PRÁCTICA 1

Programación Evolutiva

Grupo 10

Gómez Benavente, Luis José
Sánchez Perez, Guillermo

1. Ejecuciones

Para todas las ejecuciones, usaremos los mismos valores en los parámetros:

- Probabilidad de cruce = 0.6
- Probabilidad de mutación = 0.05
- Precisión = 0.001

El resto de operadores y valores escogidos dependen de la función, donde hemos escogido los que mejor encuentran una solución óptima en dicha función. Por otra parte, también describiremos ejecuciones con y sin elitismo de manera que se pueda apreciar visualmente la diferencia entre ambos.

1.1. Función 1

Para la función 1:

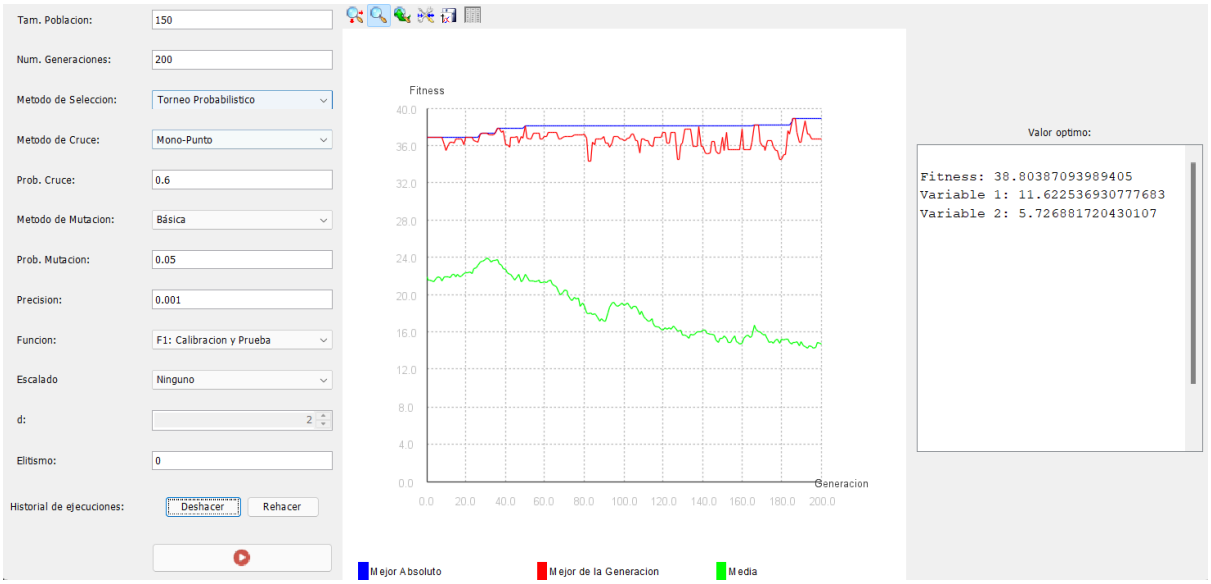
$f(x_1, x_2) = 21.5 + x_1 * sin(4 * x_1) + x_2 * sin(20 * x_2)$,

donde:

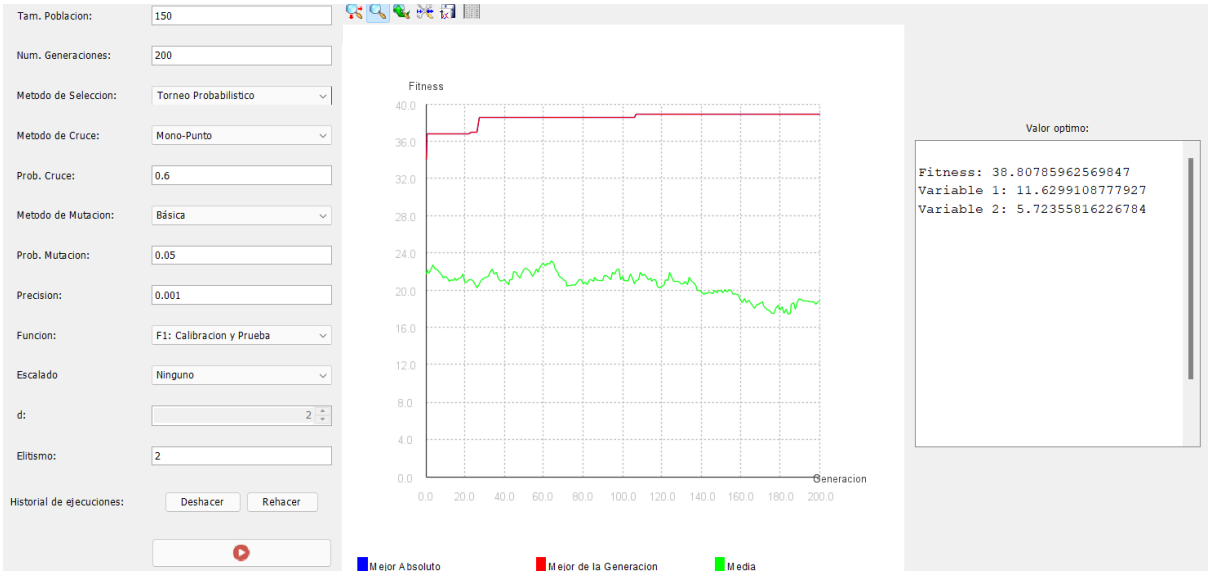
$x_1 \in [-3, 12.1]$, $x_2 \in [4.1, 5.8]$, seleccionando los siguientes parámetros, hemos encontrado una aproximación al maximo:

- **Tamaño de población: 150**
  Balance entre diversidad genética y costo computacional.
- **Número de generaciones: 200**
  Suficiente tiempo para converger sin excesivo costo computacional.
- **Método de selección: Torneo Probabilístico**
  Favorece soluciones buenas sin perder diversidad.
- **Método de cruce: Monopunto**
  Simple y efectivo para combinar información genética.

Sin elitismo:



Con elitismo:



1.2. Función 2

Respecto a la función 2, con valor:

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

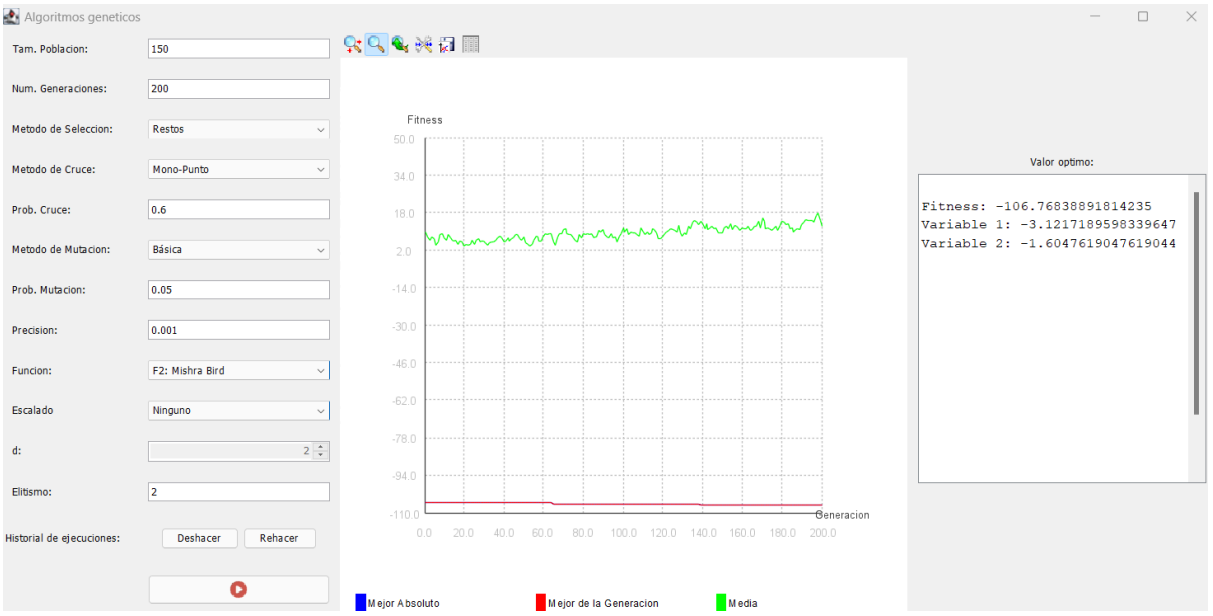
donde:

$$x_1 \in [-10, 0] \quad \text{y} \quad x_2 \in [-6.5, 0]$$

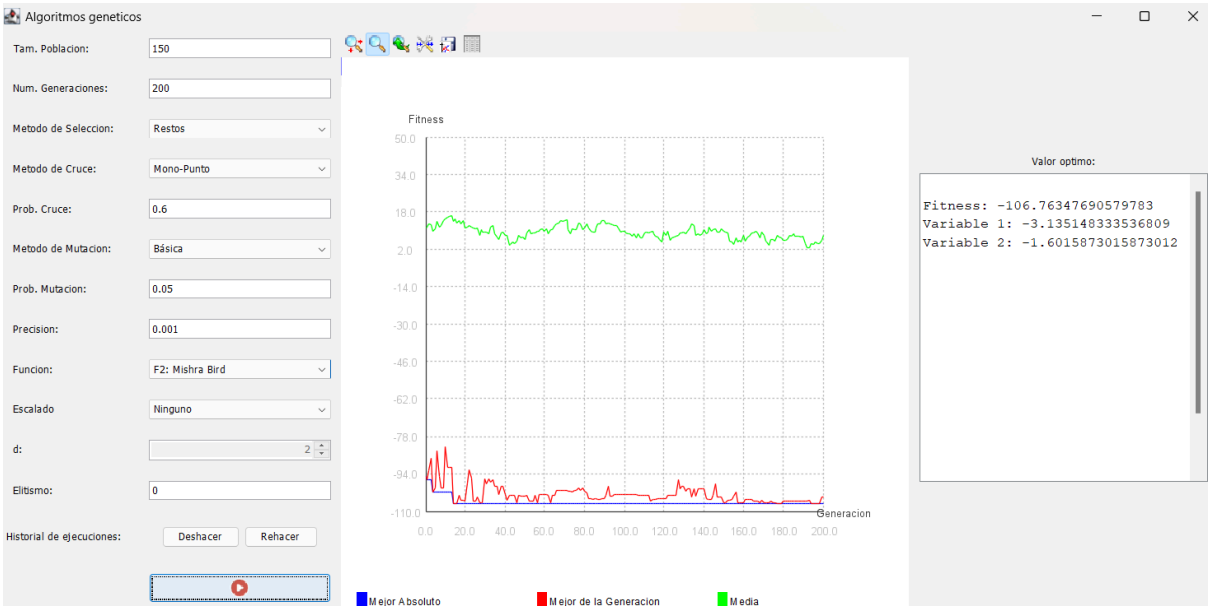
Utilizamos los valores descritos en la imagen para encontrar el mínimo absoluto de la función, a continuación explicamos el por qué de esta elección:

- **Tamaño de población: 150**
  Un tamaño de población suficientemente grande que nos permite una mayor diversidad genética y reduce la posibilidad de estancamiento en óptimos locales.
- **Número de generaciones: 200**
  Permite una exploración profunda del espacio de soluciones, asegurando que el algoritmo tenga suficiente tiempo para converger al óptimo global.
- **Método de selección: Restos**
  La selección por restos reduce la presión selectiva excesiva y su combinación de selección determinista y probabilística evita que los individuos más aptos dominen prematuramente, reduciendo el riesgo de convergencia a un óptimo local demasiado rápido.
- **Método de cruce: Mono-Punto con Probabilidad 0.6**
  Un cruce de un solo punto con una probabilidad del 60% permite una combinación eficiente de genes sin reducir demasiado la exploración.
- **Elitismo: 2**
  Garantiza que el mejor 2% de la población se mantenga, evitando la pérdida de soluciones prometedoras.

Con elitismo:



Sin elitismo:



1.3. Función 3

La función 3, Schubert:

$$f(x_1, x_2) = (\sum_{i=1}^{5} i * cos((i+1)x_1 + i))(\sum_{i=1}^{5} i * cos((i+1)x_2 + i))$$

donde:

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

- **Tamaño de población: 150**
  Se mantiene un equilibrio entre diversidad genética y costo computacional.
- **Número de generaciones: 200**
  Permite suficiente evolución sin un alto costo computacional.
- **Método de selección: Torneo Determinístico**
  Selecciona siempre el mejor de un subconjunto, promoviendo explotación y reduciendo aleatoriedad.
- **Método de cruce: Mono-Punto**
  Facilita la combinación de información genética sin fragmentar demasiado los cromosomas.

1.4. Función 4

La función 4, Michalewicz:

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

donde:

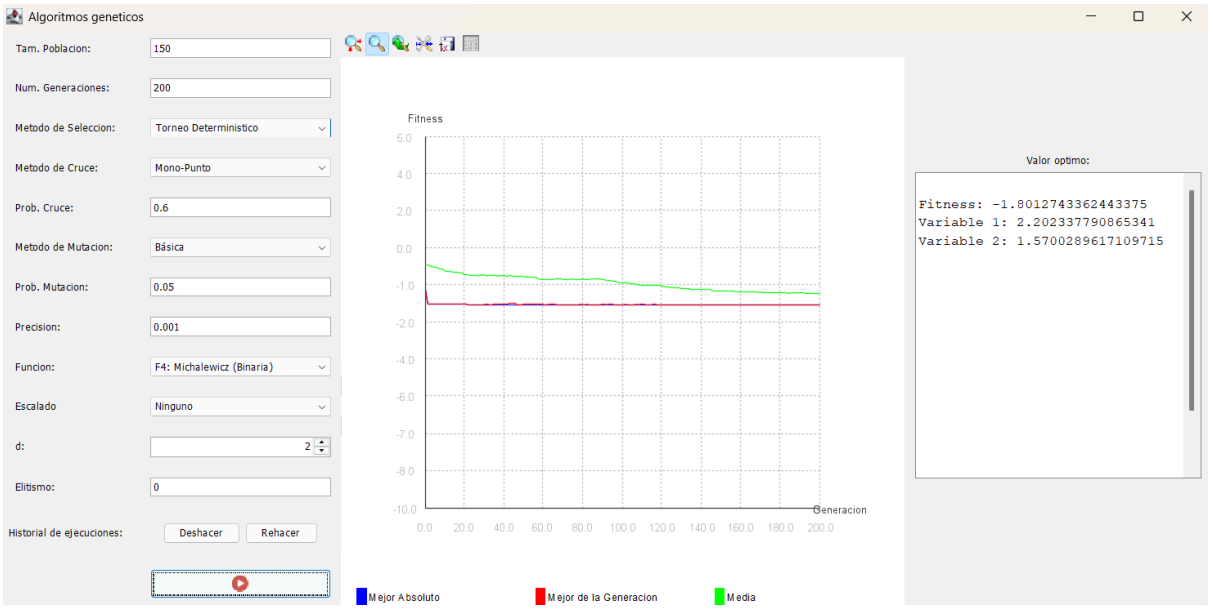
$$x_i \in [0, \pi],\ m = 10$$

y d es un parámetro de la función, que indica el número de variables de entrada.

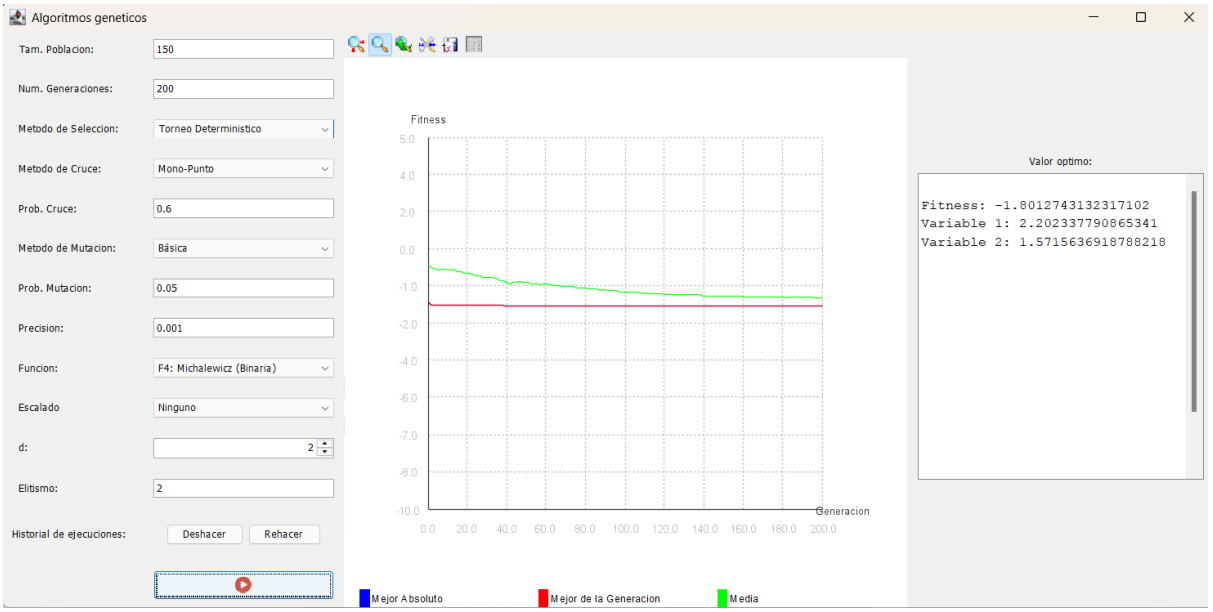
- **Tamaño de población: 150**
  Se mantiene un equilibrio entre exploración del espacio de búsqueda y eficiencia computacional.
- **Número de generaciones: 200**
  Permite que el algoritmo evolucione lo suficiente para encontrar soluciones óptimas sin sobrecostos computacionales.
- **Método de selección: Ranking**
  Reduce la presión selectiva excesiva y promueve la diversidad, evitando que los mejores individuos dominen demasiado rápido.
- **Método de cruce: Mono-Punto**
  Simplicidad y eficiencia en la recombinación genética, permitiendo el intercambio de segmentos de información sin fragmentaciones excesivas.

d = 2

Sin elitismo:

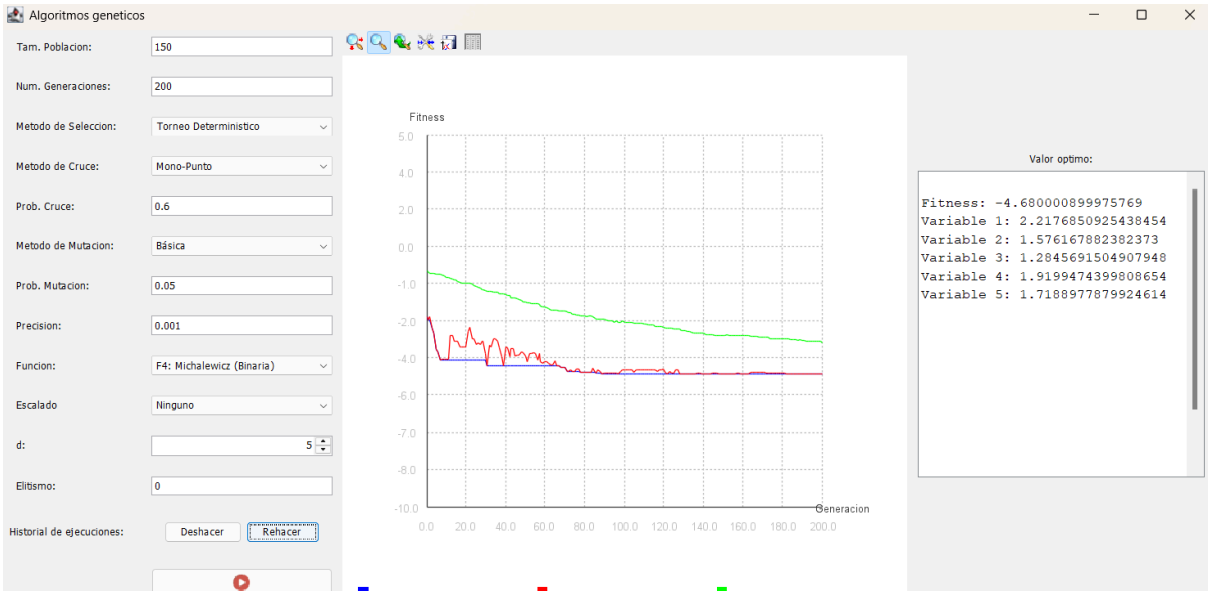


Con elitismo:

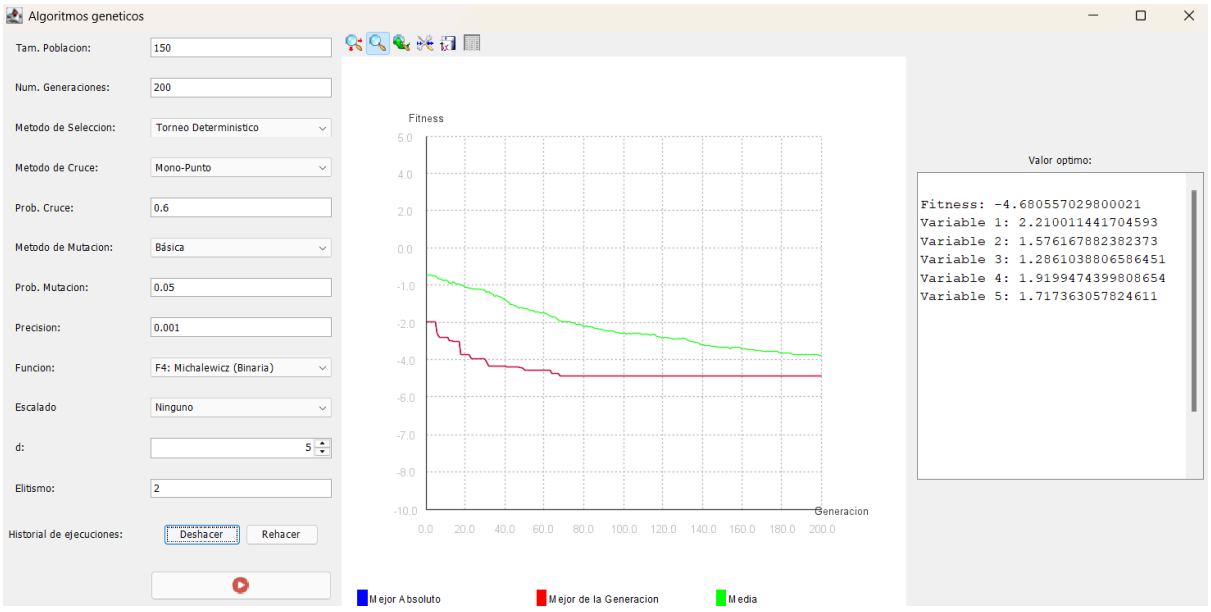


d = 5

Sin elitismo:



Con elitismo:



1.5. Función 5

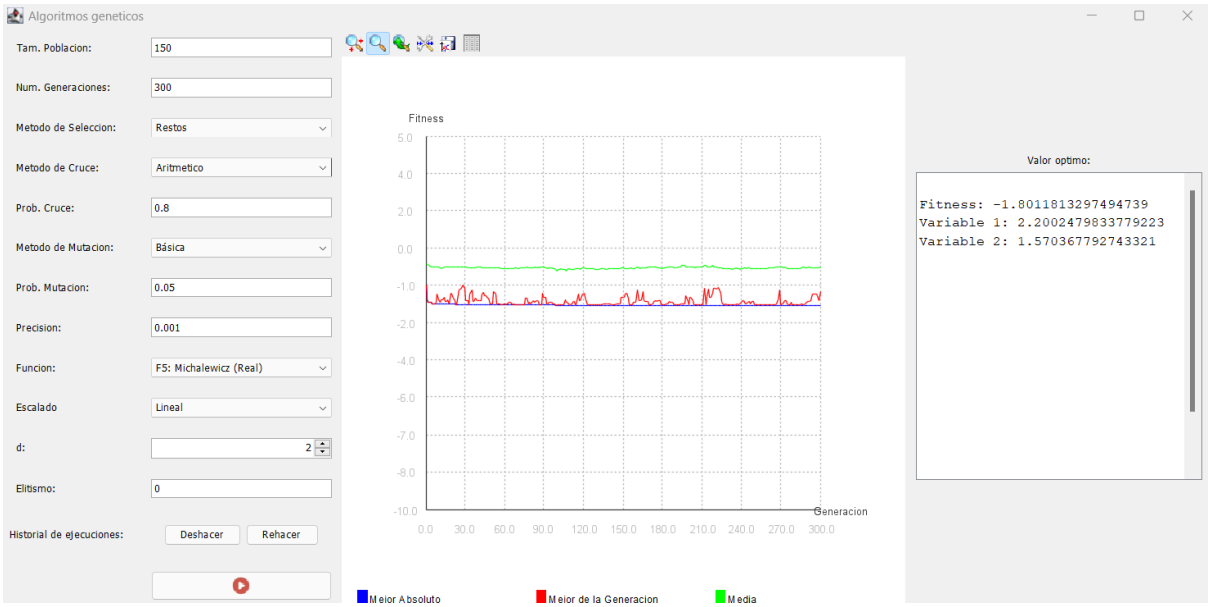
La función 5 es la misma que la 4, pero con una representación con números reales o Double, en vez de binario. Esta representación da pie a nuevos operadores genéticos como el cruce aritmético o BLX-α.

Debido a la naturaleza de la función hemos optado por usar distintos parámetros para los distintos números de genes dentro del cromosoma, adaptando así la búsqueda del valor óptimo.

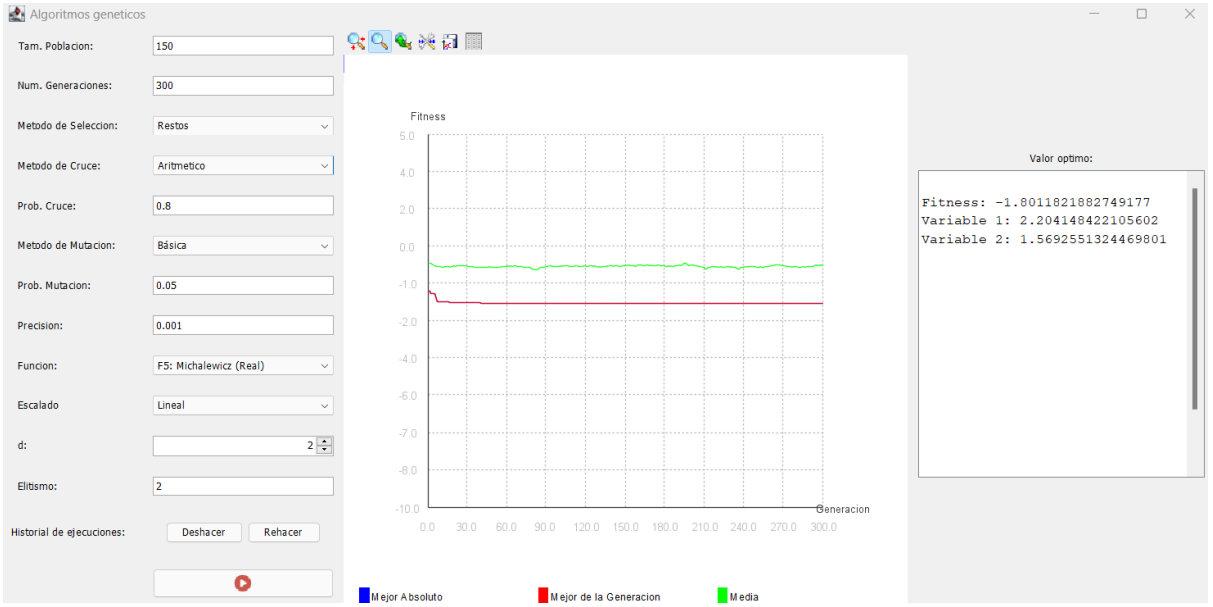
d=2

- **Tamaño de población: 150**
  Se mantiene un equilibrio entre exploración del espacio de búsqueda y eficiencia computacional.
- **Número de generaciones: 300**
  Permite que el algoritmo evolucione lo suficiente para encontrar soluciones óptimas sin sobrecostos computacionales.
- **Método de selección: Restos**
  La selección por restos reduce la presión selectiva excesiva y su combinación de selección determinista y probabilística evita que los individuos más aptos dominen prematuramente, reduciendo el riesgo de convergencia a un óptimo local demasiado rápido.
- **Método de cruce: Aritmético**
  Dado que la función tiene múltiples óptimos locales, este método balancea bien la exploración y explotación, lo que ayuda a encontrar soluciones de alta calidad sin quedar atrapado en un solo valle del fitness.

Sin elitismo:



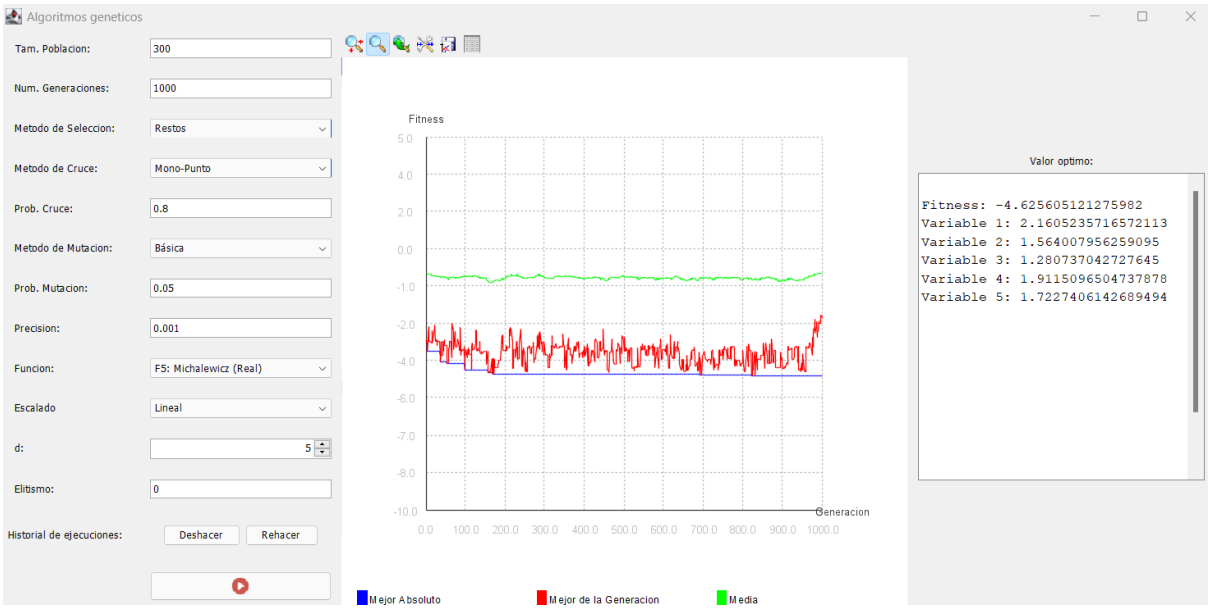
Con elitismo:



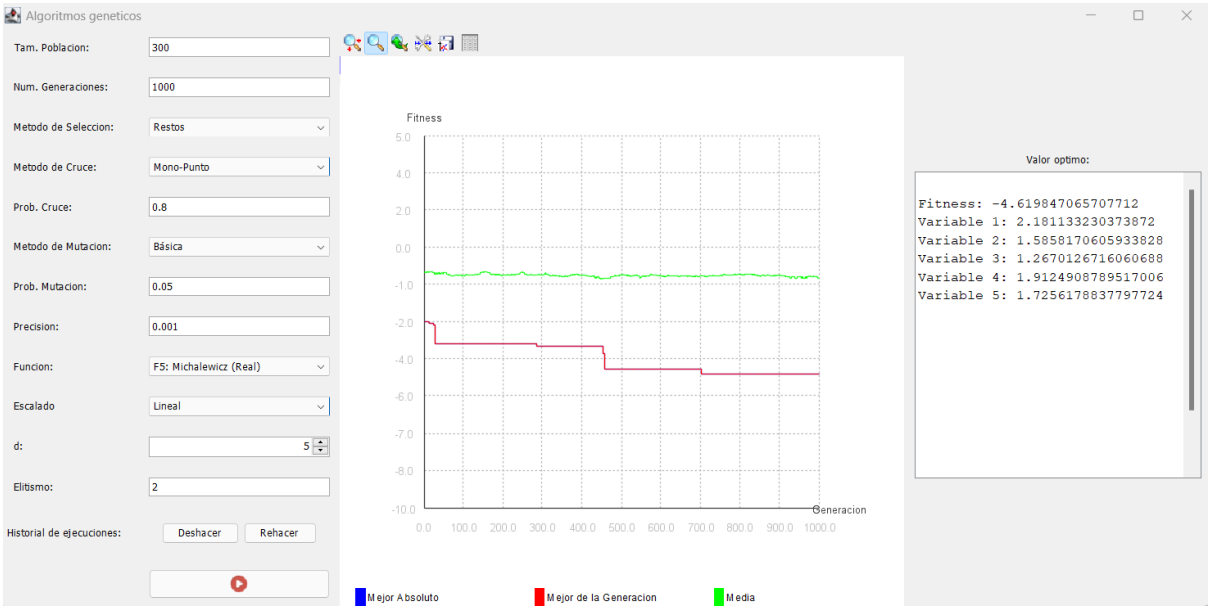
d=5

- **Tamaño de población: 300**
  Para valores mayores de d, dado que es una función muy rugosa optamos por usar una población más grande para explorar un mayor rango de soluciones.
- **Número de generaciones: 1000**
  Permite que el algoritmo evolucione lo suficiente para converger al máximo para d=5.
- **Método de selección: Restos**
  La selección por restos reduce la presión selectiva excesiva y su combinación de selección determinista y probabilística evita que los individuos más aptos dominen prematuramente, reduciendo el riesgo de convergencia a un óptimo local demasiado rápido.
- **Método de cruce: Monopunto**
  Optamos por un cruce monopunto para un mayor valor ,dado que la dimensionalidad es intermedia, monopunto logra un buen balance entre diversidad y estabilidad, favoreciendo la evolución progresiva del algoritmo genético.

Sin elitismo:



Con elitismo:



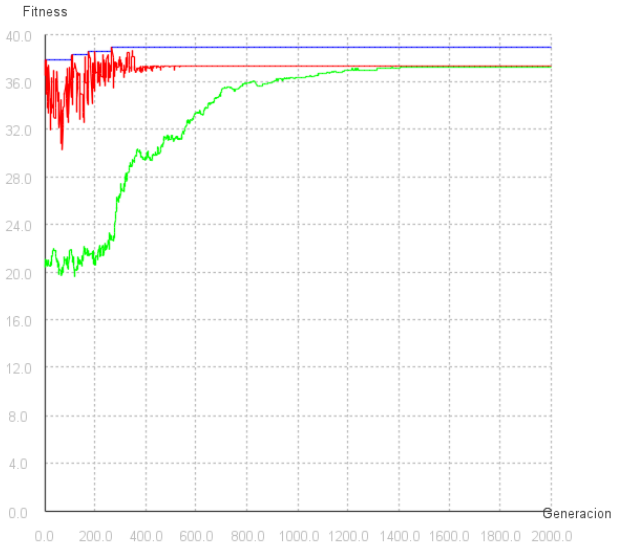
2. Conclusiones

Al usar operadores diferentes para cada función, hemos visto que hay algunos casos donde cierto método de selección o cruce se comporta mejor que otros, por lo que el usar siempre los parámetros no es lo mejor. No hay una “bala de plata” para escoger la configuración, depende del problema sobre el que quieras buscar, por lo que la variedad de configuraciones es algo importante.

En cuanto al uso de elitismo en las búsquedas, podemos observar como el uso de este es capaz de acelerar el proceso de búsqueda, encontrando el máximo unas generaciones antes que si no lo usasemos.

También hemos observado que hay algunos métodos de selección que, por lo general, funcionan mejor que otros. Los que mejores resultados nos han ofrecido a la hora de probar las funciones han sido los torneos, restos y ranking, aunque hay ciertos casos donde estos métodos resultaban ser peores que otros.

A la hora de ampliar la búsqueda con muchas generaciones e individuos, hemos notado que a partir de cierto punto, el valor no mejora y la media y el mejor de la población convergen. Esto hace que el tiempo de ejecución aumente considerablemente, sin obtener mejores resultados. En esta gráfica, perteneciente a la función 1, con 150 individuos y 2000 generaciones, usando cruce monopunto y selección por ranking, vemos cómo a partir de la generación 300 mas o menos, se encuentra el máximo y a partir de ahí deja de mejorar, por lo que hemos gastado el tiempo de computar más de 1600 generaciones sin encontrar nada mejor.



Finalmente, en el caso de las funciones 4 y 5, podemos ver como la dimensión influye mucho a la hora de la búsqueda. Para valores altos se necesitan muchas más generaciones para encontrar un buen valor, aumentando el tiempo que tarda en hacer la búsqueda. Como curiosidad, al hacer pruebas hemos notado que para valores de dimensión altos, el combinar el método de selección restos con elitismo se comporta peor que sin tener élite.

3. Detalles de implementación

La implementación de nuestro algoritmo genético, aparte de cubrir los requisitos de implementación del enunciado, incluye funcionalidades extra que encontramos útiles para el contexto de los problemas a resolver, entre estas las más destacables son el historial de ejecuciones, que ofrece al usuario la posibilidad de volver a ejecuciones anteriores y posteriores pudiendo así comparar fácilmente entre ejecuciones, y los diferentes tipos de escalados, que ofrecen distintos tipos de escalados para adaptar al problema.

En cuanto a la arquitectura de nuestra aplicación, algunos puntos destacables son:

- Hemos creado individuos abstractos generalizando y facilitando la posterior implementación de individuos específicos. Decidimos partir de un tipo Individuo abstracto, genérico sobre el tipo de los genes que va a guardar. De esta clase, sacamos otras dos abstractas, *IndividuoBoolean* e *IndividuoDouble*, los cuales implementan todos los métodos de su clase padre ya especificados para su tipo específico, a falta de la función de fitness. Por último, para cada función creamos un individuo, que hereda de alguno de los dos anteriores, el cual ya implementa la función de fitness, la cual es la propia función a maximizar/minimizar.
- Para la creación de individuos, selecciones, escalados y cruces hemos usado factorías, permitiendo desacoplar el algoritmo genético de los operadores y funciones que queramos usar, permitiendo con un solo algoritmo usar la combinación que queramos de operadores y añadir nuevos métodos sin necesidad de modificar el algoritmo.
- A la hora de seleccionar, se hace un desplazamiento de fitness para: eliminar valores negativos, a la hora de maximizar; y convertir una función de minimización a maximización. Al hacer esto, las selecciones no tienen que diferenciar casos de minimización o maximización, simplemente se centran en maximizar lo que les llega y devolver los índices seleccionados para reconstruir la selección posteriormente el el algoritmo genético, quitándole así responsabilidades a las selecciones.
- La mutación está implementada en cada individuo de un tipo (*Invididuo Boolean* e *Individuo Double)*, los cuales saben cómo son sus genes y saben cómo modificarlos correctamente en vez de usar otra clase que los modifique directamente.

4. Reparto de tareas

Para asegurarnos de que ambos miembros del grupo comprendíamos a fondo todos los conceptos de la asignatura, decidimos repartirnos las tareas equitativamente. En primer lugar, elaboramos un esquema general del algoritmo genético, definiendo sus principales componentes y la jerarquía de clases para la representación de los individuos.

En cuanto a la implementación, distribuimos el trabajo de la siguiente manera: cada uno de nosotros se encargó de la mitad de los métodos de selección, cruce e individuos. Uno de nosotros implementó ciertos métodos de selección, como Torneo Determinístico, Ruleta, Ranking y Truncamiento, mientras que el otro desarrolló Torneo probabilístico, Truncamiento y Restos. De la misma forma, dividimos la implementación de los métodos de cruce: mientras uno trabajaba en Monopunto y BLX-α , el otro se encargaba de Aritmético y Uniforme. Para los individuos, ambos trabajamos y entendimos el individuo base y

posteriormente uno de nosotros se enfocó en las representaciones binarias, mientras que el otro trabajó en la representación real.

A pesar de esta división inicial, desarrollamos el algoritmo genético en conjunto, asegurándonos de comprender cómo se integraban todos los métodos en el proceso evolutivo. De esta forma, ambos adquirimos una visión completa del flujo del algoritmo, desde la inicialización de la población hasta la convergencia de la solución. Esta metodología nos permitió afianzar los conocimientos y garantizar que, al final del proyecto, ambos domináramos todos los aspectos de la asignatura, sin que ninguna parte del algoritmo fuera desconocida para alguno de nosotros.