele atribuir a Syswerda. En este caso, se trata de una cruza de n puntos, pero en la cual el número de puntos de cruza no se fija previamente. La cruza uniforme tiene un mayor efecto disruptivo que cualquiera de las 2 cruzas 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 Pc.

Cuando se usa P c = 0.5, hay una alta probabilidad de que todo tipo de cadena binaria de longitud L sea generada como mascara de copiado de bits.



Cruza Acentuada

Esta técnica fue propuesta por Schaffer y Morishima, en un intento por implementar un mecanismo de autoadaptación para la generación de los patrones favorables (o sea, los buenos bloques constructores) de la cruza. En vez de calcular directamente la mascara (o patrón) de cruza, la idea es usar una cadena binaria de “marcas” para indicar la localización de los puntos de cruza. La idea fue sugerida por Holland, aunque en un sentido distinto. La información extra que genera la cruza acentuada se agrega al cromosoma de manera que el número y localizaciones de los puntos de cruza pueda ser objeto de manipulación por el AG. Por tanto, las cadenas tendrán una longitud del doble de su tamaño original. La convención que suele adoptarse es la de marcar con ‘1’ las posiciones donde hay cruza y con ‘0’ las posiciones donde no la hay. Asimismo, se suelen usar signos de admiración para facilitar la escritura de las cadenas.

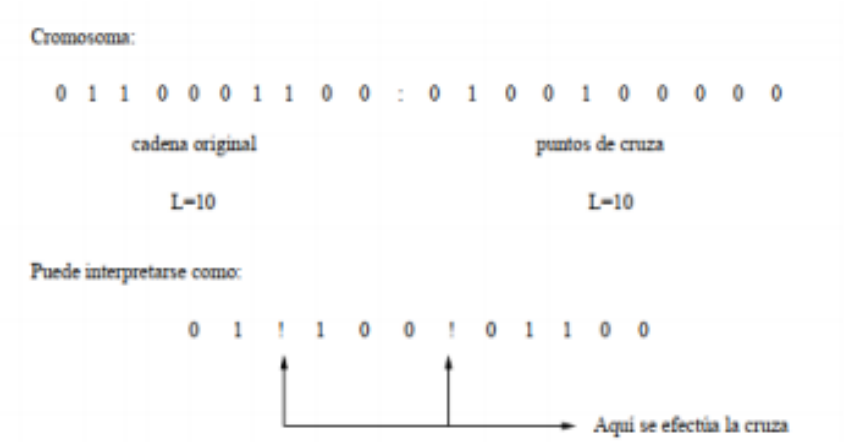
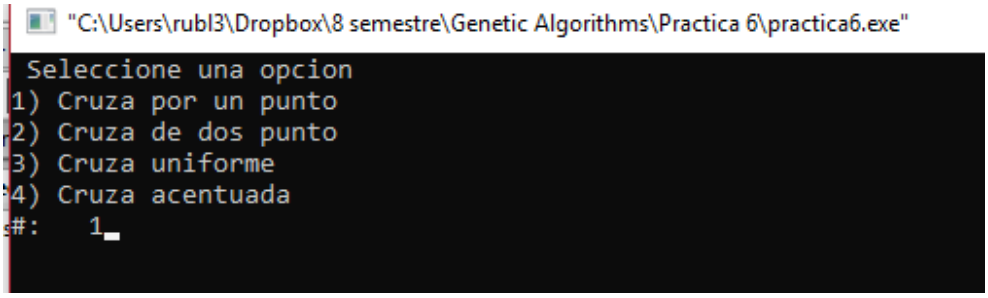
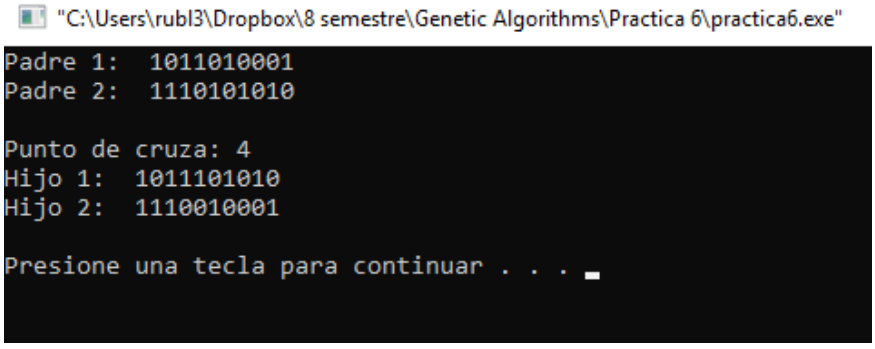


Figura 7.4: Cruza Acentuada.

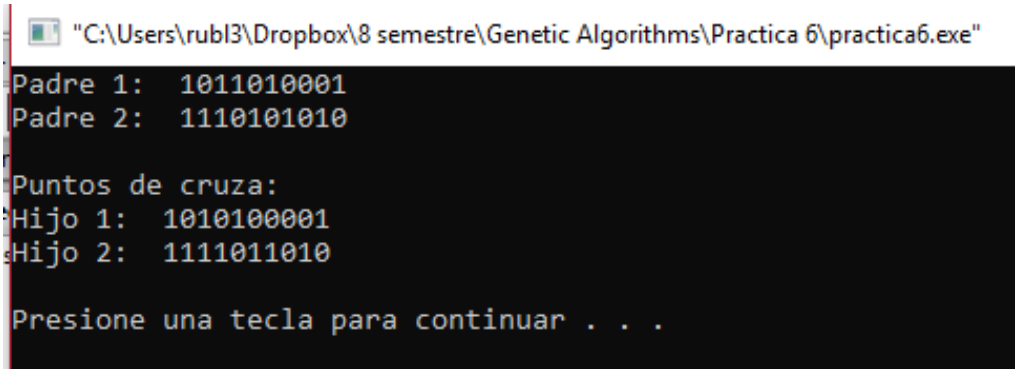
Desarrollo



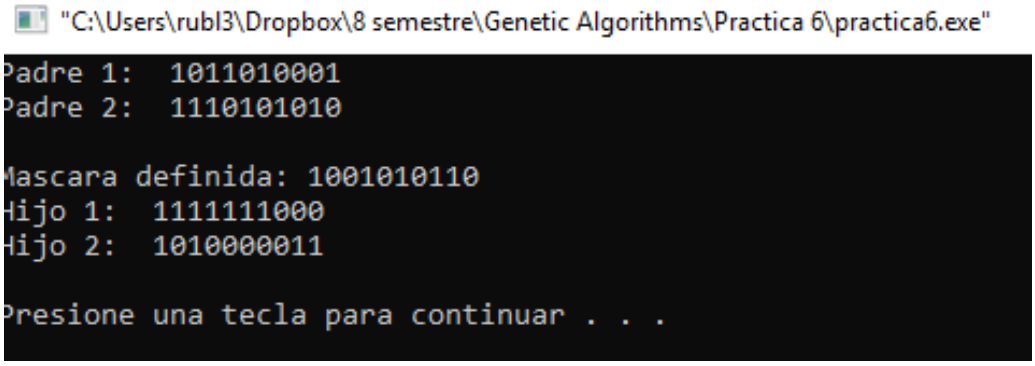
Cruza un punto



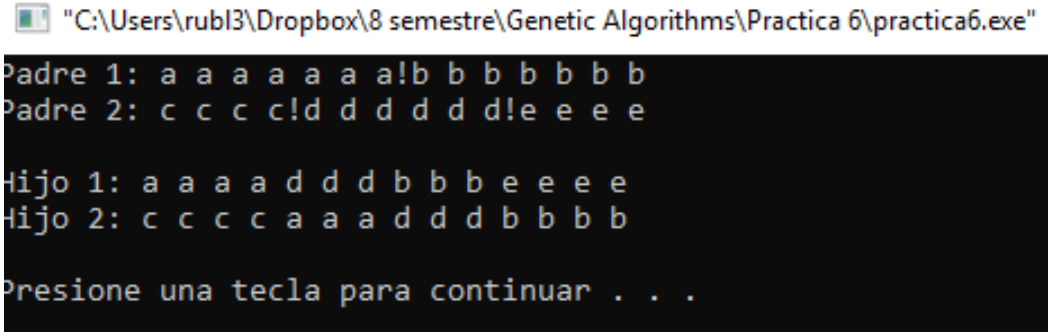
Cruza dos puntos



Cruza uniforme



Cruza acentuada



Conclusiones

Como podemos notar con las diferentes maneras de cruzas de cromosomas podemos utilizar algunas de estas para simular individuos par aun determinado fin, depende que se esté buscando se puede aplicar uno o varios tipos de cruza.

Código

```
#include <iostream>
#include <bitset>
#include <time.h>
#include <string>
#define PUNTO_CRUZA 2

using namespace std;
void llenado(bitset<10> *set);
char * cruzaAcentuada(char * padre1, char * padre2);
//cruzas
bitset<10> UnPunto(bitset<10> &padre1, bitset<10> &padre2, int
punto_cruza);
bitset<10> DosPunto(bitset<10> &padre1, bitset<10> &padre2);
bitset<10> Uniforme(bitset<10> &padre1, bitset<10> &padre2);

int main(int argc, char const *argv[])
{
      bitset<10> padre1;
      bitset<10> padre2;
      bitset<10> hijo1;
      bitset<10> hijo2;
      char padre1_2 [29] = {'a',' ','a',' ','a',' ','a',' ','a',' ','a','
','a','!','b',' ','b',' ','b',' ','b',' ','b',' ','b',' '};
      char padre2_2 [29] = {'c',' ','c',' ','c',' ','c','!','d',' ','d','
','d',' ','d',' ','d',' ','d','!','e',' ','e',' ','e',' ','e',' '};
      char *hijo1_2 = (char *)malloc(sizeof(char)*29);
      char *hijo2_2 = (char *)malloc(sizeof(char)*29);
      srand (time(NULL));
      llenado(&padre1);
      llenado(&padre2);
      int op = 0;
      while(op != 5)
      {
          system("CLS");
        cout << " Seleccione una opcion" <<'\n';
            cout << "1) Cruza por un punto" << endl;
            cout << "2) Cruza de dos punto" << endl;
            cout << "3) Cruza uniforme" << endl;
            cout << "4) Cruza acentuada" << endl;
            cout << "#:   ";
            cin >> op;
            switch(op){
                  case 1:
                      system("CLS");
                        cout << "Padre 1: ";
                        std::cout << " " << padre1 << '\n';
                        cout << "Padre 2: ";
                        std::cout << " " << padre2 << '\n';
                        cout << endl;
                        cout << "Punto de cruza: 4" << endl;
                        hijo1 = UnPunto(padre1, padre2, 5);
                        hijo2 = UnPunto(padre2, padre1, 5);
                        cout << "Hijo 1: ";
                        std::cout << " " << hijo1 << '\n';
```

```
      cout << "Hijo 2: ";
      std::cout << " " << hijo2 << '\n';
      cout << endl;
      system("PAUSE");
break;
case 2:
    system("CLS");
      cout << "Padre 1: ";
      std::cout << " " << padre1 << '\n';
      cout << "Padre 2: ";
      std::cout << " " << padre2 << '\n';
      cout << endl;
      cout << "Puntos de cruza:" << endl;
      hijo1 = DosPunto(padre1, padre2);
      hijo2 = DosPunto(padre2, padre1);
      cout << "Hijo 1: ";
      std::cout << " " << hijo1 << '\n';
      cout << "Hijo 2: ";
      std::cout << " " << hijo2 << '\n';
      cout << endl;
      system("PAUSE");
break;
case 3:
    system("CLS");
      cout << "Padre 1: ";
      std::cout << " " << padre1 << '\n';
      cout << "Padre 2: ";
      std::cout << " " << padre2 << '\n';
      cout << endl;
      cout << "Mascara definida: 1001010110 " << endl;
      hijo1 = Uniforme(padre1, padre2);
      hijo2 = Uniforme(padre2, padre1);
      cout << "Hijo 1: ";
      std::cout << " " << hijo1 << '\n';
      cout << "Hijo 2: ";
      std::cout << " " << hijo2 << '\n';
      cout << endl;
      system("PAUSE");
break;
case 4:
    system("CLS");
      cout << "Padre 1: ";
      cout << padre1_2 << endl;
      cout << "Padre 2: ";
      cout << padre2_2 << endl;
      hijo1_2 = cruzaAcentuada(padre1_2,padre2_2);
      hijo2_2 = cruzaAcentuada(padre2_2,padre1_2);
      cout << endl;
      cout << "Hijo 1: ";
      cout << hijo1_2 << endl;
      cout << "Hijo 2: ";
      cout << hijo2_2 << endl;
      cout << endl;
      system("PAUSE");
break;
default:
      cout << "Opcion incorrecta " << endl;
```

```
            }

      }

      return 0;
}

void llenado(bitset<10> *set){
            *set = rand() % 1023;
}

bitset<10> UnPunto(bitset<10> &padre1, bitset<10> &padre2, int
punto_cruza) {
    bitset<10> aux = padre1;
    for (int i = 0; i <= punto_cruza; i++)
      aux.set(punto_cruza - i, padre2[punto_cruza - i]);
    return aux;
}

bitset<10> DosPunto(bitset<10> &padre1, bitset<10> &padre2){
    bitset<10> aux1 = padre1;
    bitset<10> aux2 = padre2;
    bitset<10> r1;
    bitset<10> r2;
    bitset<10> res3;
    bitset<10> result;
    bitset<10> mascara1 (string("1100000000"));
    bitset<10> mascara2 (string("0011110000"));
    bitset<10> mask3 (string("0000001111"));
    r1 = aux1 & mascara1;
    r2 = aux2 & mascara2;
    res3 = aux1 & mask3;
    int total = r1.to_ulong() + r2.to_ulong() + res3.to_ulong();
    result = total;
    return result;
}

bitset<10> Uniforme(bitset<10> &padre1, bitset<10> &padre2){

      bitset<10> aux1 = padre1;
    bitset<10> aux2 = padre2;
    bitset<10> r1;
    bitset<10> r2;
    bitset<10> mascara1 (string("1001010110"));
    bitset<10> mascara2 (string("0110101001"));
    r1 = aux1 & mascara1;
    r2 = aux2 & mascara2;
    bitset<10> result;
    int total = r1.to_ulong() + r2.to_ulong();
    result = total;
    return result;
}

char * cruzaAcentuada(char * padre1, char * padre2){
      int band = 0;
      int i = 0;
      char *aux = (char *)malloc(sizeof(char)*29);
```

```
	while(i <= 29){
		if (band == 0)
		{
			if(*padre1 != '!' && *padre2 != '!'){
				aux[i] = *padre1;
			}
			else{
				aux[i] = ' ';
				band = 1;
			}
			padre1++;
			padre2++;
		}
		else
		{
			if(*padre1 != '!' && *padre2 != '!'){
				aux[i] = *padre2;
			}
			else{
				aux[i] = ' ';
				band = 0;
			}
			padre1++;
			padre2++;
		}
		i++;
	}

	return aux;
}
```