



# Programación Evolutiva

## Tema 3: Implementación del algoritmo genético simple

Carlos Cervigón, Lourdes Araujo 2015-2016

## AG: definición

- ❑ Al ejecutar un AG, una **población** de individuos, que representan a un conjunto de candidatos a soluciones de un problema, es sometida a un proceso de **selección** que favorece a los mejores individuos y luego a una serie de **transformaciones (cruce y mutación)** con las que se actualiza la búsqueda.
- ❑ Cada ciclo de selección+búsqueda constituye una **generación**.
- ❑ Se espera que después de una serie de generaciones, el mejor individuo represente a un candidato lo suficientemente próximo a la solución buscada.

## Algoritmo genético simple (SGA)

- El AGS (SGA, simple genetic algorithm, Goldberg, 1989) es un algoritmo genético que incorpora los siguientes métodos y criterios:
  - **Criterio de codificación:** Específico de cada problema. Debe hacer corresponder a cada punto del dominio del problema un elemento del espacio de búsqueda: cadenas binarias.
  - **Criterio de tratamiento de los individuos no factibles:** No hay. Se considera que la codificación se hace de tal manera que todas las cadenas posibles representan a individuos factibles.
  - **Criterio de inicialización:** La población inicial está formada por cadenas binarias generadas al azar.

## Algoritmo genético simple (SGA)

- ❑ **Funciones de evaluación y aptitud:** La función de aptitud coincide con la de evaluación. La función de evaluación viene dada a través de una función objetivo.
- ❑ **Operadores genéticos:** Cruce monopunto y mutación bit a bit sobre los individuos codificados.
- ❑ **Criterio de selección:** Por ruleta.
- ❑ **Criterio de reemplazo:** Inmediato.
- ❑ **Criterio de parada:** Fijando el número máximo de iteraciones.
- ❑ **Parámetros de funcionamiento:** Discrecionales. Una posible elección
  - $Tampob=100$     $MaxIter=100$
  - $pcru=60\%$     $pmut=2\%$

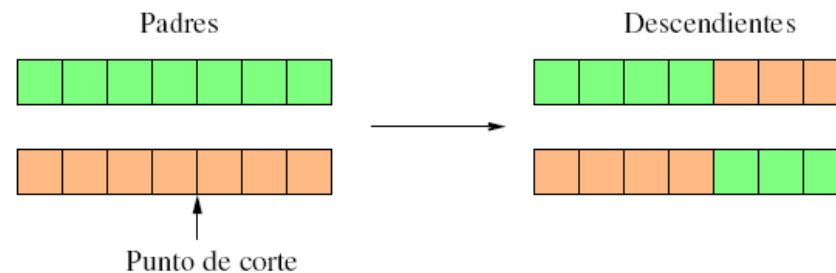
## Cruce monopunto

- El cruce monopunto genera dos descendientes a partir de dos progenitores cortándolos en una posición elegida al azar e intercambiando los respectivos segmentos.
- Dados dos progenitores codificados como

$$\mathbf{v} = (b_1 \dots b_l) \quad \text{y} \quad \mathbf{w} = (c_1 \dots c_l) \quad \text{con} \quad b_j, c_j \in \{0, 1\} \quad (\forall j = 1, \dots, l)$$

Se genera un número aleatorio  $pos \leftarrow \text{aleaent}[1, l-1]$  y se procede así:

$$\begin{array}{ccc} \langle b_1, \dots, b_{pos-1} | b_{pos}, \dots, b_l \rangle & & \langle b_1, \dots, b_{pos-1} | c_{pos}, \dots, c_l \rangle \\ & \Rightarrow & \\ \langle c_1, \dots, c_{pos-1} | c_{pos}, \dots, c_l \rangle & & \langle c_1, \dots, c_{pos-1} | b_{pos}, \dots, b_l \rangle \end{array}$$



## Algoritmo genético simple (SGA)

- ❑ Cada aplicación de la mutación bit a bit conmuta un bit de entre todos los  $(n \times l)$  de la población.
- ❑ La determinación de las parejas a cruzar y a mutar se realiza de acuerdo con las probabilidad de aplicación de los operadores genéticos.
- ❑ Para el cruce:
  - Para cada individuo se genera un número aleatorio
$$r_i \leftarrow \text{alea } [0,1]$$
  - Se comparan los  $r_i$  con la probabilidad de cruce,  $P_{cru}$  que es un parámetro del método.
- ❑ Son seleccionados para cruzarse todos los individuos  $v_i$  para los que  $r_i < P_{cru}$

## Algoritmo genético simple (SGA)

- ❑ Para la mutación se realiza el mismo proceso con cada uno de los  $(n \times l)$  bits de la población y usando la probabilidad de mutación  $P_{mut}$
- ❑ El emparejamiento de los individuos a cruzar se hace al azar: las etapas anteriores al cruce han introducido el suficiente desorden en la población como para que un emparejamiento sucesivo se pueda considerar aleatorio.
- ❑ Cuando el número de individuos a cruzar es impar, el último individuo quedará desemparejado y se elimina o se empareja con otro elegido al azar.

## Ejemplo : maximización de una función

$$f(x) = \text{abs}((x-5)/(2+\text{sen}(x)))$$

$$x \in [0,15]$$

*Representación:* cadena binaria de 4 bits

*Selección:* ruleta

*Prob. Cruce:* 0.7

*Prob. Mutación:* 0.3

*Tamaño población:* 2

*Número de generaciones:* 4

individuo

0,34→0	0,82→1	0,77→1	0,71→1	0111
0,35→0	0,75→1	0,48→0	0,40→0	0100



## Ejemplo : maximización de una función

### GENERACIÓN 1

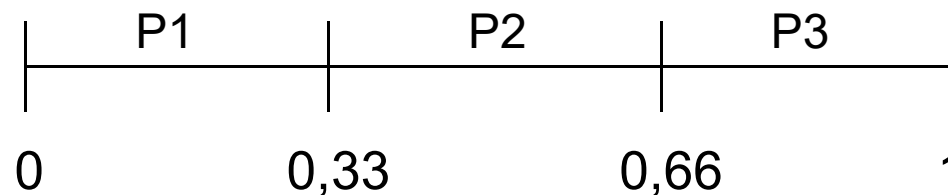
	<i>genotipo</i>	<i>fenotipo</i>	<i>f(x)</i>	<i>p.selección</i>	<i>p. acum</i>
Ind 1	0111	7	0.75	0.48	0.48
Ind 2	0100	4	0.80	0.52	1.00
			1.55		

- ❑ **SELECCIÓN** -> generamos un número aleatorio entre 0 y 1. En este caso, según la tabla de calidades:
  - Si el número aleatorio cae en el intervalo  $[0, 0.48)$  el individuo elegido será el 0111.
  - Si el valor aleatorio cae dentro del intervalo  $[0.48, 1)$  elegiremos el individuo 0100. (ej: 0111 y 0100)
- ❑ **CRUCE** : Ejecutar el cruce con probabilidad  $P_{cruce}$ . Si no hay que emparejar, salir.

**$0,15 < 0,7$  -> si hay cruce**

## Ejemplo : maximización de una función

- ❑ Elegir un punto de corte de las cadenas entre 1 y L-1. Miro el punto de corte con una elección equiprobable entre los 3 posibles puntos ( $1/N$ ).



**0,85 -> como cae en el tercer intervalo, el punto de corte es 3**

- ❑ Las cadenas que representan a los individuos se parten en dos trozos (por el punto de corte) y se intercambian, dando lugar a dos individuos nuevos: **011|1**                      **010|0**
- ❑ Por tanto los 2 nuevos individuos son: **0110** y **0101**

## Ejemplo : maximización de una función

### ■ **MUTACIÓN**

- Para cada gen miro si hay que mutar generando un número aleatorio:
  - Si el número aleatorio es menor que la *ProbMutación* (0,3) entontes cambiar el gen por su complementario (con alfabeto no binario para elegir un elemento del alfabeto se utiliza el procedimiento de elección equiprobable).
  - Si no se deja el gen como está.

0110	0.51→0	0.44→1	0.89→1	0.85→0	0110
0101	0.43→0	0.07→0	0.97→0	0.93→1	0001

## Ejemplo : maximización de una función

### GENERACIÓN 2

	genotipo	fenotipo	f(x)	p.selección	p. acum
Ind 1	0110	6	0.58	0.29	0.29
Ind 2	0001	1	1.41	0.71	1.00
			1.99		

- ❑ **SELECCIÓN** -> Generamos un número aleatorio entre 0 y 1. En este caso, según la tabla de calidades:
  - Si el número aleatorio cae en el intervalo  $[0, 0.29)$  el individuo elegido será el 0110.
  - Si el valor aleatorio cae dentro del intervalo  $[0.29, 1)$  elegiremos el individuo 0001. (ej: 0110 y 0001)
- ❑ **CRUCE** → Ejecutar el cruce con probabilidad  $P_{cruce}$ . Si no hay que emparejar, salir.

**$0,75 > 0,7$  -> no hay cruce**

## Ejemplo : maximización de una función

### ■ **MUTACIÓN**

- Para cada gen miro si hay que mutar generando un número aleatorio:
  - Si el número aleatorio es menor que la P. Mutación (0,3) entontes cambiar el gen por su complementario (con alfabeto no binario para elegir un elemento del alfabeto se utiliza el procedimiento de elección equiprobable).
  - Si no se deja el gen como está.

0110	0.90→0	0.51→1	0.62→1	0.67→0	0110
0001	0.15→1	0.89→0	0.87→0	0.86→1	1001

## Ejemplo : maximización de una función

### GENERACIÓN 3

	genotipo	fenotipo	f(x)	p.selección	p. acum
Ind 1	0110	6	0.58	0.26	0.26
Ind 2	1001	9	1.65	0.74	1.00
			2.23		

- ❑ **SELECCIÓN** -> Generamos un número aleatorio entre 0 y 1. En este caso, según la tabla de calidades:
  - Si el número aleatorio cae en el intervalo  $[0, 0.26)$  el individuo elegido será el 0110.
  - Si el valor aleatorio cae dentro del intervalo  $[0.26, 1)$  elegiremos el individuo 1001. (ej: 1001 y 1001)
- ❑ **CRUCE** → Ejecutar el cruce con probabilidad  $P_{cruce}$ . Si no hay que emparejar, salir.

**$0,84 > 0,7 \rightarrow$  no hay cruce**

## Ejemplo : maximización de una función

### ■ **MUTACIÓN**

- Para cada gen miro si hay que mutar generando un número aleatorio:
  - Si el número aleatorio es menor que la P. Mutación (0,3) entontes cambiar el gen por su complementario (con alfabeto no binario para elegir un elemento del alfabeto se utiliza el procedimiento de elección equiprobable).
  - Si no se deja el gen como está.

1001	0.98→1	0.33→0	0.25→1	0.07→0	1010
1001	0.18→0	0.84→0	0.27→1	0.08→0	0010

## Ejemplo : maximización de una función

### GENERACIÓN 4

	genotipo	fenotipo	f(x)	p.selección	p. acum
Ind 1	1010	10	3.43	0.77	0.77
Ind 2	0010	2	1.03	0.23	1.00
			4.46		

- **SELECCIÓN** -> Generamos un número aleatorio entre 0 y 1. En este caso, según la tabla de calidades:

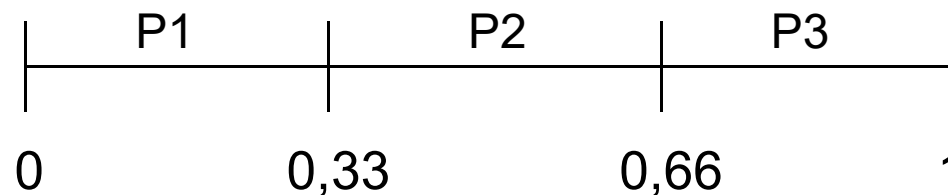
  - Si el número aleatorio cae en el intervalo  $[0, 0.77)$  el individuo elegido será el 1010.
  - Si el valor aleatorio cae dentro del intervalo  $[0.77, 1)$  elegiremos el individuo 0010. (ej: 1010 y 1010)
- **CRUCE** → Ejecutar el cruce con probabilidad  $P_{cruce}$ . Si no hay que emparejar, salir.

**$0,28 < 0,7 \rightarrow$  si hay cruce**



## Ejemplo : maximización de una función

- ❑ Elegir un punto de corte de las cadenas entre 1 y L-1. Miro el punto de corte con una elección equiprobable entre los 3 posibles puntos.



**0,6 -> como cae en el tercer intervalo, el punto de corte es 2**

- ❑ Las cadenas que representan a los individuos se parten en dos trozos (por el punto de corte) y se intercambian, dando lugar a dos individuos nuevos: 10|10 10|10
- ❑ Por tanto los 2 nuevos individuos son: 1010 y 1010

## Ejemplo : maximización de una función

### ■ **MUTACIÓN**

- Para cada gen miro si hay que mutar generando un número aleatorio:
  - Si el número aleatorio es menor que la P. Mutación (0,3) entonces cambiar el gen por su complementario (con alfabeto no binario para elegir un elemento del alfabeto utilizo el procedimiento de elección equiprobable).
  - Si no se deja el gen como está.
  - Después de 4 iteraciones el individuo de mayor calidad es : **1010**

## El algoritmo

```
funcion Algoritmo_Genético() {  
    TPoblacion pob;      // población  
    . . .  
    obtener_parametros(parametros);  
    pob = poblacion_inicial();  
    evaluacion(pob, tam_pob, pos_mejor, sumadaptacion);  
    // bucle de evolución  
    mientras no se alcanza la condición de terminación hacer{  
        selección(pob, parámetros);  
        reproduccion(pob, parámetros);  
        evaluacion(pob, parámetros, pos_mejor, sumadaptacion);  
    }  
    devolver pob[pos_mejor];  
}
```

## El algoritmo

Clase AGenetico {

```
Cromosoma[] pob; // población
entero tam_pob; // tamaño población
entero num_max_gen; // número máximo de generaciones
Cromosoma elMejor; // mejor individuo
entero pos_mejor; // posición del mejor cromosoma
real prob_cruce; // probabilidad de cruce
real prob_mut; // probabilidad de mutación
real tol; // tolerancia de la representación
. . .
```

## El algoritmo

```
public static void main(String[] args) {  
    . . .  
    AGenetico AG = new AGenetico();  
    AG.inicializa(); //crea población inicial de cromosomas  
    AG.evaluarPoblacion(); //evalúa los individuos y coge el mejor  
    while (!GA.terminado()) {  
        AG.numgeneracion++;  
        AG.seleccion();  
        AG.reproduccion();  
        AG.mutacion();  
        AG.evaluarPoblacion();  
        . . .  
    }  
    devolver pob[pos_mejor];  
}
```

## El individuo: Cromosoma

```
Clase abstracta Cromosoma {  
    boolean[] genes; //cadena de bits (genotipo)  
    real fenotipo; //fenotipo  
    real aptitud; //función de evaluación fitness adaptación);  
    real puntuación; //puntuación relativa(aptitud/suma)  
    real punt_acum; /puntuación acumulada para selección  
    . . .  
}  
  
tipo TPoblacion: vector de Cromosoma (individuos);
```

## El individuo: Cromosoma concreto

```
class CromosomaProblemaConcreto extends Cromosoma {  
    . . .  
    // extremos del intervalo considerado  
    // para los valores del dominio  
    xmin= . . .;  
    xmax=. . .;  
  
    . . .  
    longitudCromosoma = . . .  
    . . .  
    double fenotipo() {. . .} //el valor del individuo  
    double evalua() { . . .} //adaptacion del individuo  
    . . .  
}
```

## Calculo de longitud de cromosoma

- ❑ Buscamos una codificación: representación del dominio del problema mediante enteros binarios sin signo.
- ❑ La elección más sencilla consiste en discretizar el intervalo  $[x_{min}, x_{max}]$  en una cantidad de  $2^{lcrom}$  puntos tal que la distancia entre puntos consecutivos sea menor que la tolerancia especificada,

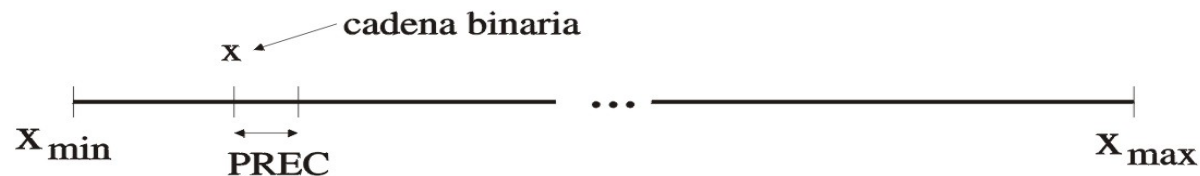
$$\frac{x_{max} - x_{min}}{2^{lcrom} - 1} < TOL$$

- ❑ Cada punto del espacio de búsqueda queda representado mediante un entero binario de longitud  $lcrom$ , comenzando por 0...0 que representa a  $x_{min}$  y terminando en 1...1 que representa a  $x_{max}$



## Calculo de longitud de cromosoma

- Buscamos una codificación: representación del dominio del problema mediante enteros binarios sin signo.



- La elección más sencilla consiste en discretizar el intervalo  $[x_{min}, x_{max}]$  en una cantidad de  $2^{lcrom}$  puntos tal que la distancia entre puntos consecutivos sea menor que la tolerancia especificada,

$$\frac{x_{max} - x_{min}}{2^{lcrom} - 1} < TOL$$

- Cada punto del espacio de búsqueda queda representado mediante un entero binario de longitud  $lcrom$ , comenzando por 0...0 que representa a  $x_{min}$  y terminando en 1...1 que representa a  $x_{max}$

## Calculo de longitud de cromosoma

- La longitud de los individuos se calcula a partir de la tolerancia  $TOL$ :

$$lcrom = \left\lceil \log_2 \left( 1 + \frac{x_{max} - x_{min}}{TOL} \right) \right\rceil$$

- Recíprocamente se puede calcular el punto  $x$  que corresponde a un individuo  $v$  mediante la siguiente fórmula de decodificación:

$$x(v) = x_{min} + \text{bin2dec}(v) \cdot \frac{x_{max} - x_{min}}{2^{lcrom} - 1}$$

## Generación de la población inicial

```
//crea los cromosomas y los inicializa aleatoriamente  
private void inicializa() {  
...  
    for (int j = 0; j < tampoblacion; j++) {  
        poblacion[j] = new CromosomaProblemaConcreto();  
        poblacion[j].inicializaCromosoma();  
        poblacion[j].aptitud=poblacion[j].evalua();{aptitud}  
    }  
    public void inicializaCromosoma() {  
        for (int i = 0; i < longitudCromosoma; i++) {  
            genes[i] = MyRandom.boolRandom();  
        }  
    }  
}
```

Si es < 0.5, asigna 0; caso contrario asigna 1

## Función de adaptación (fitness)

```
Clase CromosomaProblemaConcreto {  
    . . .  
    double evalua( ){  
        //calcula fitness o adaptacion del cromosoma  
        double x; // fenotipo  
        x = fenotipo();  
        devolver f(x); //valor de la función a optimizar  
    }  
    double fenotipo( ){  
        . . .  
        valor = x_min + (x_max - x_min) * bin_dec(genes, lcrom)  
            / (pow(2,lcrom) - 1);  
        devolver valor;  
    }  
}
```

## Evaluación de la población

- En la evaluación se revisan los contadores de aptitud relativa, y puntuación acumulada de los individuos de la población; además se calcula la posición del mejor individuo.

```
private void evaluarPoblacion() {  
    real punt_acu = 0; // puntuación acumulada  
    real aptitud_mejor = 0; // mejor aptitud  
    real sumaptitud = 0; // suma de la aptitud  
    . . .  
    para cada i desde 0 hasta tam_pob hacer {  
        sumaptitud = sumaptitud + poblacion[i].aptitud;  
        si (poblacion[i].aptitud > aptitud_mejor){  
            pos_mejor = i;  
            aptitud_mejor = poblacion[i].aptitud;  
        }  
    }  
}
```

## Evaluación de la población

- En la evaluación se revisan los contadores de aptitud relativa, y puntuación acumulada de los individuos de la población.

```
para cada i desde 0 hasta tam_pob hacer {  
    pob[i].puntuacion = pob[i].aptitud / sumaptitud;  
    pob[i].punt_acu = pob[i].puntuacion + punt_acu;  
    punt_acu = punt_acu + pob[i].puntuacion;  
} }
```

```
//Si el mejor de esta generación es mejor que el mejor que  
    tenía de antes  pues lo actualizo
```

```
    if (aptitud_mejor > elMejor.dameAptitud()) {  
        elMejor ← pob[pos_mejor]  
    }
```

- Aquí tendremos que hacer variaciones en caso de que sea una minimización.

## Selección y reproducción

- ❑ Los operadores de selección, cruce y mutación se combinan para producir una nueva generación.
- ❑ En un AGS, el proceso comienza con la selección de los individuos que sobreviven.
- ❑ La implementación de la reproducción consiste en la selección de los individuos a reproducirse entre los de la población resultante, y en la aplicación del operador de cruce a cada una de las parejas.
- ❑ En todos los casos se puede realizar la mutación bit a bit en función de la probabilidad de mutación.

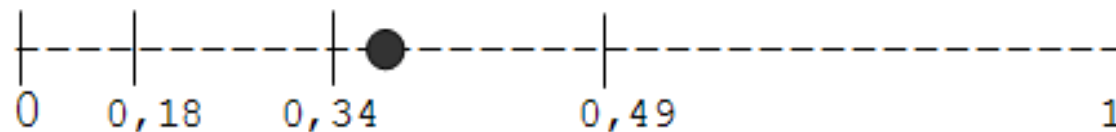
## Selección

- ❑ La función de selección escoge, por ejemplo, por el método de ruleta, un número de supervivientes igual al tamaño de la población.
- ❑ La función modifica la población que pasa a estar formada únicamente por ejemplares de los individuos supervivientes.
- ❑ Antes de hacer la selección hemos calculado las puntuaciones de cada individuo:
  - **aptitud** (fitness)
  - **puntuación** ( $\text{fitness} / \text{suma\_fitness}$ )
  - **puntuación acumulada**



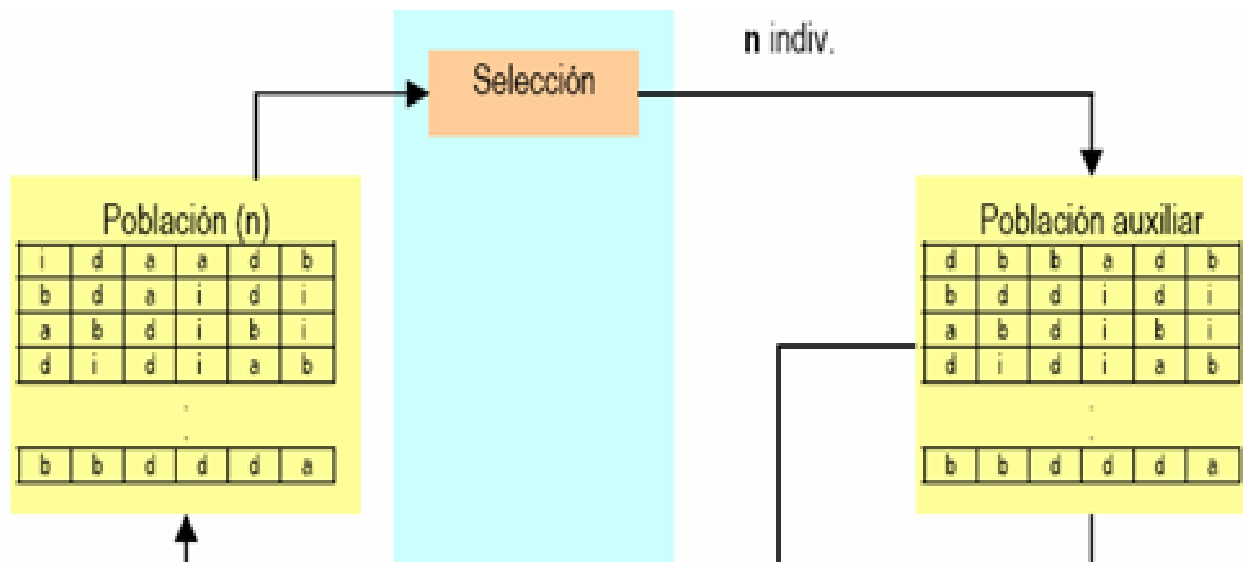
## Selección

```
void selecciónRuleta ( pob, nuevaPob, tam_pob) {  
    entero sel_super[tam_pob]; //seleccionados para sobrevivir  
    real prob; // probabilidad de seleccion  
    entero pos_super; // posición del superviviente  
    para cada i desde 0 hasta tam_pob hacer {  
        prob = alea();  
        pos_super = 0; → 0,38  
        mientras ((prob > pob[pos_super].punt_acu) y  
                   (pos_super < tam_pob)) pos_super++;  
        sel_super[i] = pos_super;  
    }  
}
```



# Selección

```
// se genera la poblacion intermedia
para cada i desde 0 hasta tam_pob hacer {
  copiar (pob[sel_super[i]], nuevaPob);
}
```



## Reproducción : cruce

```
reproduccion ( ) {  
  
    //seleccionados para reproducir  
    entero sel_cruce[tam_pob];  
  
    //contador seleccionados  
    entero num_sele_cruce = 0;;  
    entero punto_cruce;  
    real prob;  
    Cromosoma hijo1,hijo2;
```

## Reproducción : cruce

```
//Se eligen los individuos a cruzar
para cada i desde 0 hasta tam_pob {
    //se generan tam_pob números aleatorios en [0 1)
    prob = alea();
    //se eligen los individuos de las posiciones i si prob <
    prob_cruce
    si (prob < prob_cruce){
        sel_cruce[num_sel_cruce] = i;
        num_sel_cruce++;
    }
}
// el numero de seleccionados se hace par
si ((num_sel_cruce mod 2) == 1)
    num_sel_cruce--;
```

## Reproducción : cruce

```
// se cruzan los individuos elegidos en un punto al azar

punto_cruce = alea_ent(0,lcrom);
para cada i desde 0 hasta num_sel_cruce avanzando 2{
    cruce(pob[sel_cruce[i]], pob[sel_cruce[i+1]]
        hijo1, hijo2, punto_cruce, . . . );

// los nuevos individuos sustituyen a sus progenitores
pob[sel_cruce[i]] = hijo1;
pob[sel_cruce[i+1]] = hijo2;
}
}
```

## Reproducción : cruce

- ❑ El operador de cruce toma dos padres y genera dos cadenas hijas. Recibe la probabilidad de cruce. La función calcula la aptitud de los nuevos individuos.

```
cruce(padre1,padre2,hiijo1,hiijo2, puntoCruce){
```

```
    entero i;
```

```
    hiijo1.genes.iniciar();
```

```
    hiijo2.genes.iniciar();
```

```
    // primera parte del intercambio: 1 a 1 y 2 a 2
```

```
    para cada i desde 0 hasta punto_cruce hacer{
```

```
        hiijo1.genes.insertar(padre1.genes[i]);
```

```
        hiijo2.genes.insertar(padre2.genes[i]);
```

```
    }
```

```
// segunda parte: 1 a 2 y 2 a 1
```

```
para cada i desde punto_cruce hasta lcrom; hacer{  
    hijo1.genes.insertar(padre2.genes[i]);  
    hijo2.genes.insertar(padre1.genes[i]);  
}
```

```
// se evalúan
```

```
hijo1.aptitud = evalua(hijo1, . . . );  
hijo2.aptitud = evalua(hijo2, . . . );  
}
```

## Mutación

- ❑ El operador de mutación considera la posible mutación de cada gen del genotipo con la probabilidad indicada.
- ❑ La función revisa la aptitud del individuo en caso de que se produzca alguna mutación.

mutacion(pob, tam\_pob, prob\_mut, . . . ) {

    booleano mutado;

    entero i,j;

    real prob;



## Mutación

```
para cada i desde 0 hasta tam_pob hacer{
    mutado = false;
    para cada j desde 0 hasta lcrom hacer{
        // se genera un numero aleatorio en [0 1)
        prob = alea();
        // mutan los genes con prob<prob_mut
        si (prob<prob_mut){
            pob[i].genes[j] = not( pob[i].genes[j]);
            mutado = true;
        }
    }
    si (mutado)
        pob[i].aptitud = pob[i].evalua();
}
}
```

## Ejemplo: óptimo de una función

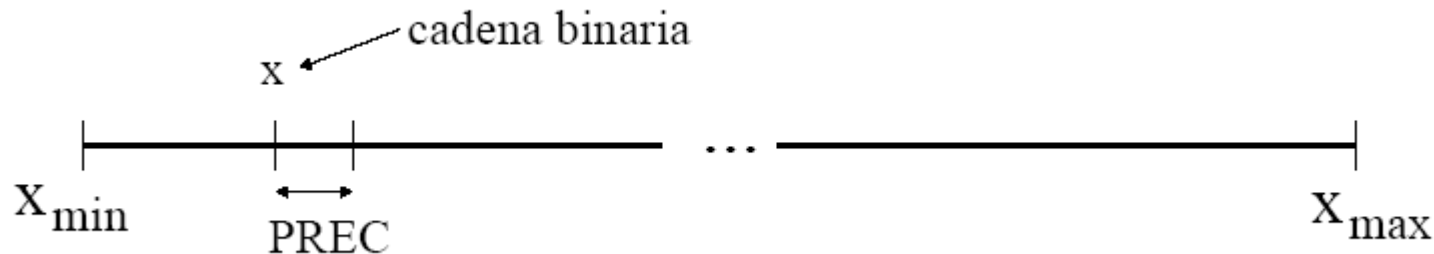
- ❑ Vamos a aplicar nuestro algoritmo genético a la búsqueda del máximo en el intervalo  $[0,20]$  de una sencilla función:

$$f(x) = \frac{x}{1+x^2}$$

- ❑ La sencillez de la función no justifica la necesidad de aplicación de un AG. Sin embargo, precisamente esta sencillez nos permite utilizarla para clarificar el funcionamiento del AG, que se aplicaría exactamente de la misma forma a otras funciones más complejas.
- ❑ Hemos elegido los siguientes valores para los parámetros:
  - Precisión o tolerancia 0.0001
  - Tamaño de población 30
  - Tasa de cruces 40%
  - Tasa de mutaciones 1%
  - Número de generaciones 10

## Ejemplo: óptimo de una función

- Para discretizar el intervalo real contenido entre  $x_{\min}$  y  $x_{\max}$ , lo dividimos en pequeñas porciones de anchura menor que PREC, como muestra la figura.



$$\text{tamaño porción} = \frac{x_{\max} - x_{\min}}{2^l - 1} < \text{PREC}$$

## Ejemplo: óptimo de una función

- De esta fórmula podemos obtener la longitud de cadena binaria que necesitamos utilizar para que nuestro algoritmo pueda alcanzar la precisión requerida:

$$l = \left\lceil \log_2 \left( 1 + \frac{x_{max} - x_{min}}{PREC} \right) \right\rceil$$

- En nuestro caso, en que  $x_{min} = 0$  y  $x_{max} = 20$ , tenemos

$$l = \left\lceil \log_2 \left( 1 + \frac{20 - 0}{0,0001} \right) \right\rceil = 18$$

## Ejemplo: óptimo de una función

- ❑ Una vez calculada la longitud de las cadenas binarias de la población podemos generar la población inicial. La siguiente tabla muestra el resultado de esta operación en una ejecución de nuestro algoritmo ejemplo.
- ❑ La tabla presenta para cada individuo su genotipo (la cadena binaria), su fenotipo (el valor real que le corresponde a la cadena binaria) y el valor de adaptación correspondiente al fenotipo.
- ❑ En este caso la adaptación media de la población es de 0.183415 y el mejor individuo es el de la posición 17, con un valor de adaptación de 0.486318.
- ❑ Podemos observar que en la población inicial los valores de las adaptaciones son muy variados.

## Ejemplo: óptimo de una función

Posición	Individuo	x	adaptación
1	000010100010010000	0.709231	0.471874
2	100000010100101011	16.6115	0.059982
3	101110010101101001	11.7698	0.084354
4	001001101111001100	4.05061	0.232694
5	000011011110111101	14.8377	0.0670911
6	001010110011111001	12.4381	0.0798818
7	100000000110111000	2.30477	0.365143
8	110001101101010100	3.34741	0.274262
9	000000110000111110	9.70219	0.101986
10	100010000000101000	1.5638	0.453871
11	000000011011001011	16.5137	0.0603344
12	100111011110100111	17.9634	0.0554968
13	001000111101111111	19.9174	0.0500812
14	010010110110110011	16.0708	0.0619848
15	101111001000110010	5.96171	0.163147
16	001100101101101011	16.7832	0.0593726
17	100101100001010000	0.789264	0.486318
18	010011100010010101	13.2119	0.0752583
19	001000011011000100	2.76399	0.31992
20	110101110001101010	6.7367	0.14524

## Ejemplo: óptimo de una función

- ❑ En este caso la adaptación media de la población es de 0.183415 y el mejor individuo es el de la posición 17, con un valor de adaptación de 0.486318.
- ❑ Podemos observar que en la población inicial los valores de las adaptaciones son muy variados.
- ❑ Siguiendo con los pasos del algoritmo, la siguiente fase es el proceso de selección de supervivientes por el método de la ruleta. La siguiente tabla muestra en la primera columna los valores generados por este método y la población resultante del proceso.

## Ejemplo: óptimo de una función

Selección	Individuo	x	adaptación
11	000000011011001011	16.5137	0.0603344
10	100010000000101000	1.5638	0.453871
19	001000011011000100	2.76399	0.31992
6	001010110011111001	12.4381	0.0798818
15	101111001000110010	5.96171	0.163147
17	100101100001010000	0.789264	0.486318
20	110101110001101010	6.7367	0.14524
3	101110010101101001	11.7698	0.084354
17	100101100001010000	0.789264	0.486318
8	110001101101010100	3.34741	0.274262
20	110101110001101010	6.7367	0.14524
20	110101110001101010	6.7367	0.14524
8	110001101101010100	3.34741	0.274262
12	100111011110100111	17.9634	0.0554968
17	100101100001010000	0.789264	0.486318
17	100101100001010000	0.789264	0.486318
8	110001101101010100	3.34741	0.274262
2	100000010100101011	16.6115	0.059982
17	100101100001010000	0.789264	0.486318
6	001010110011111001	12.4381	0.0798818



## Ejemplo: óptimo de una función

- ❑ Tras el proceso de selección, la adaptación media de la población sube a 0.252348.
- ❑ Podemos observar que los individuos más adaptados, como el 17, tienden a recibir más copias en la nueva población, mientras que los individuos de baja adaptación, como el 2, tienden a desaparecer.

## Ejemplo: óptimo de una función

- ❑ El siguiente paso de la evolución es la reproducción o generación de nuevos individuos mediante el operador de cruce.
- ❑ La siguiente tabla muestra los padres elegidos aleatoriamente para el cruce comprobando la tasa de mutación, y el punto de cruce, también elegido aleatoriamente.

Punto de cruce	padre1	padre2
13	2	3
4	7	12
3	13	15
3	16	17
16	18	19

- ❑ Y después del cruce:

## Ejemplo: óptimo de una función

Posición	Individuo	x	adaptación
1	000000011011001011	16.5137	0.0603344
2	100010000000100100	2.81381	0.315537
3	001000011011001000	1.51398	0.459877
4	001010110011111001	12.4381	0.0798818
5	101111001000110010	5.96171	0.163147
6	100101100001010000	0.789264	0.486318
7	110101110001101010	6.7367	0.14524
8	101110010101101001	11.7698	0.084354
9	100101100001010000	0.789264	0.486318
10	110001101101010100	3.34741	0.274262
11	110101110001101010	6.7367	0.14524
12	110101110001101010	6.7367	0.14524
13	110101100001010000	0.789416	0.48634
14	100111011110100111	17.9634	0.0554968
15	100001101101010100	3.34726	0.274272
16	100001101101010100	3.34726	0.274272
17	110101100001010000	0.789416	0.48634
18	100000010100101000	1.61141	0.448032
19	100101100001010011	15.7893	0.0630809
20	001010110011111001	12.4381	0.0798818

## Ejemplo: óptimo de una función

- ❑ En este caso, la adaptación media a descendido ligeramente, a 0.250673.
- ❑ El mejor individuo en este caso sigue siendo el de la posición 17, y es el producto de uno de los cruces, teniendo un valor de adaptación ligeramente mejor (0.48634) que el de sus progenitores (0.486318 y 0.274262).
- ❑ Finalmente se aplica el operador de mutación. La siguiente tabla muestra los posiciones de los individuos y posiciones (dentro del individuo) de los genes que han mutado.

Individuo	gen
9	15
10	16
14	8
20	2

## Ejemplo: óptimo de una función

Posición	Individuo	x	adaptación
1	000000011011001011	16.5137	0.0603344
2	100010000000100100	2.81381	0.315537
3	001000011011001000	1.51398	0.459877
4	001010110011111001	12.4381	0.0798818
5	101111001000110010	5.96171	0.163147
6	100101100001010000	0.789264	0.486318
7	110101110001101010	6.7367	0.14524
8	101110010101101001	11.7698	0.084354
9	100101100001011000	2.03927	0.395313
10	110001101101010000	0.8474	0.493223
11	110101110001101010	6.7367	0.14524
12	110101110001101010	6.7367	0.14524
13	110101100001010000	0.789416	0.48634
14	100111001110100111	17.9536	0.0555268
15	100001101101010100	3.34726	0.274272
16	100001101101010100	3.34726	0.274272
17	110101100001010000	0.789416	0.48634
18	100000010100101000	1.61141	0.448032
19	100101100001010011	15.7893	0.0630809
20	011010110011111001	12.4382	0.0798808

## Ejemplo: óptimo de una función

- ❑ Tras la mutación, la adaptación media de la población ha pasado a ser 0.257073. El mejor individuo es ahora el 10, es el producto de una mutación, con una adaptación de 0.493223, un valor que ya se acerca al óptimo de la función que está en 0.5.
- ❑ La evolución de los valores medio y máximo de la adaptación con las generaciones aparece en la siguiente tabla:

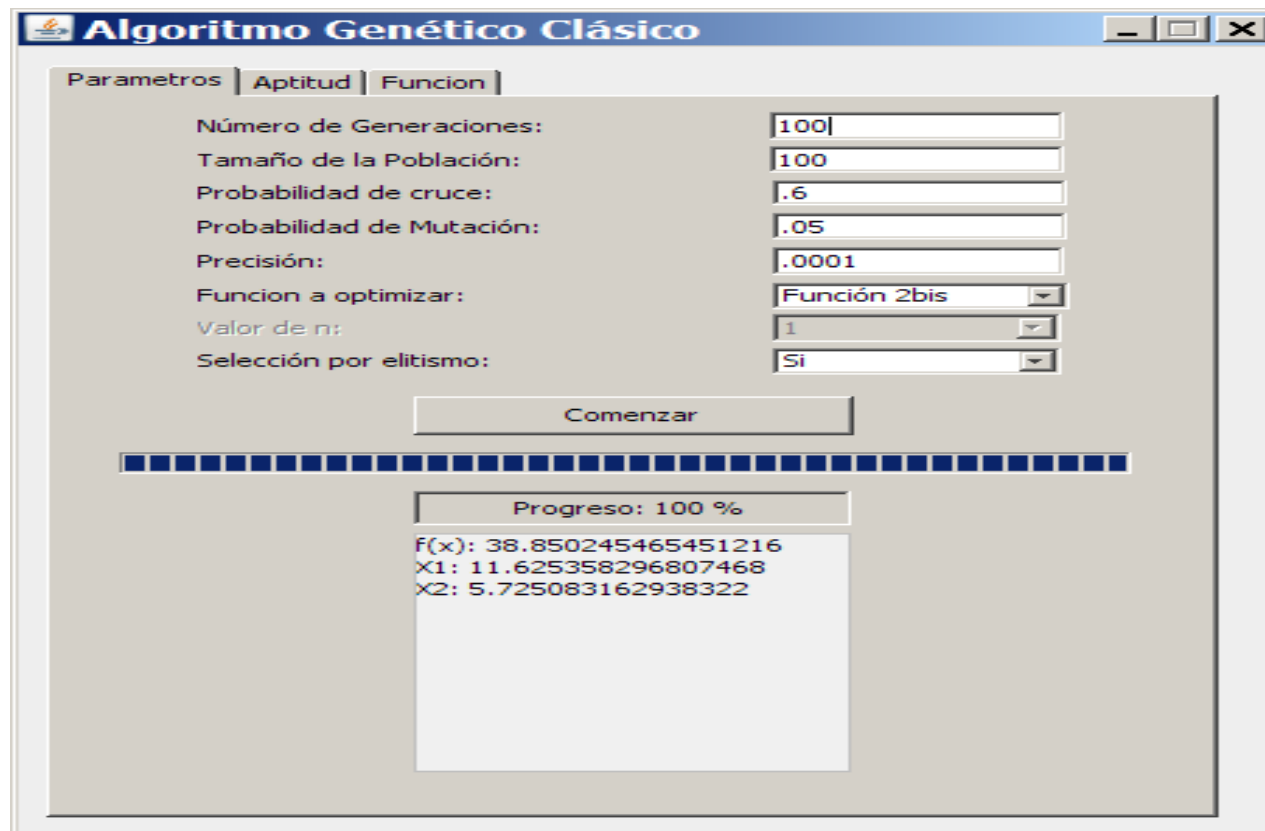
Generación	Adap. Media	Adap. Max.	x
1	0.257073	0.493223	0.8474
2	0.378307	0.498506	0.925525
3	0.395765	0.49323	0.847476
4	0.454126	0.499953	0.986408
5	0.458169	0.499953	0.986408
6	0.467791	0.499953	0.986408
7	0.483441	0.499953	0.986408
8	0.483225	0.499953	0.986408
9	0.489634	0.499997	1.00365
10	0.488957	0.499997	1.00357

## Ejemplo: óptimo de una función

- ❑ Podemos ver como a medida que avanzan las generaciones los valores medio y máximo de la adaptación de la población tienden a mejorar.
- ❑ Sin embargo, la mejora de la adaptación de una generación a otra no está garantizada. a menos que se introduzca elitismo, una técnica para garantizar la supervivencia del mejor, o de algunos de los mejores, de generación en generación.
- ❑ Así, en nuestro ejemplo podemos ver que de la generación 2 a la 3 la adaptación del mejor individuo de la población baja de 0.498506 a 0.49323.

## Ejemplo

- $f(x,y) = 21.5 + x.\text{sen}(4\pi x) + y.\text{sen}(20\pi y)$  :  
que presenta un máximo de 38.809 en 11.625 y 5.726



The screenshot shows a software window titled "Algoritmo Genético Clásico". It has three tabs: "Parametros", "Aptitud", and "Funcion". The "Parametros" tab is active, displaying the following settings:

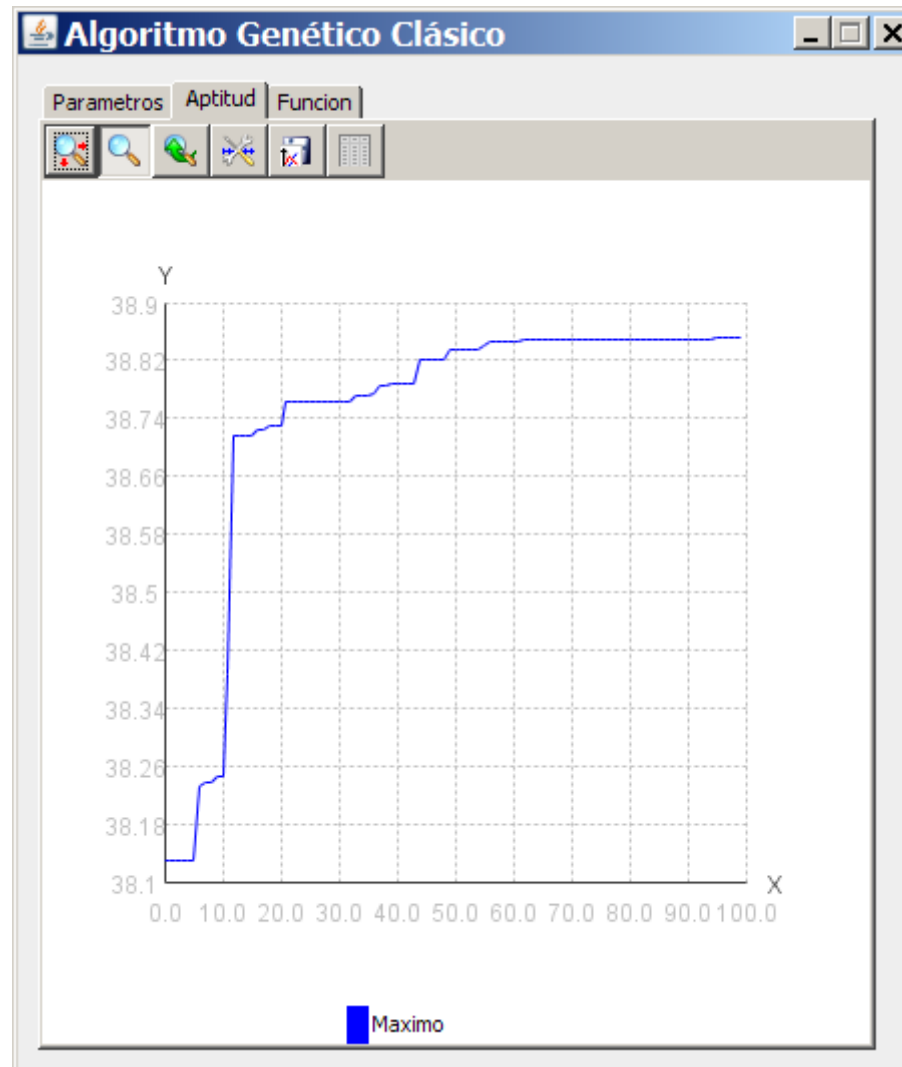
Parameter	Value
Número de Generaciones:	100
Tamaño de la Población:	100
Probabilidad de cruce:	.6
Probabilidad de Mutación:	.05
Precisión:	.0001
Funcion a optimizar:	Función 2bis
Valor de n:	1
Selección por elitismo:	Si

Below the parameters is a "Comenzar" button. Underneath the button is a progress bar consisting of 100 small blue squares, all of which are filled. Below the progress bar is a text box displaying the results:

Progreso: 100 %  
f(x): 38.850245465451216  
X1: 11.625358296807468  
X2: 5.725083162938322



## Ejemplo: Gráfica de evolución



### ❑ Presión Selectiva:

“Fuerza” del mecanismo de la selección:

$$\textit{fitness máximo/fitness medio}$$

Si la presión selectiva es baja los valores de aptitud están muy cerca entre sí y se produce un estancamiento de la búsqueda.

### ❑ Convergencia prematura

Se produce cuando los individuos muy aptos dominan la población rápidamente. Óptimos locales.