Algoritmos Genéticos

Estructura y componentes básicos de los AGs

Definiciones

Alelo. Son los distintos valores con los cuales se puede representar un gen.

Gen. Es el valor de un alelo dentro de un arreglo.

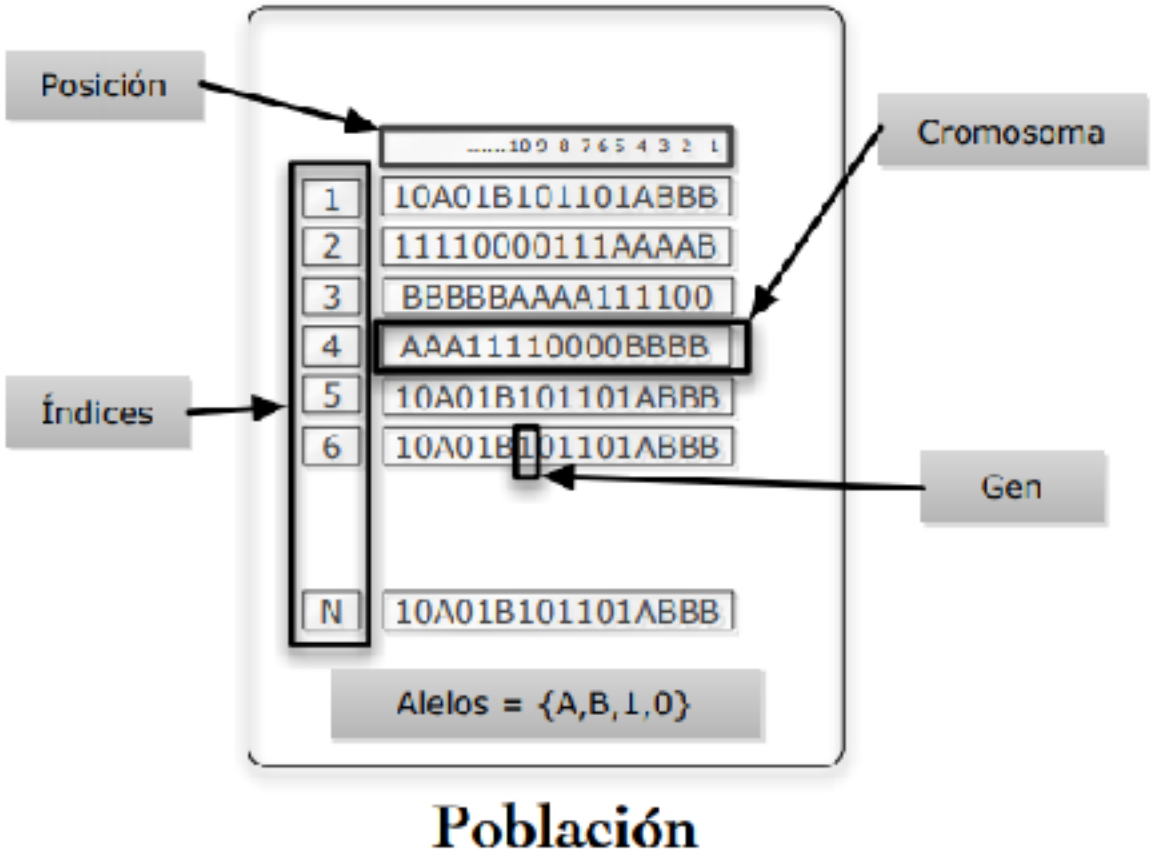
Cromosoma. Es una colección de genes en forma de arreglo.

Posición. Es el lugar que ocupa un gen dentro del cromosoma.

Índice. Es la posición que tiene el individuo dentro de la población

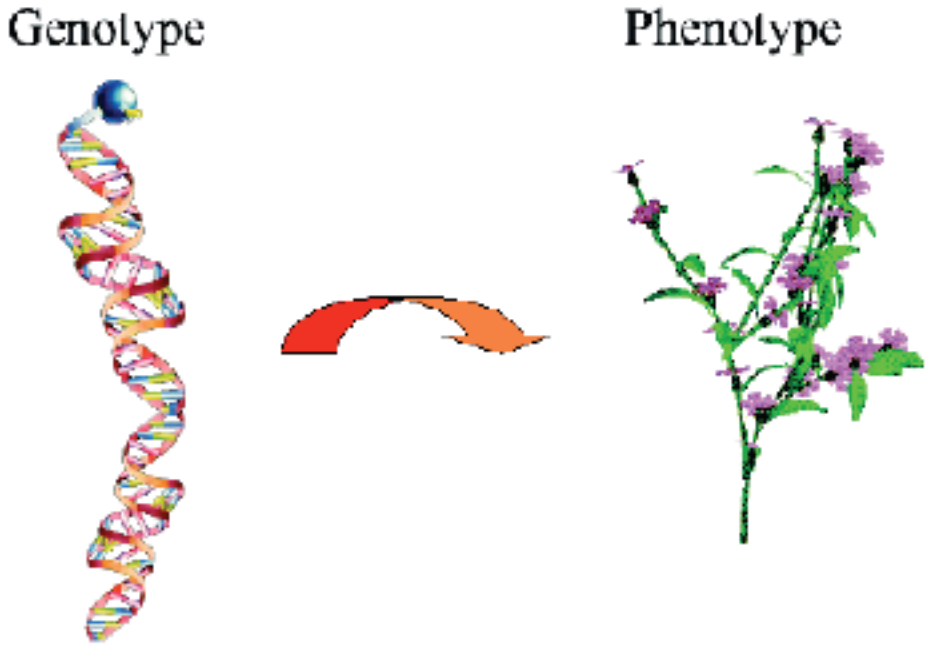
Definiciones

- Alelo
- Gen
- Cromosoma
- Población
- Índice



Población

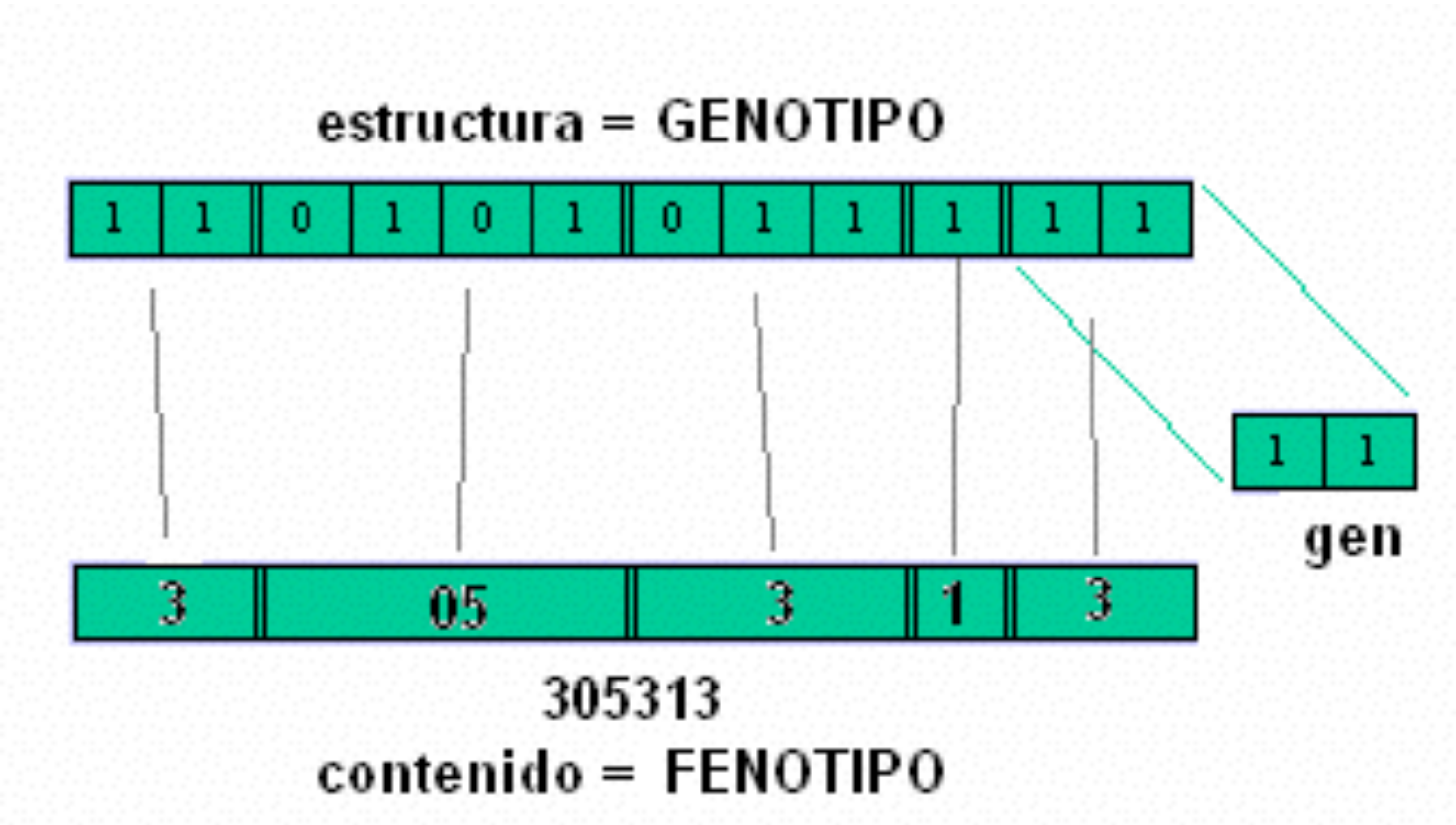
Terminología



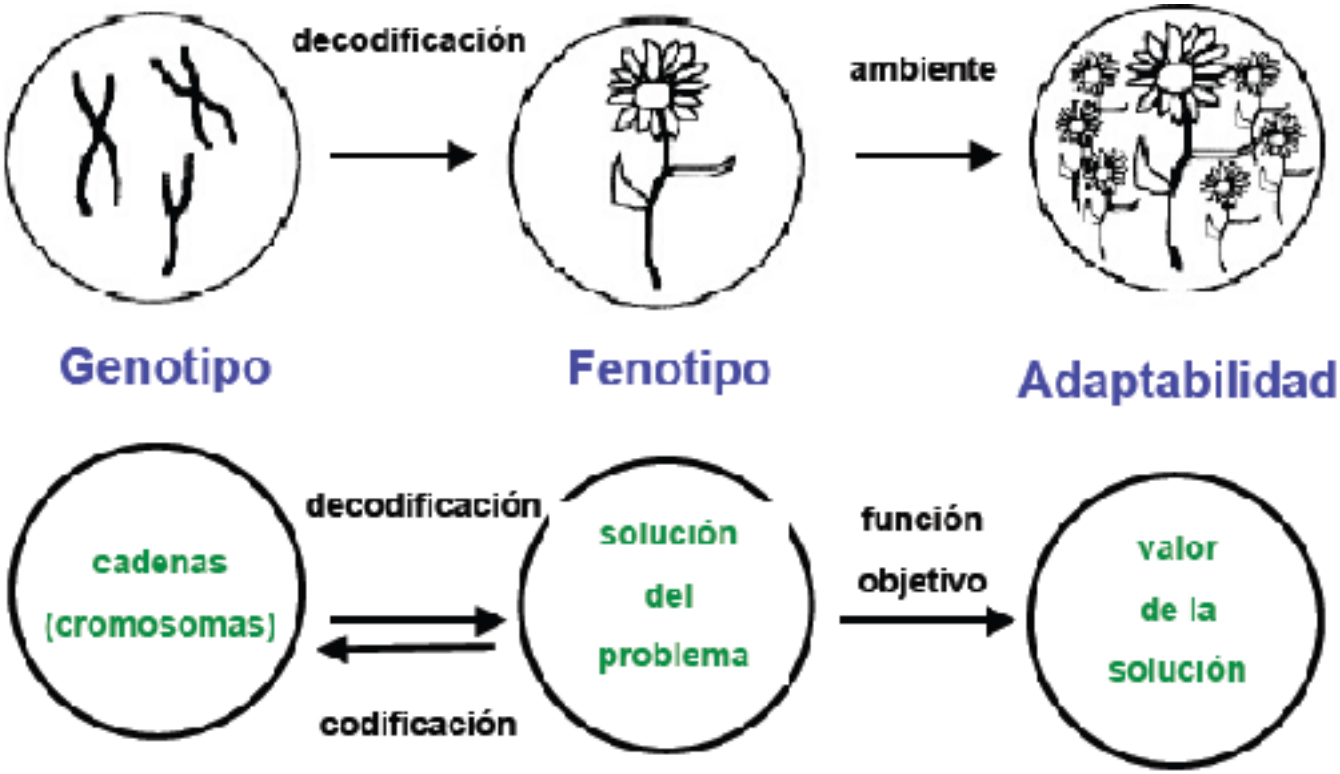
- El **genotipo** es la totalidad de la información genética que posee un organismo en particular
- El **fenotipo** a la expresión del genotipo en función de un determinado ambiente, característica externa.

Los individuos

Estructura y contenido de los individuos



Adaptabilidad



Bucle básico de un AG

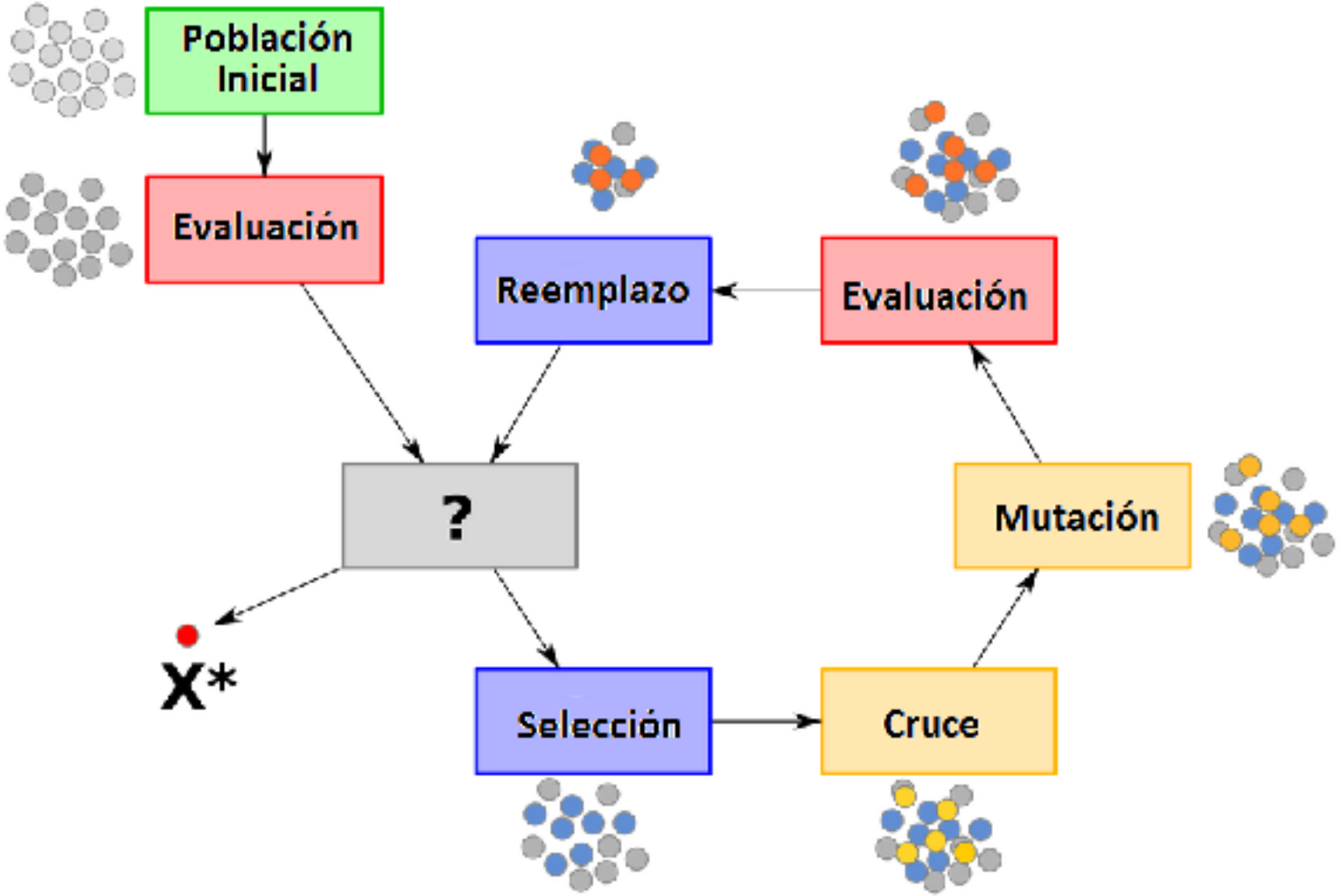
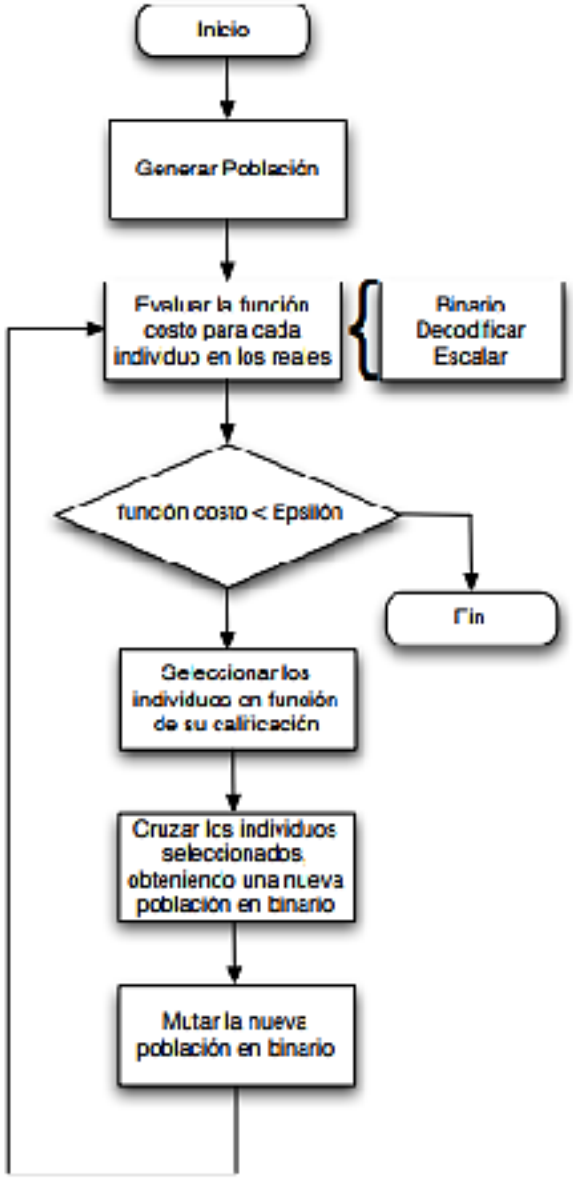


Diagrama de flujo de un AG

- Generar la población inicial
- Evaluar la aptitud de los individuos
- Verificar si cumple el objetivo propuesto por el algoritmo
- Realizar el sorteo de los individuos
- Seleccionar los individuos a cruzar
- Cruzar los individuos seleccionados
- Sustituir los progenitores
- Generar los bits a mutar y mutar los bits correspondientes



Población inicial

Tamaño de la Población

Este parámetro nos indica el número de individuos que tenemos en nuestra población para una generación determinada.

En caso de que esta medida sea insuficiente, el algoritmo genético tiene pocas posibilidades de realizar reproducciones con lo que se realizaría una búsqueda de soluciones escasa y poco óptima.

Por otro lado si la población es excesiva, el algoritmo genético será excesivamente lento.

Población inicial

T=TAMAÑO DE LA MATRIZ POBLACIONAL PARA UNA INCOGNITA

T – N, L

N – cantidad de individuos por generación

L = longitud del cromosoma

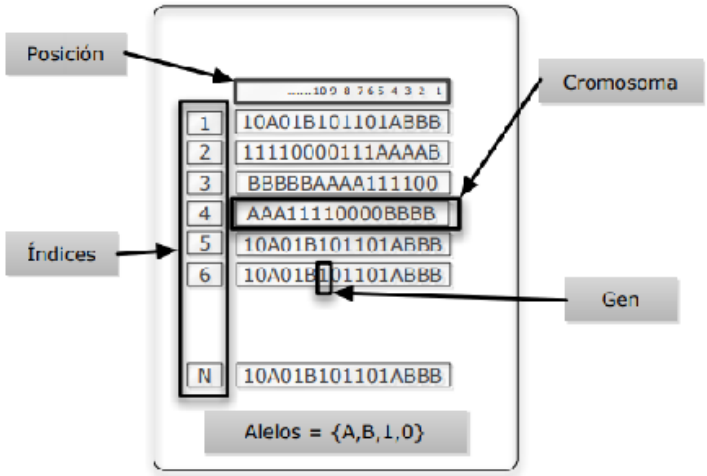
T= TAMAÑO DE LA MATRIZ POBLACIONAL PARA MULTIPLES INCOGNITAS

T = N, (L*V) o T=N, (L1+L2+....+Lm)

N = cantidad de individuos por generación

L = longitud del cromosoma

V= cantidad de variables o incógnitas



Población

Población inicial

La población aleatoria de un AG binario esta definida básicamente por N = número de individuos y l = longitud de cromosomas. Ejemplo: N individuos con una longitud de cromosomas de 10 bits

1	0110111010	$\rightarrow$	442
2	1111000111	$\rightarrow$	967
3	1110101011	$\rightarrow$	939
$\vdots$	$\vdots$	$\vdots$	$\vdots$
N	1111000011	$\rightarrow$	963

Población inicial

Ejemplo: Una población de N individuos, dos variables y una longitud de 6 bits.

	var 1	var 2	var 1	var 2
1	101010	110011	42	51
2	111111	000001	63	1
3	011011	101110	27	46
⋮	⋮	⋮		
N	101111	100111	47	39

Evaluación de los individuos

La aptitud de los individuos se evalúa a partir de:

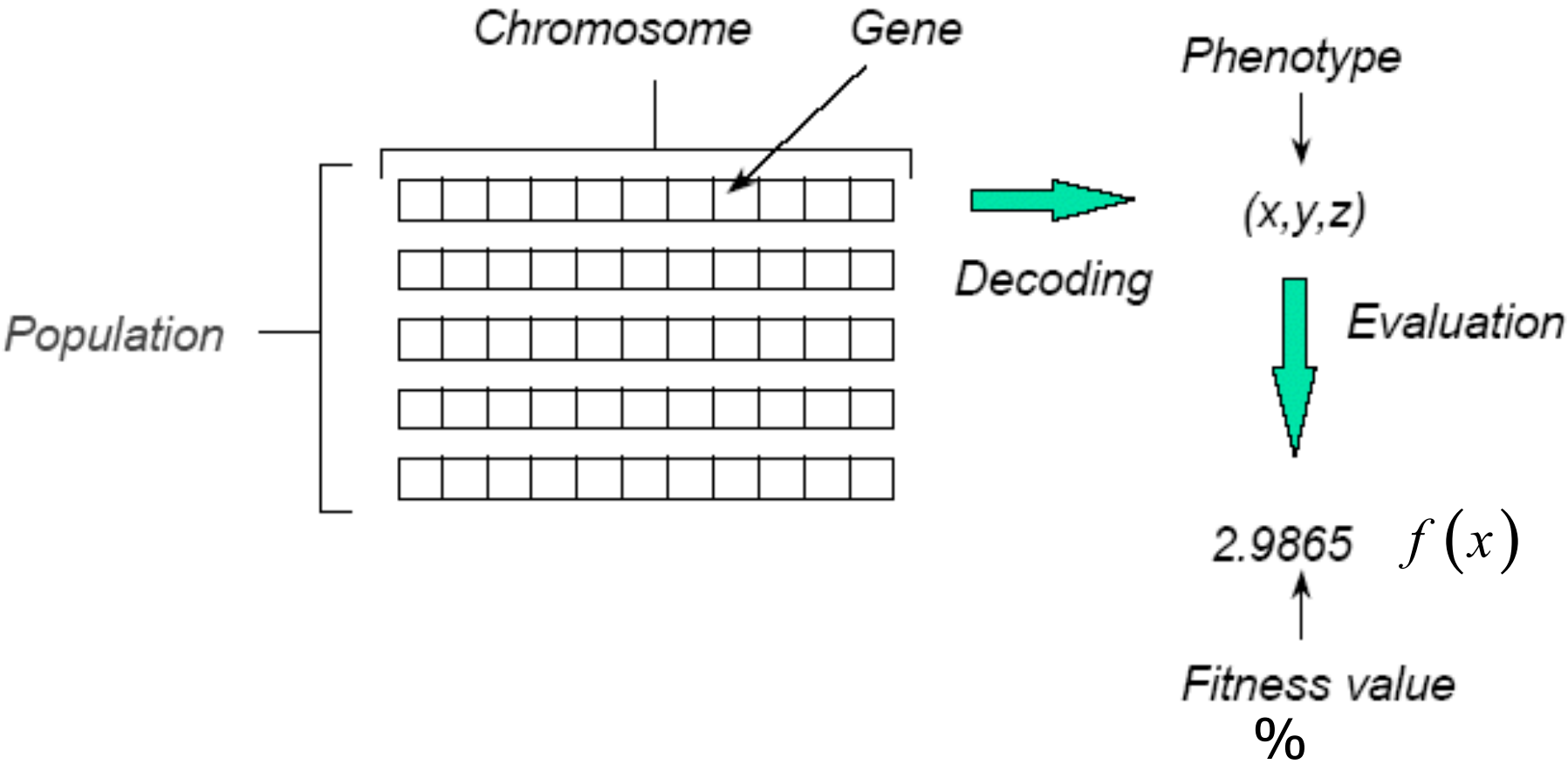
- Fitness: con la función de evaluación,

  Evaluación, o "aptitud bruta" $f(x)$

- Porcentaje fitness: con la función de aptitud,

  Aptitud, o "aptitud neta" $u(x)$

Evaluación de los individuos



Procedimientos básicos de un AG

- El proceso de selección de progenitores se hace

A partir de la población inicial o la población actual, mediante diferentes técnicas de muestreo.

Mecanismos de muestreo de poblaciones

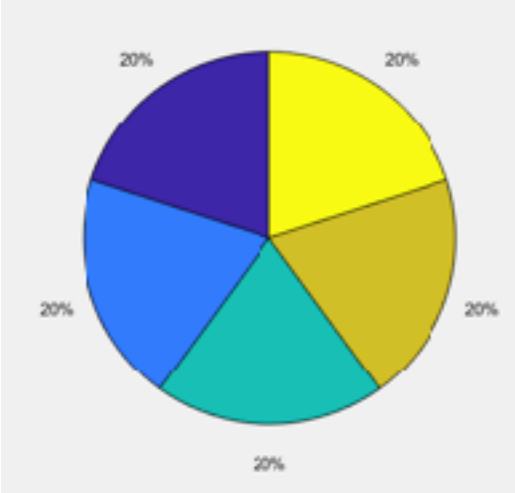
- Selección directo:

  Se toma un subconjunto de individuos de la población siguiendo un criterio fijo: ordenamiento mayor a menor fitness, los k mejores, a dedocracia, etc,…

Mecanismos de muestreo de poblaciones

- Selección aleatoria simple o equiprobable

Se asigna a todos los elementos de la población las mismas probabilidades de formar parte de la muestra.

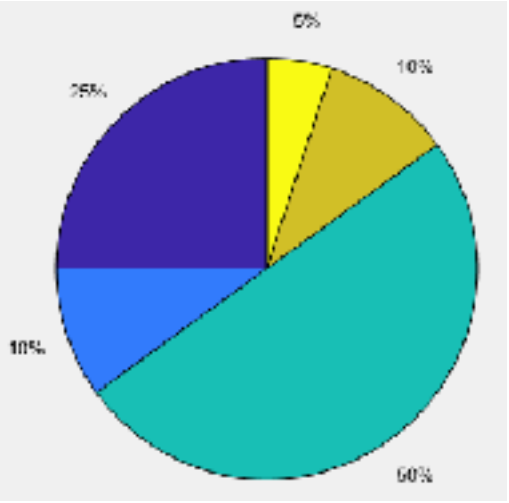


Mecanismos de muestreo de poblaciones

- Selección aleatoria según fitness: (elitismo y aleatoriedad)

Se asignan las probabilidades de selección función de su aptitud.

$$p_i = \frac{u_i}{u_1 + \cdots + u_n}$$

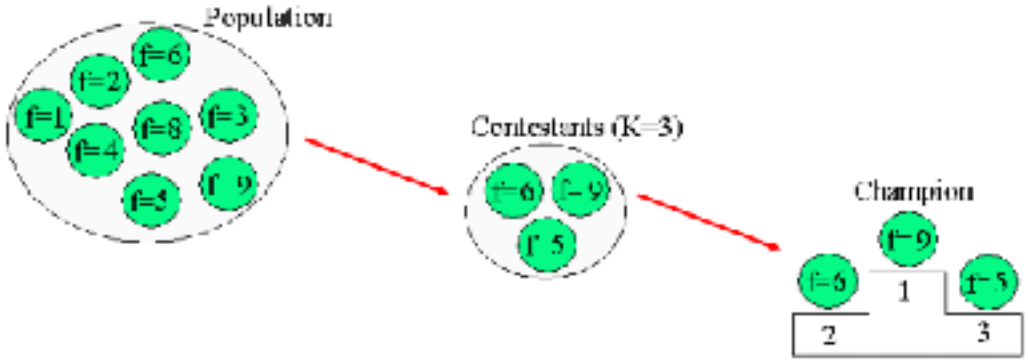


Por defecto, la puntuación p_i, asociada al individuo x_i de la población $P=\{x_1,...,x_n\}$, se calcula como la aptitud relativa de dicho individuo: esto es, siendo $u_1, \ldots, u_n$ las respectivas aptitudes que poseen los individuos.

Mecanismos de muestreo de poblaciones

- Selección por torneo (elitismo y aleatoriedad)

Se escogen de forma aleatoria un número de individuos de la población, y el que tiene puntuación mayor se reproduce, sustituyendo su descendencia al que tiene menor puntuación de la población.



Procedimientos básicos de un AG

La reproducción o crossover

Consiste en el intercambio de material genético entre dos cromosomas de los individuos padres. El objetivo del crossover es conseguir que el descendiente mejore la aptitud de sus padres.

Para aplicar la reproducción habrá que seleccionar con anterioridad dos individuos de la población con una de las diversas técnicas de selección de la etapa anterior.

Procedimientos básicos de un AG

El proceso de reproducción o crossover

Se realiza a partir de la población de progenitores para esto se deben seleccionar algunos miembros y luego aplicar los operadores genéticos de transformación:

- Cruce
- Mutación

Operadores genéticos de recombinación

Cruce o apareamiento

Actúan sobre parejas y originan otro par de individuos que combinan características de los progenitores.

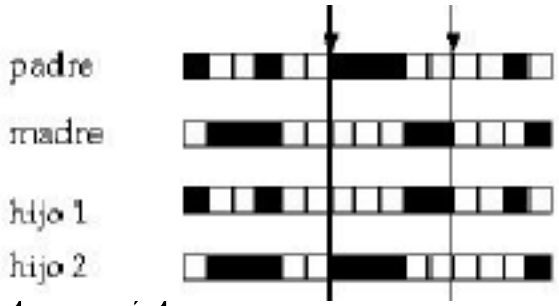
cruce

Operadores genéticos de recombinación

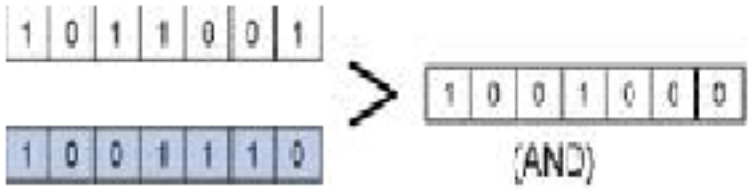
Formas de realizar el cruce de los progenitores

- Cruce a 1 punto

- Cruce a 2 puntos



- Cruce por operador matemático



Operadores genéticos de recombinación

Probabilidad de cruce, Pc:

p_c típicamente entre 0.6 y 0.9

Indica la frecuencia con la que se realizan cruces entre los cromosomas padre.

En caso de que no exista probabilidad de reproducción, los hijos serán copias exactas se los padres.

En caso de haberla, los hijos tendrán partes de los cromosomas de los padres.

Operadores genéticos de alteración

Actúan sobre un individuo, realizando una pequeña modificación a alguno o algunos de sus genes.

Mutación

Actúa luego del cruce sobre cada individuo hijo de forma particular.

Operadores genéticos de alteración

Mutación

Consiste en modificar ciertos genes de forma aleatoria de acuerdo con la probabilidad de mutación establecida Pm.

Las mutaciones son beneficiosas pues contribuyen a la diversidad genética de la especie. Además previenen obtener soluciones que convergen a un óptimo local.

Operadores genéticos de alteración

Probabilidad de mutación

Indica la frecuencia con que los genes de un cromosoma son mutados.

Si es 100% todo el cromosoma es mutado.

p_m típicamente entre
1/tamaño de la población y
1/longitud del cromosoma

Operadores genéticos de alteración

Mutación

- Catastrófica: produce un hijo no viable.
- Neutral: produce un hijo que no altera fitness drásticamente.
- Ventajosa: produce un hijo con característica beneficiosa.

Procedimientos básicos de un AG

El proceso de reemplazo

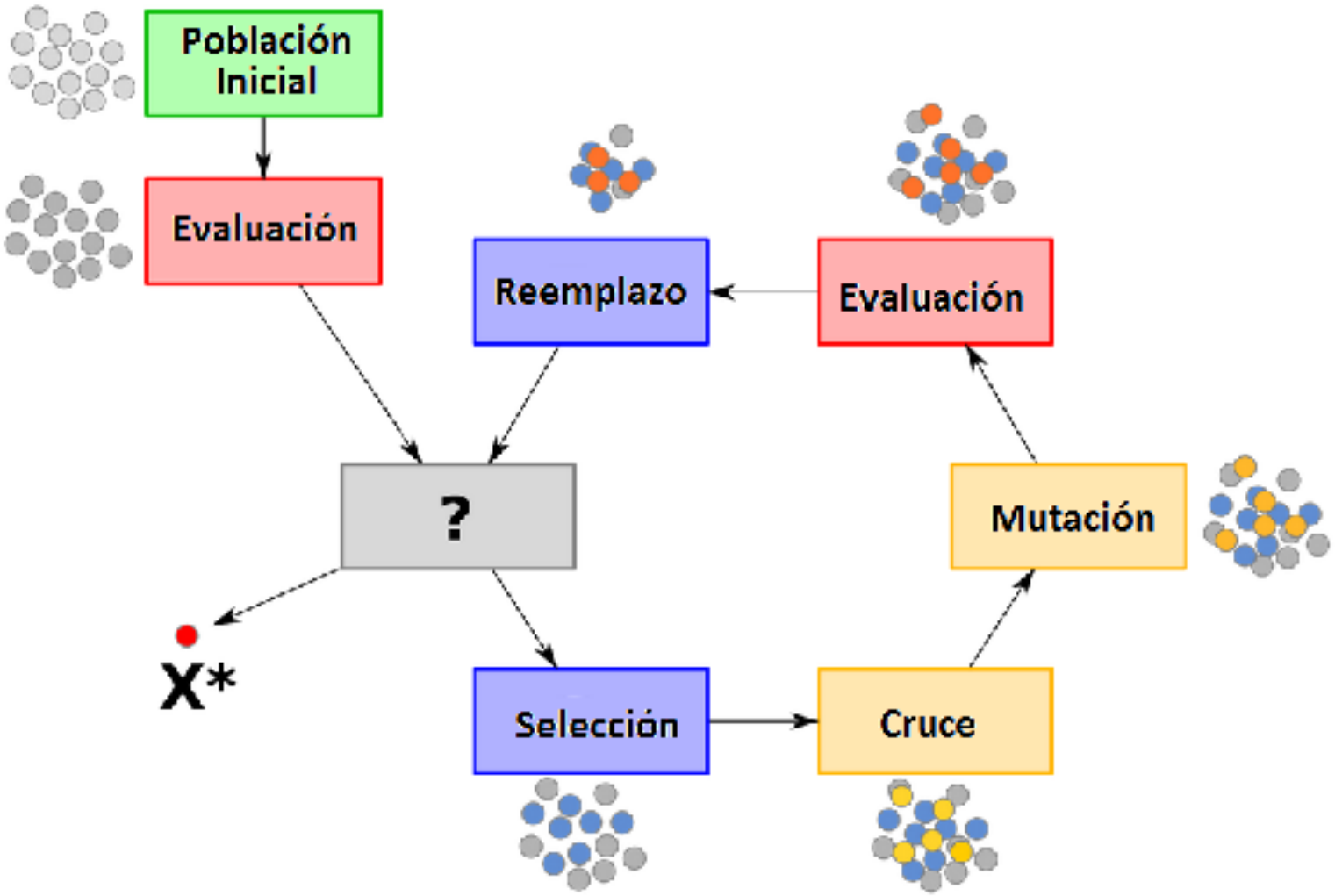
- n: cantidad de progenitores
- s: cantidad de descendientes
- El numero de descendientes *s* puede ser aleatorio
- Para valores grandes de *s* varia más la población de una generación a otra. (muchas características)
- También se puede considerar *s* menor o igual al 60% de *n*. O simplemente s=n.

Procedimientos básicos de un AG

El proceso de remplazo

- Remplazo inmediato de los progenitores (hijos por padres)
- Remplazo por inserción. Si: (para mantener la población)
  - $s < n$ se sustituyen s progenitores de la población.
  - s > n se muestrean n miembros de la población y se descartan s de menor fitness (muestreo entre padres e hijos)
- Remplazo por inclusión se pasa a muestrear los s+n, la población crece.

Bucle básico de un AG



Ejemplo: maximizar la función

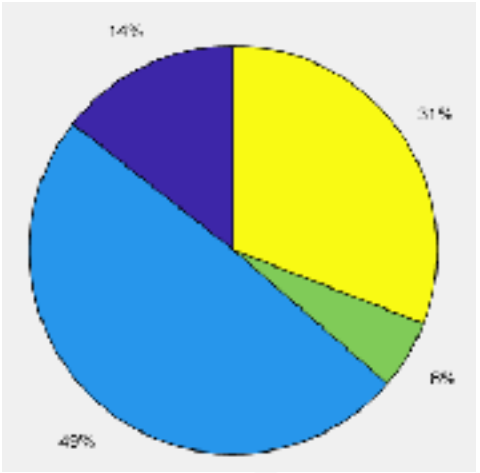
$f(x) = x^2$ sobre los enteros [0..31]

- Generación 1

Fitness bruto del individuo en la resolución del problema

Fitness neto

N°	String	Valor X	Fitness X^2	% del total de fitness
1	01101	13	169	14.4
2	11000	24	576	49.2
3	01000	8	64	5.5
4	10011	19	361	30.9
Total			1170	100.0



- Ejemplo de selección Proporcional al valor del fitness
- Cruce y mutación
- Remplazo

Pseudocódigo de un AG

```
BEGIN /* Algoritmo Genetico Simple */
      Generar una poblacion inicial.
      Computar la funcion de evaluacion de cada individuo.
      WHILE NOT Terminado DO
      BEGIN /* Producir nueva generacion */
            FOR Tamaño poblacion/2 DO
            BEGIN /*Ciclo Reproductivo */
                  Seleccionar dos individuos de la anterior generacion,
                  para el cruce (probabilidad de seleccion proporcional
                  a la funcion de evaluacion del individuo).
                  Cruzar con cierta probabilidad los dos
                  individuos obteniendo dos descendientes.
                  Mutar los dos descendientes con cierta probabilidad.
                  Computar la funcion de evaluacion de los dos
                  descendientes mutados.
                  Insertar los dos descendientes mutados en la nueva generacion.
            END
            IF    la poblacion ha convergido THEN
                  Terminado := TRUE
      END
END
```

Como construir el AG

Decodificación de una población y escalización

"hallar el fenotipo a partir del genotipo con mayor o menor resolución de conversión"

Escalar la población en decimal P_{10} en un intervalo de búsqueda $[I_{j,min}, I_{j,max}]$

$$P_{i,j} = (I_{j,max} - I_{j,min})\frac{P_{i,j}}{2^L - 1} + I_{j,min} \qquad (3)$$

Donde $i = 1, 2, \ldots, N$ es i-ésimo individuo de la población, j es la dimensión de la función a optimizar y N es el número máximo de individuos que contiene la población. $P_{i,j}$ es un número decodificado en decimal a partir de los cromosomas de la población.

¿Cómo construir el AG?

TAREA: Decodificación de una población y escalización.

Utilizando la ecuación de decodificación y escalización binaria para algoritmos genéticos. Usted debe realizar en Matlab un algoritmo para los siguientes ejercicios:

1. Generar una matriz de población binaria aleatoria en R^2(dos variables) de 12 individuos en un intervalo de búsqueda de [-0.5 a 1.3 y -3 a 1]. Suponga una longitud de cromosoma de 12 bits. Decodificar y escalizar.
2. Generar una matriz de población binaria aleatoria en R^3(tres variables) de 50 individuos en un intervalo de búsqueda de [-5 a 5 , -1 a 1 y 2 a 8.5]. Suponga una longitud de cromosoma de 8 bits. Decodificar y escalizar.

Algoritmo de ejemplo

- Generar la población inicial
- Evaluar la aptitud de los individuos
- Realizar el sorteo de los individuos
- Seleccionar los individuos a cruzar
- Cruzar los individuos seleccionados
- Sustituir los progenitores
- Generar los bits a mutar
- Mutar los bits correspondientes

Ejercicio: Encontrar el valor que minimiza la función $f(x) = x^2$ en el intervalo [-30,50] utilizando Matlab.

Algoritmo de ejemplo

Función principal – Parámetros del AG

```
%% Algoritmo Genetico basico
% Encontrar el valor que minimiza la funcion
% FUN = x^2 en el intervalo [-30,50]
  clc;
  clear all;
  close all;

%% Definicion de los parametros del algoritmo
   L_cromosoma    = 10;        % longitud del cromosoma
   N_poblacion    = 50;        % tamaño de la poblacion
   Pc_cruce       = 0.7;       % probabilidad de crossover
   Pm_mutacion    = 0.02;      % probabilidad de mutacion
   Generaciones   = 50;        % maxima cantidad de generaciones (ciclos de ejecucion)
```

Algoritmo de ejemplo

Función principal – Función a minimizar

```
%% Definición de la funcion a hallar el minimo

%  Definicion del intervalo de la variable de diseño
   MinInterv =  30;
   MaxInterv =  50;

%  Visualizar la funcion FUNCION en el intervalo indicado
   x     = MinInterv:0.1:MaxInterv;
   y     = A_funcion(x);

   figure(1);
   plot(x,y);
   grid on;
   title('Funcion objetivo F(x) = x^2');
```

Algoritmo de ejemplo

Sub-funciones – función a minimizar

```
function y = A_funcion(x)

    y = x .* x;

end
```

Sub-funciones – crear población inicial

```
function[Pob] = A_poblacion(L,N)
    % N = Numerodeindividuos
    % L = Longituddelcromosoma

    Pob = round(rand(L,N));
end
```

Algoritmo de ejemplo

Función principal – Función de creación de población inicial

```
%% Construir la poblacion inicial
   Pob = B_poblacion(L_cromosoma, N_poblacion);
```

Algoritmo de ejemplo

Función principal – Bucle central de repetición

```
for gen 1:Generaciones
    % Evaluacion de la nueva generacion
      FenoLipos    = C_GenoLipos_FenoLipos(Pob, MinVar, MaxVar);
      Salida       = A_Funcion(FenoLipos);        % Fitness bruto
      Fitness      = 1./Salida;                   % funcion porcentual
                                                  % grande el fitness indica
                                                  % que se converge a una
                                                  % solucion que minimiza x^2

    % fitness promedio
      FitMax = max(Fitness);                      % valor maximo de fitness
      FitAVG = sum(Fitness)/N poblacion;          % fitness promedio
      EvalFitness(gen,:) = [FitMax; FitAVG];% vector que guarda los datos
                                                  % de fitnes para cada generacion
    % plot de la nueva generacion con las nuevas soluciones sobre la curva
      hold on;
      plot(x,y);
      title('Funcion objetivo F(x)    x^2');
      hold on;
      plot(FenoLipos,Salida,'*b');
      pause(0.5);

    % Creacion de una nueva generacion
      Pob = U_generar(Pob, Fitness, Pc_cruce, Pm_mutacion);
end
```

Algoritmo de ejemplo

Función principal – Gráficos de comportamiento generacional

```
%% Comportamiento de la poblacion a traves de las generaciones
   h2 = figure(2);
   plot (EvalFitness,'*')
   axis([1 gen 0 max(EvalFitness(:))*1.1 ])
   title('Capacidad de adaptación de la población')
   xlabel('generación')
   legend('FitMax','FitAVG', 'Location','SouthEast')

%% Adaptacion de la poblacion final
   [Respuesta,Ind] = sort(Fitness,'descend');
   figure(1)
   Mejores = [Fenotipos(Ind(1)); Fenotipos(Ind(2)); Fenotipos(Ind(3))];
   Salida = A_funcion(Mejores);
   plot(Mejores(1:3),Salida(1:3),'r*')
   legend('función','población','solución')
```

Algoritmo de ejemplo

Sub-funciones – obtención de fenotipo

```
function Fen = C_Genotipo_Fenotipo(Pop, MinInterv, MaxInterv)
    % Esta funcion convierte la matriz Pop, cuyas columnas son cromosomas,
    % en un vector fila con los fenotipos (puntos donde debe evaluarse la funcion)

    [L, N] = size(Pop);                          % obtencion de tamaño de poblacion
    Fen = [];                                    % vector vacio
    numero   = bi2de(Pop','left-msb');           % conversion de vectores izq a der
    ValorMax = 2^L - 1;                          % valor maximo en binario con 10 bits
    Fen = MinInterv + (numero./ValorMax) .* (MaxInterv - MinInterv); % escalizar
    Fen = Fen';

end
```

Algoritmo de ejemplo

Sub-funciones – Generar la nueva generación

```
function y = D_generar(Pob, Fitness, Pc_cruce, Pm_mutacion)
% Genera una nueva poblacion de individuos a partir de Pob
% utilizando seleccion proporcional y con probabilidad de crossover
% y probabilidad de mutacion

[l, N] = size(Pob);
col = round(N / 2);          % por cada iteracion se generan dos hijos

y = [];                      % vector vacio

for i=1:1:col
    % seleccionar dos padres
      padre1 = F_select(Fitness);
      padre2 = F_select(Fitness);

    % generar los dos hijos
      [hijo1, hijo2] = F_cruzar_y_mutar(Pob,padre1,padre2,Pc_cruce,Pm_mutacion);

       y = [y, hijo1, hijo2];
end
```

Algoritmo de ejemplo

Sub-funciones – Selección de padres

```
function y = P_select(Fitness)

% Dado un vector fila con los fitness de los individuos de la poblacion
% selecciona utilizando SELECCION PROPORCIONAL

  cuantos    = length(Fitness);        % longitud poblacion
  sumFitness = sum(Fitness);           % suma de todos los fitness
  aleatorio  = rand * sumFitness;      % umbral aleatorio a sobrepasar

  %buscando el slot correspondiente
  suma = Fitness(1);
  j = 1;
while (suma < aleatorio) & (j<cuantos) % seleccion con umbral aleatorio
      j = j + 1;
      suma = suma + Fitness(j);
  end
  %retorna el individuo elegido
  y = j;
end
```

Algoritmo de ejemplo

Sub-funciones –

Cruce y mutación

```
function [hijo1, hijo2] = F_cruzar_y_mutar(Pob,padre1,padre2,Pc_cruce,Pm_mutacion)
% Aplica crossover y mutacion a las columnas PADRE1 y PADRE2 de la poblacion
% y calcula dos vectores columna con ambos hijos
[L, N] = size(Pob);
if padre1>N, padre1=N; end              % restriccion de seguridad al padre
if padre2>N, padre2=N; end              % cuando supera N-50

%ver si corresponde aplicar crossover o no
hayCrossover = ((Pc_cruce==1)  | (rand <= Pc_cruce)); % operador logico OR
hijo1=zeros(L,1);
hijo2=zeros(L,1);

if hayCrossover    1                                  % CRUCE A UN PUNTO
    hijo1 = [  Pob(1:5, padre1); Pob(6:10, padre2) ];
    hijo2    [  Pob(1:5, padre2); Pob(6:10, padre1) ];
else
    hijo1   Pob(:,padre1);
    hijo2 = Pob(:,padre2);
end

if hayCrossover    1                                  % MUTACION DE LOS HIJOS
    for i=1:1:L
        hijo1(i,1)   F_mutar(hijo1(i,1),Pm_mutacion);
        hijo2(i,1) = F_mutar(hijo2(i,1),Pm_mutacion);
    end
end
end
```

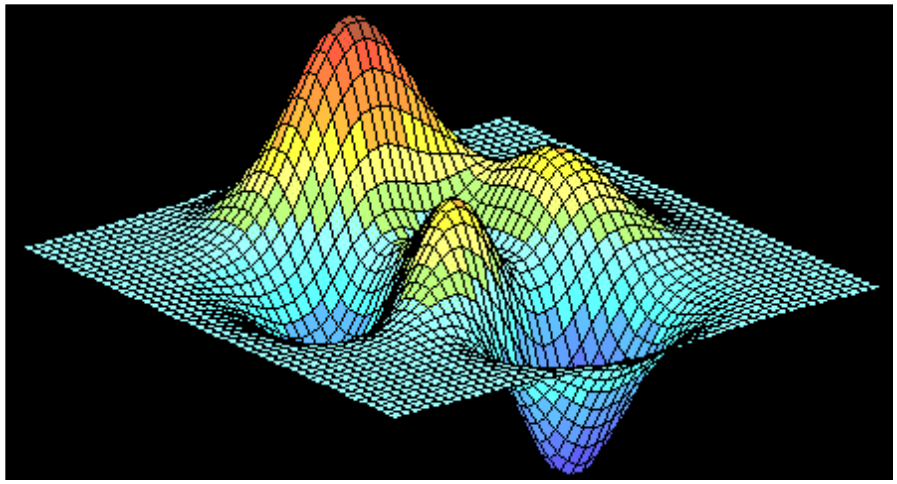
Algoritmo de ejemplo

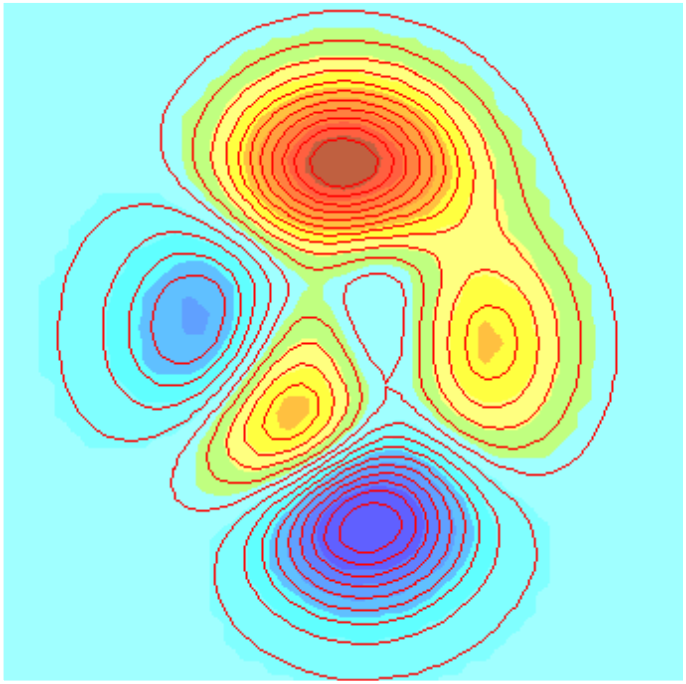
Sub-funciones – mutación

```
function NewAlelo = M_mutar( Alelo, probabilidad)

%muta el ALELO con la probabilidad indicada

if (probabilidad==1) || (rand<=probabilidad)
    % hay mutacion
    NewAlelo = ~Alelo;  % cambia el estado del bit cuando esta dentro de la probabilidad
else
    NewAlelo = Alelo;   % no cambia porque esta fuera de la probabilidad de mutacion.
end
```

Ejemplo: maximizar la funcion *peaks*

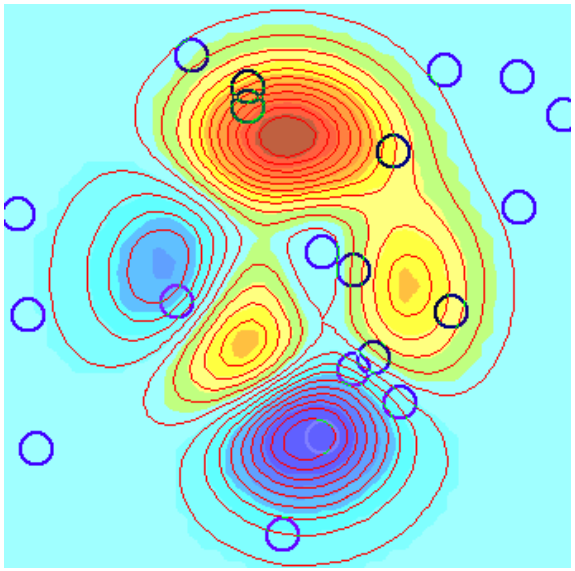
- Encontrar el max. de la funcion "peaks"
- z = f(x, y) = 3*(1-x)^2*exp(-(x^2) - (y+1)^2) - 10*(x/5 - x^3 - y^5)*exp(-x^2-y^2) -1/3*exp(-(x+1)^2 - y^2).



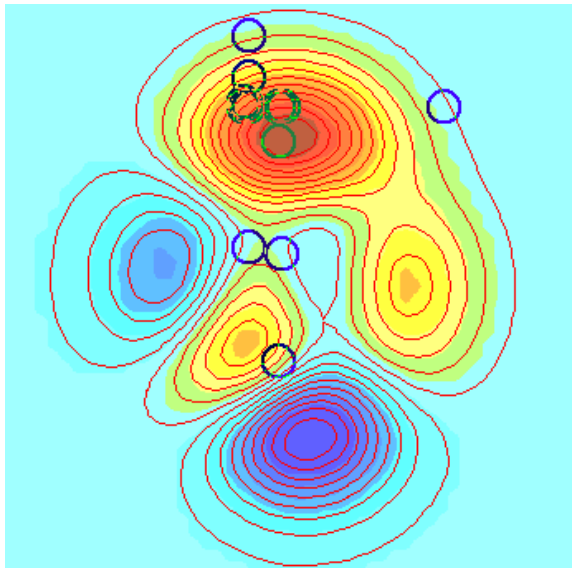


Ejemplo: maximizar la funcion *peaks*

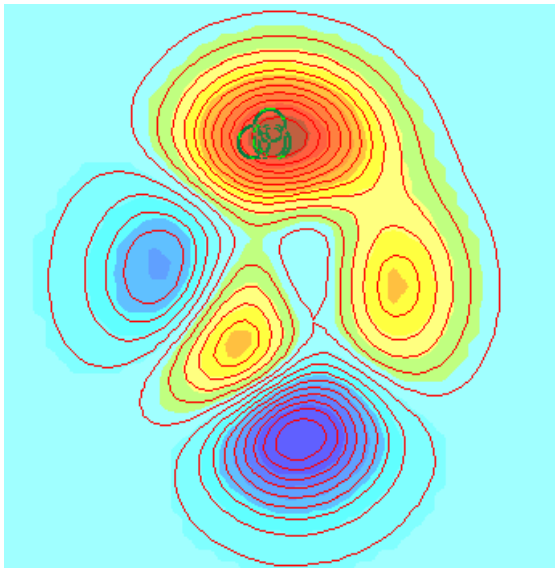
- Proceso aplicando Algoritmos Genéticos:



poblacion Inicial



5a generacion



10a generacion

Ventajas de un AG

Implementación computacional. Las operaciones a realizar en un AG es a través de operaciones aritméticas, lógicas y de ordenamiento.

Información del Sistema. No necesitan información a priori. El AG genera multiples soluciones de forma aleatoria y si algunas de ellas mejoran, entonces, son soluciones que se tomarán en cuenta para evolucionar la población.

Desventajas de un AG

Funcion costo. La única forma para evaluar el desempeño de los individuos en las evoluciones del AG es a través de una función de evaluación o una colección de datos.

Reglas. No existen reglas para determinar el número de individuos en una población, que tipo de selección aplicar o como realizar la mutación.

Programación serie. Cuando los AG se programan en plataformas de procesamiento serie, no son tan rápidos en su tiempo de ejecución y por lo tanto, es difícil implementarlos en aplicaciones donde es necesario realizar ajustes en tiempo real o en línea.

Preguntas

Gracias…