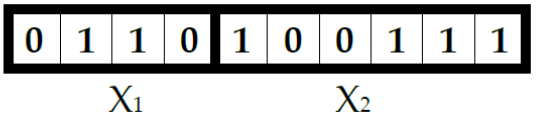
Programación evolutiva
Facultad de Informática U.C.M.
Curso 2021/2022

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.

Diseño del algoritmo:

- **Representación de los individuos**: se representan mediante cadenas binarias que se corresponden con los puntos del espacio de búsqueda. La cadena binaria codificará los valores de las variables que utilice cada función. Por ejemplo, en un problema con una función de dos variables X_1, X_2 se utilizará un cromosoma similar a éste:



- **Función de evaluación**: el fitness es el resultado de evaluar la función considerada en el punto que resulta de la decodificación del individuo, por ejemplo $f(x_1, x_2)$
- **Selección**: por ruleta, 2 torneos, estocástico universal, truncamiento y restos.
- Se incluirá la opción para seleccionar **elitismo**.
- **Operadores: cruce** (monopunto y uniforme) y **mutación** básica.
- **Resultados: v**alores obtenidos por el algoritmo y gráficas de evolución

Consideramos la optimización de las siguientes funciones:

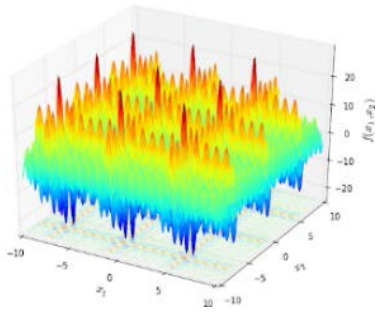
- **Función 1:**

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

que presenta un máximo de **38.809** en 11.625 y 5.726 $x_1 \in [-3.0,12.1]$ $x_2 \in [4.1,5.8]$

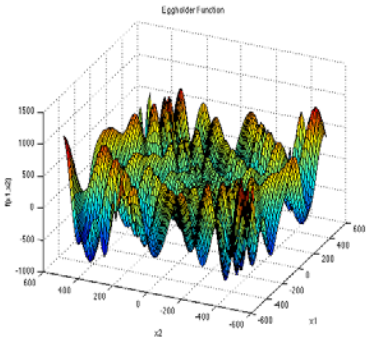
- **Función 2: Schubert**

$$f(x_i, i = 1..2) = \left(\sum_{i=1}^{5} i.\cos((i+1)x_1 + i)\right)\left(\sum_{i=1}^{5} i.\cos((i+1)x_2 + i)\right)$$



$x_i \in [-10,10]$ que presenta 18 mínimos de **-186.7309**

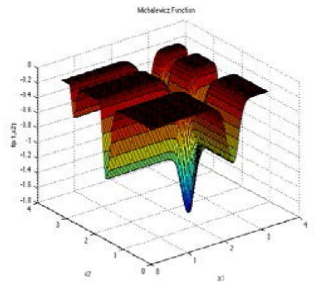
- **Función 3: EggHolder**



$$f(\mathbf{x}) = -(x_2 + 47)\sin\left(\sqrt{\left|x_2 + \frac{x_1}{2} + 47\right|}\right) - x_1 \sin\left(\sqrt{|x_1 - (x_2 + 47)|}\right)$$

que presenta un mínimo de -959.6407 en (512, 404.2319) x1, x2 ∈ [-512, 512]

- **Función 4: Michalewicz**



$$f(x_i|i = 1..n) = -\sum_{i=1}^{n} \text{sen}(x_i)\text{sen}^{20}\left(\frac{(i+1)x_i^2}{\pi}\right) : x_i \in [0,\pi]$$

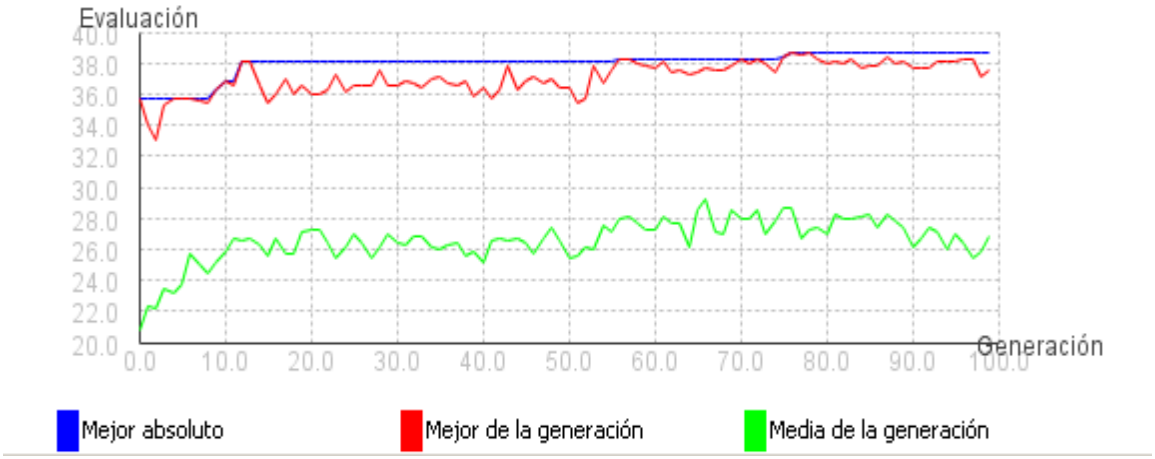
$x_i \in [0,\pi]$ que presenta los siguientes mínimos en función de ***n***:

n	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Mínimo	-1	-1.959091	-2.897553	-3.886358	-4.886358	-5.879585	-6.862457

Parámetros del algoritmo: La aplicación tendrá una interfaz de usuario que permita variar los parámetros interactivamente. Los parámetros predeterminados son:

- Tamaño de la población (100)
- Número de generaciones (100)
- Porcentaje de cruces (60%)
- Porcentaje de mutaciones (5%)
- Precisión o valor de error para la discretización del intervalo (0.001)
- Método Selección
- Método cruce
- Método Mutación
- Posibilidad de seleccionar porcentaje de elitismo

- **Representación gráfica de evolución:** Se mostrarán 3 gráficas:
  - Color **rojo**: mejor valor de aptitud obtenido en cada generación.
  - Color **azul**: mejor valor absoluto de aptitud obtenido hasta el momento en cada generación.
  - Color **verde**: Aptitud media. Se puede utilizar cualquier herramienta para hacer las gráficas: ***jmathtools, jfreechart***.... Además de las gráficas se mostrarán los resultados obtenidos (valor óptimo de la función y puntos donde se obtiene).



Ampliamos la práctica para que la **función 4 también soporte cromosomas con representación real**. Ahora el cromosoma puede estar formado por números reales. En número de variables a utilizar ***n*** será un valor seleccionable en la interfaz. Por ejemplo, para **n = 6** variables, el cromosoma podría ser:

3.1241	2.7112	2.3454	0.3425	1.6832	2.9342
x_1	x_2	x_3	x_4	x_5	x_6

Como operadores de cruce hay que implementar:

- Cruce Monopunto
- Cruce Discreto Uniforme
- Cruce Aritmético y BLX-α
- Mutación Uniforme.

Entrega

- **Plazo de entrega: 11 de marzo a a las 16:00**. Debes entregar mediante la tarea de entrega del Campus Virtual un archivo comprimido con el código java de la aplicación (**proyecto en Eclipse** o **NetBeans)** que incluya una breve memoria que contenga el estudio de las gráficas y los resultados obtenidos con cada función. Aquí se valorarán las conclusiones y observaciones que se consideren interesantes respecto al resultado obtenido.

- No olvidéis nombrar correctamente el proyecto e incluir en el código todas las librerías necesarias. El archivo comprimido y el nombre del proyecto Eclipse tienen que ser **GXXP01**, donde **XX** es el número de grupo. Ejemplo nombre del proyecto-archivo: **G01P1** (por ejemplo, para el grupo 01).

- Se valorará positivamente la inclusión de 2 funciones extra a seleccionar entre las que aparecen mostradas aquí:
  https://www.sfu.ca/~ssurjano/optimization.html

- La memoria debe ser breve, pero deberá incluir al menos lo siguiente:
  - Portada con el nombre de los componentes del grupo
  - Una gráfica representativa de cada ejecución, de cada función (seleccionar la mejor, con los parámetros que quieras).
  - Conclusiones de los resultados: como afecta el elitismo, con que método de selección se comporta mejor, si habéis incluido mejoras..., análisis de la convergencia, problemas o curiosidades encontradas, etc....
  - Podéis describir brevemente algunos detalles de la implementación, arquitectura de la aplicación, organización e incluso una breve guía de uso.
  - Al final de la memoria una breve descripción del **reparto de tareas** para reflejar lo que ha hecho cada miembro del grupo.

- La **corrección** incluirá sesiones de laboratorio para evaluar la práctica y responder las preguntas del profesor o **un test de evaluación de la práctica**. La corrección podrá realizarse a los dos miembros del grupo a la vez o individualmente, según lo decida el profesor. Es importante conocer bien la práctica y los aspectos teóricos en los que se basa, pues es lo que determina la calificación final.

- Para aprobar la práctica es requisito que funcionen correctamente todas las funciones y que implemente todas las opciones pedidas. Se pueden incluir en la práctica cualquiera de las mejoras vistas en clase.

- Durante el curso se realizará control de copias de todas las prácticas, comparando las entregas de todos los grupos de PE. Se considera copia la reproducción total o parcial del código de otros alumnos o cualquier código extraído de Internet o de cualquier otra fuente, salvo aquellas autorizadas explícitamente por el profesor. Si se detecta algún tipo de copia sin justificar se calificará como suspenso.

- **Si lo has pactado previamente con el profesor y utilizas un lenguaje distinto a Java, debes añadir claramente las instrucciones de ejecución y prueba de la práctica**.