

# Programación Evolutiva

## Práctica 1

Optimización de funciones

---

Carlos Cervigón Rückauer  
Facultad de Informática  
U.C.M

# PE: Práctica 1

El objetivo de esta práctica es implementar un algoritmo genético clásico para hallar el máximo o mínimo de diferentes funciones de forma evolutiva.

**Función de prueba:**  $f(x_1, x_2) = 21.5 + x_1 \cdot \text{sen}(4\pi x_1) + x_2 \cdot \text{sen}(20\pi x_2) :$

que presenta un máximo de 38.809 en 11.625 y 5.726

$$x_1 \in [-3.0, 12.1]$$

$$x_2 \in [4.1, 5.8]$$



# El algoritmo

```
class AlgoritmoGenetico {  
  
    private int tamPoblacion;  
    private Individuo[] poblacion;  
    private double[] fitness;  
    private int maxGeneraciones;  
    private double probCruce;  
    private double probMutacion;  
    private int tamTorneo;  
    private Individuo elMejor;  
    private int pos_mejor;  
  
    . . .  
}
```



# El algoritmo

```
run() {  
    Iniciar poblacion  
    Evaluar población  
    Para cada generación {  
        //Selección  
        //Cruce  
        //Mutación  
        evaluar población  
    }  
    Devolver mejor  
}
```



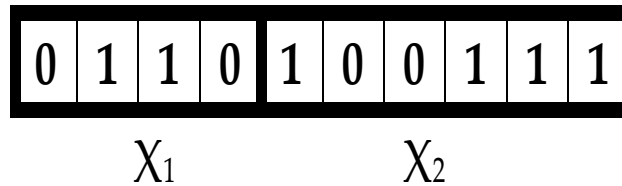
# La representación

## Función 1:

que presenta un máximo de 38.809 en 11.625 y 5.726

$$x_1 \in [-3.0, 12.1]$$

$$x_2 \in [4.1, 5.8]$$



# La representación

$$f(x_1, x_2) = 21.5 + x_1.\text{sen}(4\pi x_1) + x_2.\text{sen}(20\pi x_2) :$$

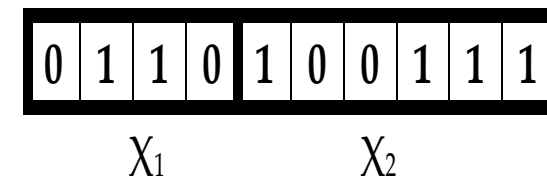
Las posibles soluciones serán valores de  $x_1$  y  $x_2$ . cada uno en su intervalo:

$$x_1 \in [-3.0, 12.1]$$

$$x_2 \in [4.1, 5.8]$$

Nuestro cromosoma o individuo será un **array de genes** ( $x_1, x_2, \dots$ ) y cada gen se codifica como una array de bits (en otros problemas cada gen podría ser un array de enteros, double, etc..)

Lo primero es calcular la longitud del cromosoma, según el número de valores que queremos representar y la precisión



# La representación

```
public abstract class Individuo<T> {  
    T[] cromosoma;  
    int[] tamGenes;  
    . . .  
    . . .  
}
```



# La representación

```
public class IndividuoFuncion1 extends Individuo<Boolean>{  
//public class IndividuoFuncion5 extends Individuo<Double>  
    public IndividuoFuncion1() {  
        this.tamGenes = new int[2];  
        this.min = new double[2];  
        this.max = new double[2];  
        this.min[0] = -3.000;  
        this.min[1] = 4.100;  
        this.max[0] = 12.100;  
        this.max[1] = 5.800;  
        this.tamGenes[0] = this.tamGen(this.valorError, min[0], max[0]);  
        this.tamGenes[1] = this.tamGen(this.valorError, min[1], max[1]);  
        int tamTotal = tamGenes[0] + tamGenes[1];  
        this.cromosoma = new Boolean[tamTotal];  
        for(int i = 0; i < tamTotal; i++) this.cromosoma[i] = this.rand.nextBoolean();}
```

0	1	1	0	1	0	0	1	1	1
---	---	---	---	---	---	---	---	---	---

$\chi_1$

$\chi_2$

$x_1 \in [-3.0, 12.1]$

$x_2 \in [4.1, 5.8]$

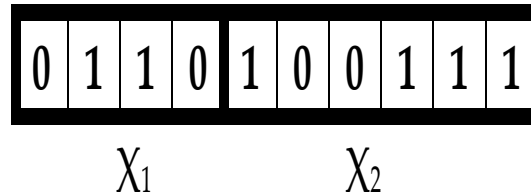




# La representación

```
public class IndividuoFuncion1 extends Individuo<Boolean>{  
    . . .  
    public double getValor() {  
        double x1 = this.getFenotipo(0), x2 = this.getFenotipo(1);  
        return (21.5 + x1 * Math.sin(4 * Math.PI * x1) + x2 * Math.sin(20 * Math.PI * x2));  
    }  
}
```

```
public double getFitness() {  
    return this.getValor();  
}
```



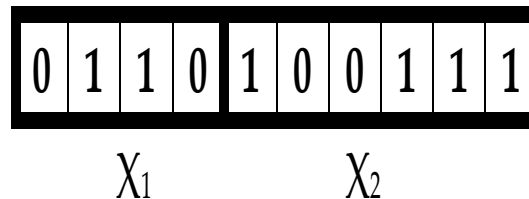
$$x(\mathbf{v}) = x_{min} + \text{bin2dec}(\mathbf{v}) \cdot \frac{x_{max} - x_{min}}{2^{l_{chrom}} - 1}$$



# La representación

- El tamaño del cromosoma será la suma del tamaño de los genes
- El tamaño de cada gen se puede calcular:

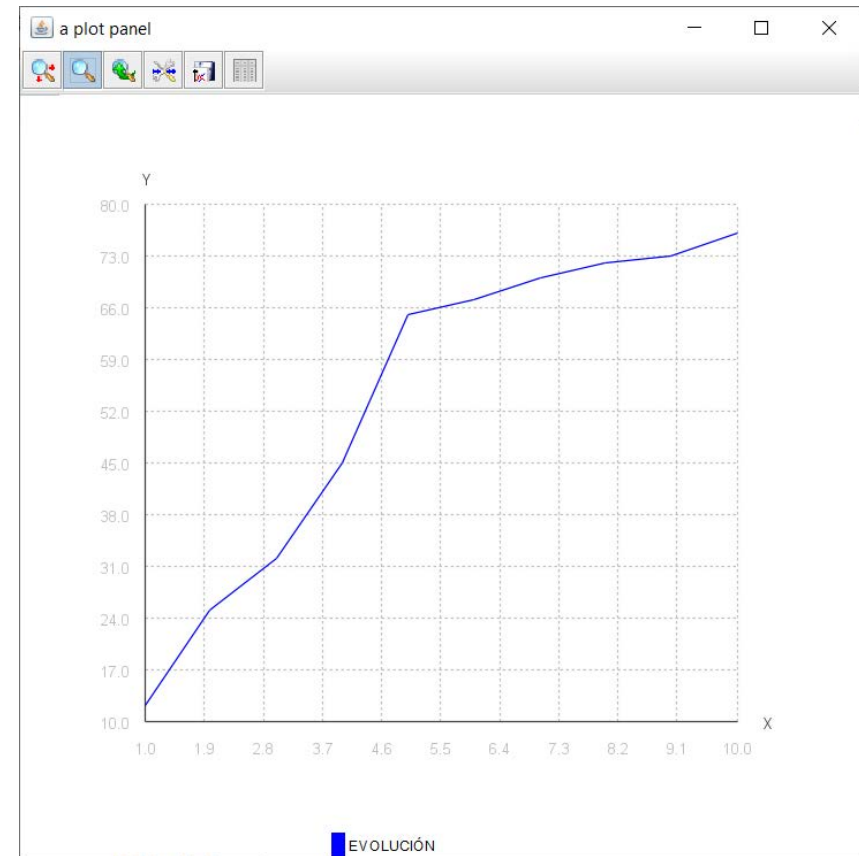
```
public int tamGen(double valorError, double min, double max) {  
    return (int) (Math.Log10(((max - min) / precision) + 1) / Math.Log10(2));  
}
```



# Gráfica de evolución

```
import javax.swing.*;
import org.math.plot.*;

public class Pruebaplot{
    public static void main(String[] args) {
        // define your data
        double[] generaciones = { 1, 2, 3, 4, 5, 6, 7, 8, 9 ,10 };
        double[] fitness = { 12, 25, 32, 45, 65, 67 , 70, 72, 73, 76};
        // create your PlotPanel (you can use it as a JPanel)
        Plot2DPanel plot = new Plot2DPanel();
        // define the legend position
        plot.addLegend("SOUTH");
        // add a line plot to the PlotPanel
        plot.addLinePlot("EVOLUCIÓN", generaciones, fitness);
        // put the PlotPanel in a JFrame like a JPanel
        JFrame frame = new JFrame("a plot panel");
        frame.setSize(600, 600);
        frame.setContentPane(plot);
        frame.setVisible(true);
    }
}
```






# Acerca de *Creative Commons*



- Licencia CC ([Creative Commons](https://creativecommons.org/licenses/by-nc-sa/4.0/))

Este tipo de licencias ofrecen algunos derechos a terceras personas bajo ciertas condiciones.

Este documento tiene establecidas las siguientes:

-  Reconocimiento (*Attribution*):  
En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.
-  No comercial (*Non commercial*):  
La explotación de la obra queda limitada a usos no comerciales.
-  Compartir igual (*Share alike*):  
La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

Pulsa en la imagen de arriba a la derecha para saber más.

