Programación Evolutiva

Memoria de la práctica 1

David Alfonso Starry González

Daniel Pizarro Gallego

2024

Índice:

1. Arquitectura de la aplicación:

La aplicación se aplica modelo vista controlador. Al ejecutar el programa, con la clase **Main** se inicializa la interfaz usando la clase **MainWindow**, que extiende a JFrame, y añade el controlador.

El controlador es la clase **ControlPanel** que extiende a JPanel, para crear 2 paneles dentro del principal.

- En el panel izquierdo se introducen los valores para las variables, mediante componentes de JSwing, como JButton, JTextField o JComboBox.
- El panel derecho imprime el gráfico 2D cuando termina una ejecución. Este gráfico tiene 2 variables. El eje x es el número de generaciones, y el eje y el valor fitness, de 1. el mejor absoluto (azul), 2. el mejor de la generación (rojo) y 3. la media de la generación (verde).

Una vez pulsado el botón de ejecutar, se ejecuta el algoritmo genético usando la clase **AlgoritmoGenetico**. En esta clase se guardan los valores almacenados en la interfaz, mediante una clase que almacena los datos, llamada **Valores**.

Los individuos y funciones las implementamos con un método de factorías para heredar los valores de sus respectivos padres.

- Los individuos se generan en la clase padre **Individuo**, almacenando los fenotipos de los genes, y su fitness. Se divide entre binarios (**IndividuoBin**) para las cuatro primeras funciones. Este individuo tiene un array de **Gen**, para almacenar los valores binarios de cada gen. Y para los reales (**IndividuoReal)** en la quinta función.
- Las funciones se generan con la clase padre **Funcion**, y dentro de la misma están las cinco clases hijas. Cada una tiene su propia función de evaluación, función compara mejor y función compara peor, para el desplazamiento de maximización y minimización. También los valores de intervalos máximos y mínimos de los fenotipos de la población y los valores máximos y mínimos de los valores fitness para así poder acotar el gráfico con los resultados.

Los métodos de selección se implementan en la clase **Seleccion**. Se aplican igual para los dos individuos. Estos son:

- Ruleta: Selección aleatoria con la probabilidad acumulada de su fitness.
- Torneo determinístico: Se eligen 'k' individuos de la población de forma aleatoria y se elige el mejor. Este proceso se repite hasta llenar la población.
- Torneo probabilístico: Igual que el anterior, pero se elige peor o mejor (aleatoriamente)
- Estocástico universal (2 métodos):
  - Método1:
  - Método2:
- Truncamiento: Se ordenan por fitness y con el porcentaje 'trunc' se eligen los mejores, 1/trunc veces.
- Restos: Las probabilidades acumuladas se multiplican por 'k', y se seleccionan este número redondeado para abajo veces, y los que falten con otro método.

Los métodos de cruce se implementan en la clase **Cruce**. Son cuatro métodos en total. De los cuales los dos primeros se pueden aplicar a los individuos binarios, a los reales se pueden aplicar los cuatro.

- Mono-Punto: Con la probabilidad dada, se cruzan si el numero aleatorio es menor. Para los binarios se corta en un punto aleatorio entre los alelos del individuo. Los reales no tienen alelos binarios por lo que el corte es entre los genes. Una vez elegido el punto de corte se generan dos hijos; cada hijo se genera con una mitad diferente de sus padres.
- Uniforme: Se recorre el individuo entero. En los binarios se intercambian alelos si la probabilidad es menor al numero aleatorio. En los reales genes.
- Aritmético: Con la probabilidad dada, se cruzan si el numero aleatorio es menor. Se hace una media aritmética con los genes, pero usando un valor α para que los hijos no sean idénticos. (αp1i + (1-α)p2i)
- BLX-α: Con la probabilidad dada, se cruzan si el numero aleatorio es menor. Para cada gen de los hijos, se calcula un intervalo con los genes de los padres, y para sus dos hijos se generan números aleatorios en ese intervalo. H1i = [Cmin – I·α, Cmax + I· α] I =Cmax-Cmin. $\alpha \in [0,1]$
  Cmin= mínimo fenotipo de los genes de los padres, Cmax=máximo…

El método de mutación se implementa en la clase **Mutacion**. Solo contiene un método, la mutación básica, que en caso de los individuos binarios recorre todos los alelos y de forma aleatoria con la probabilidad de mutación aplica negación lógica. En el caso de números reales cambia el número entero con un valor aleatorio.

Se aplica un **algoritmo de divide** y vencerás O(log2N) para reducir el tiempo de ejecución a la hora de elegir los valores como en ruleta y estocástica universal.

El **Elitismo** consiste en asegurar la supervivencia de un grupo con los mejores individuos de la población. En el controlador se puede especificar un porcentaje entero para el conjunto elitista que sobrevive en cada generación.
Este conjunto se calcula en la etapa de evaluación, y se comparan los individuos con su fitness, almacenando los 'r' mejores en una cola de prioridad de mínimos. Así reducimos la complejidad a O(Nlog2R) en el caso peor, siendo N el tamaño de población y R el conjunto de élite. Se compara el mínimo valor de los mejores con cada individuo, si el menor de la cola es peor que el individuo actual, se elimina de la cola y se introduce el nuevo, subiendo hasta su posición en la cola.

2. Guía de uso

Antes de ejecutar el proyecto hay que comprobar si la librería externa, **JMathPlot** está incluida. Para ello hay que entrar en la configuración del proyecto con los siguientes pasos en eclipse:

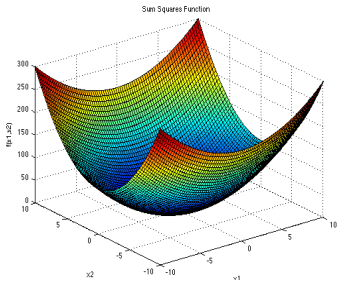
Acceder a "Properties" del proyecto →"Java Build Path" → "Libraries" → "Add External JARs". E incluir el jmathplot.jar de la carpeta **lib** del proyecto.

El proyecto se ejecuta en la clase **Main**. Aparece la interfaz y se rellenan los datos.

- **Prob. Cruce** y **Prob. Mutación:** son "double", con un intervalo de [0, 1]
- **Precisión** es un "double" y se tiene que introducir de la forma 0.1, 0.01, 0.001…
- **Elitismo** es un porcentaje, por lo que es un intervalo de [0, 100].

2. Funciones

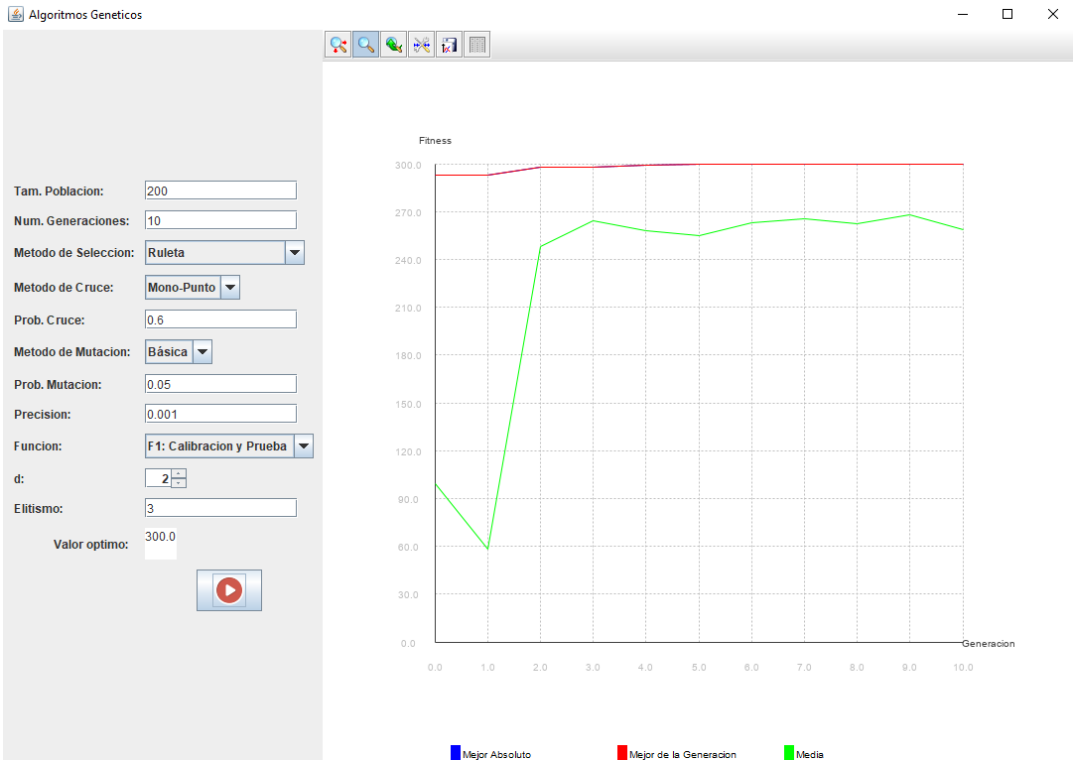
2.1 F1: Calibración y prueba



$f(x_1, x_2) = x_1^2 + 2x_2^2$

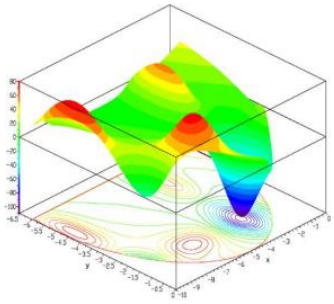
que presenta **máximo** de **300** en $x_1 = 10$ y $x_2 = 10$

x_1 y $x_2 \in [-10,10]$



2.2 F2: Mishra Bird

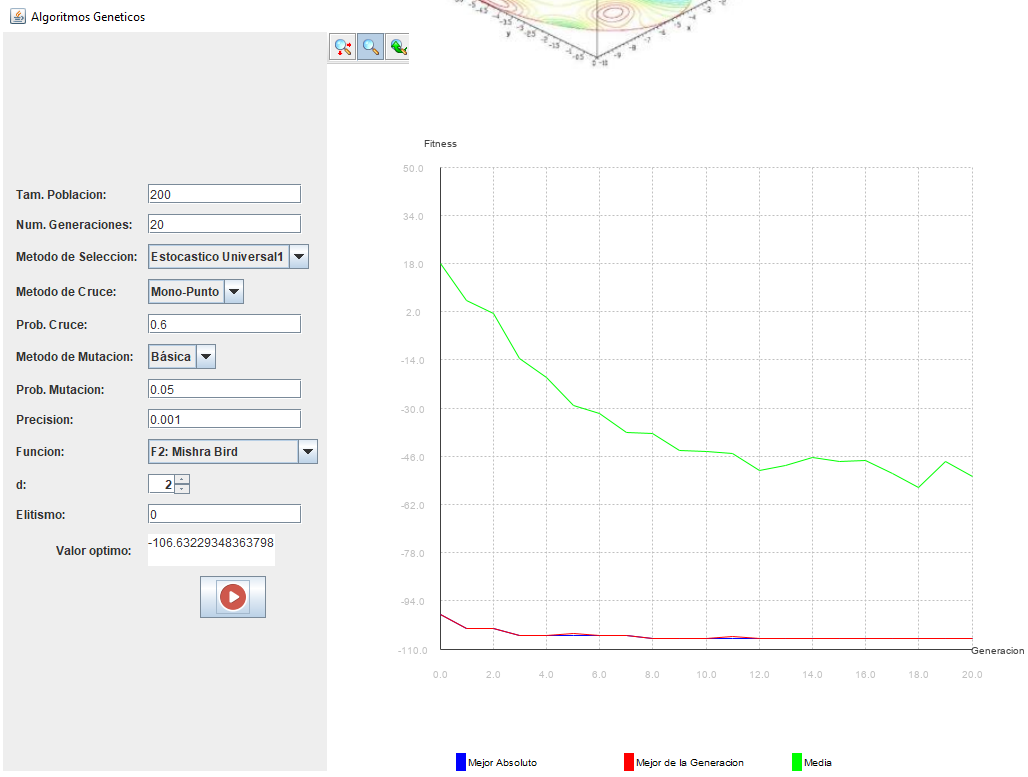
$$f(x_1,x_2) = \sin(x_2)\exp(1-\cos(x_1))^2 + \cos(x_1)\exp(1-\sin(x_2))^2 + (x_1-x_2)^2$$

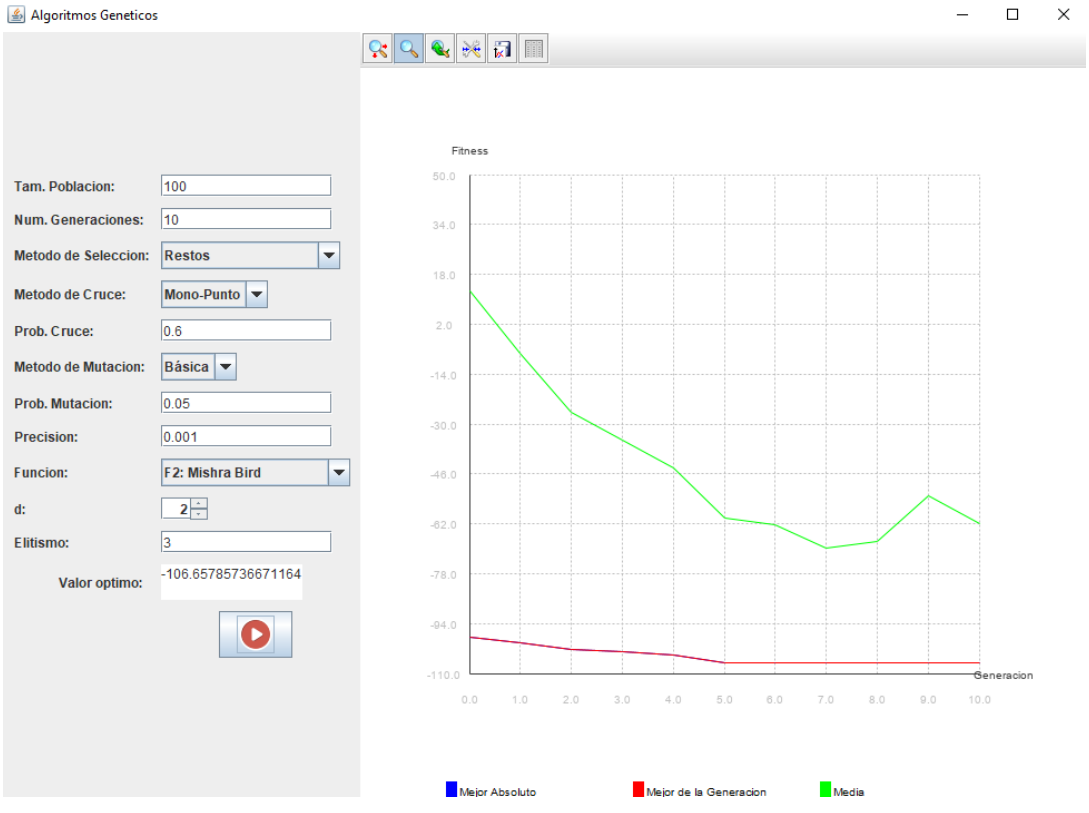


$x_1 \in [-10, 0]$

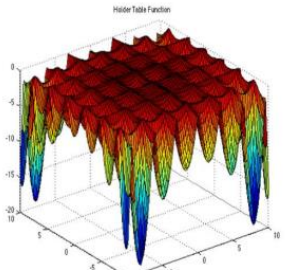
$x_2 \in [-6.5, 0]$

que presenta un mínimo global de **−106.7645367** en $x^* = -3.1302468, -1.5821422$





2.3 F3: Holder table

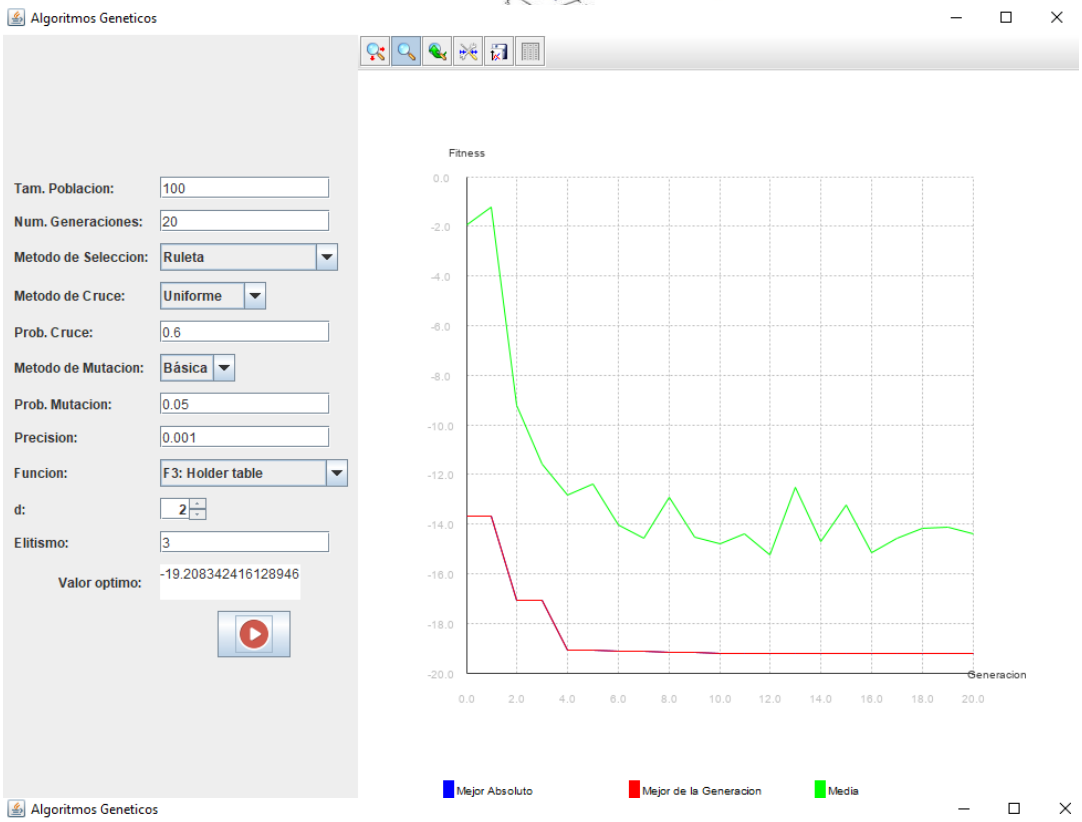


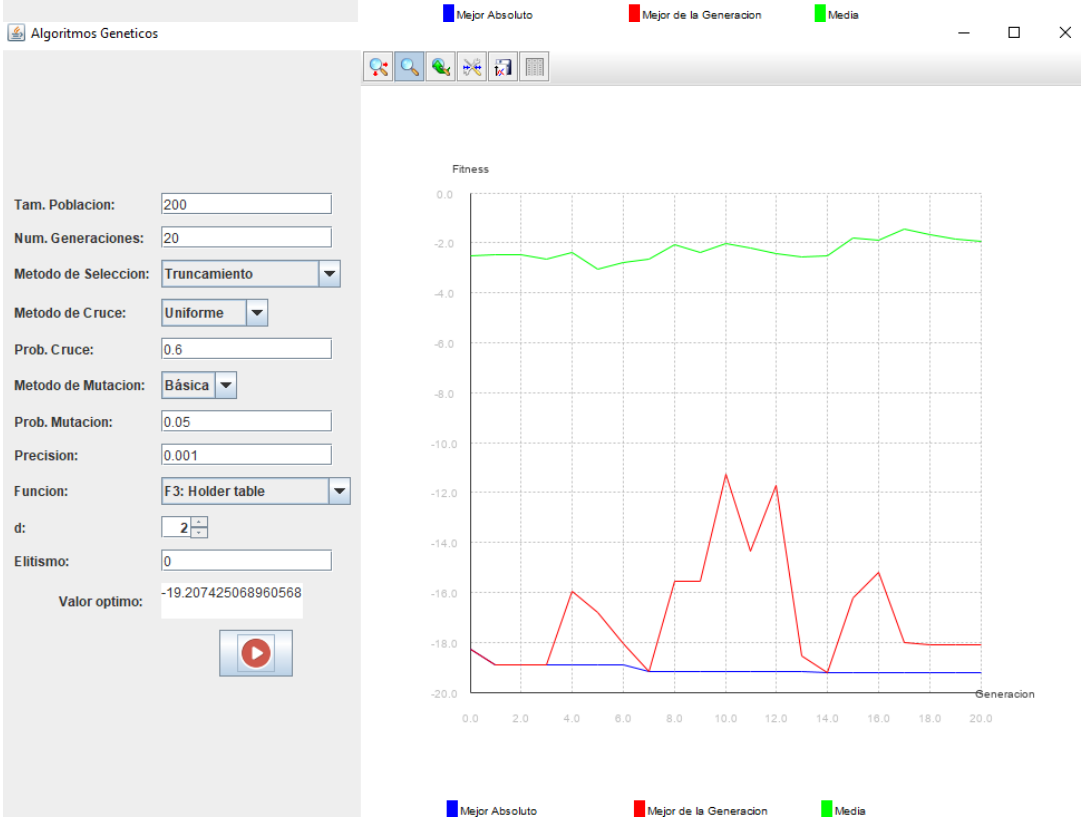
$$f(\mathbf{x}) = -\left| \sin(x_1)\cos(x_2)\exp\left(\left|1 - \frac{\sqrt{x_1^2 + x_2^2}}{\pi}\right|\right)\right|$$

$x_1\ x_2 \in [-10, 10]$

Tiene cuatro mínimos globales de **-19.2085** en

$x^* =$ **(8.05502,9.66459) (8.05502, -9.66459) (-8.05502,9.66459) (-8.05502, -9.66459)**





2.4 F4: Michalewicz con codificación binaria

$$f(\mathbf{x}) = -\sum_{i=1}^{d} \sin(x_i)\sin^{2m}\left(\frac{ix_i^2}{\pi}\right)$$

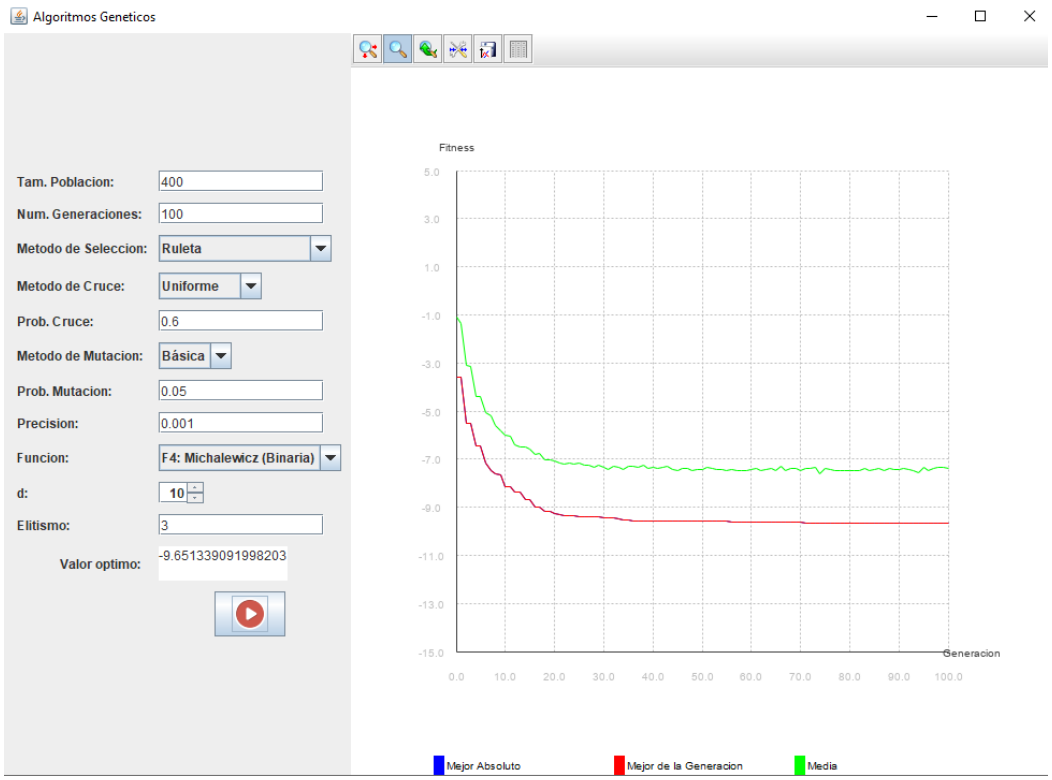
$x_i \in [0, \pi] \quad m = 10$

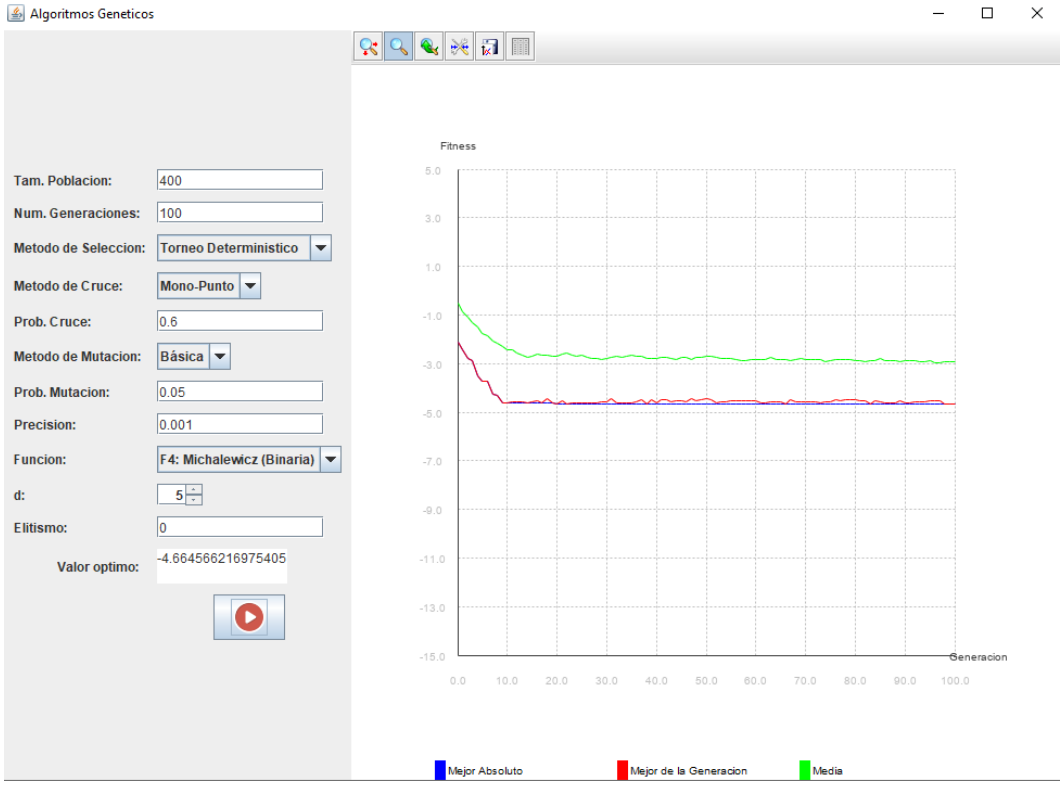
que presenta los siguientes mínimos en función de d:

d= 2 f(x*) = - 1.8013 en x* = (2.20, 1.57)

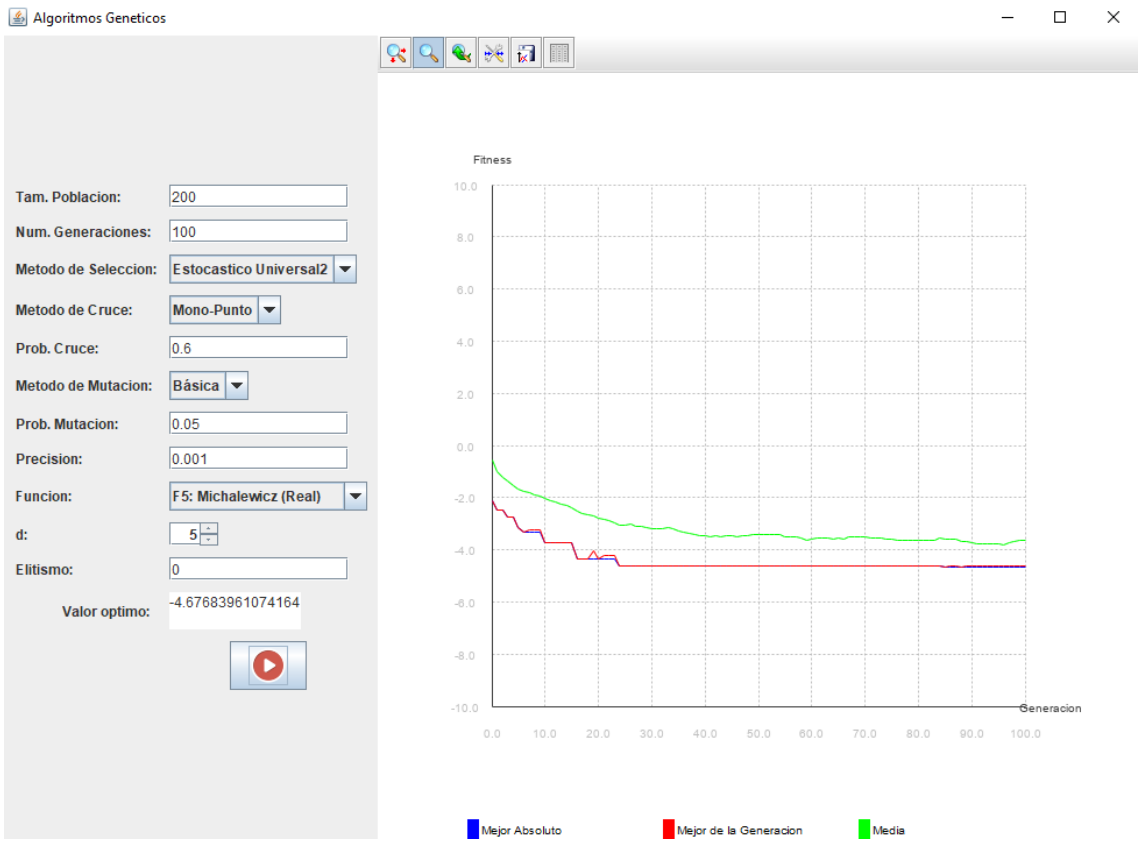
d = 5 f(x*) = - 4.6876

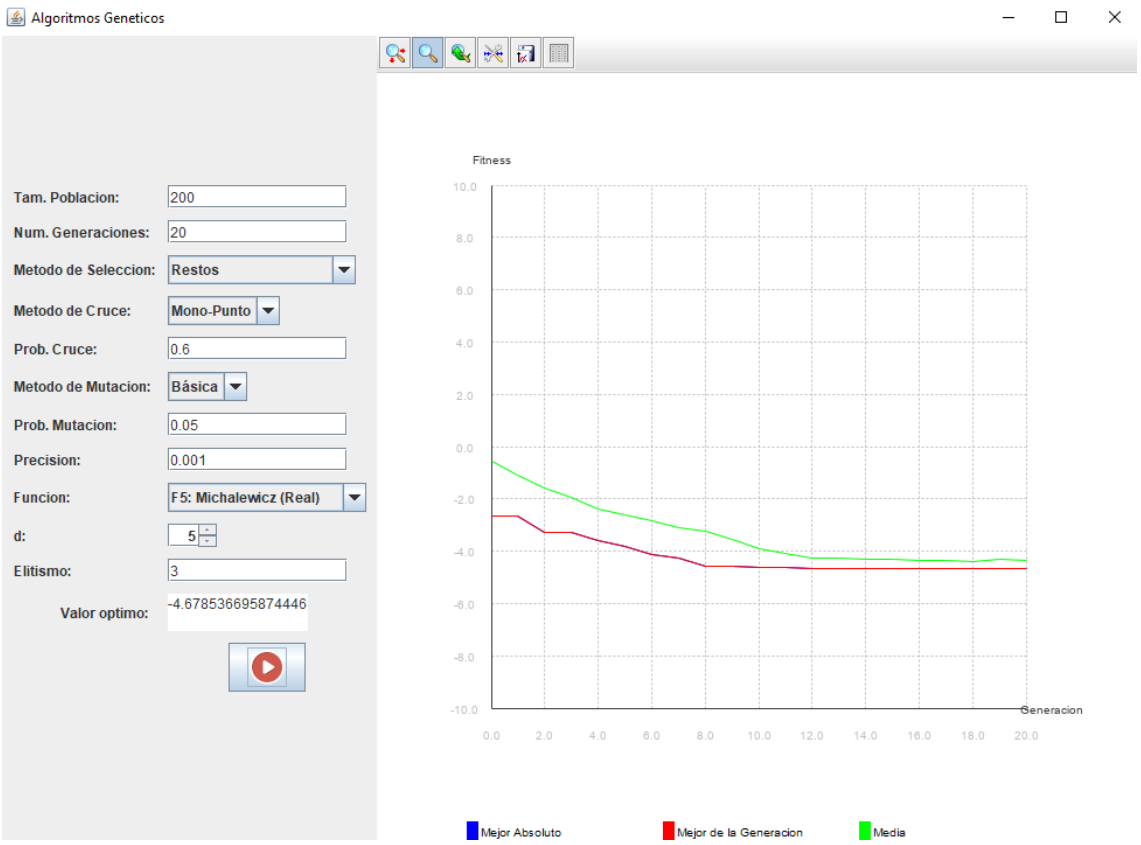
d = 10 f(x*) = - 9.6601





2.5 F5: Michalewicz con codificación binaria





3. Observaciones

4. Reparto de Tareas

Daniel ha hecho toda la GUI. La lógica del programa la hemos juntos.