Universidad Carlos III

Curso de Algoritmos genéticos y evolutivos 2020-21

Práctica 1

Curso 2021-22

Desarrollo de algoritmo genético

Fecha: **13/10/2021** - PRÁCTICA: **1**

GRUPO: **83**

Alumno: **Jacobo Javier Galache Gómez-Coalla**

Introducción

Este documento trata de un análisis del rendimiento del algoritmo genético en referencia a búsquedas realizadas sobre las funciones de fitness propuestas, calculadas en las direcciones:

http://memento.evannai.inf.uc3m.es/age/test?c=<chromosome>

http://memento.evannai.inf.uc3m.es/age/alfa?c=<chromosome>

donde <chromosome> se sustituye por una cadena de bits de longitudes 64 y 384 respectivamente.

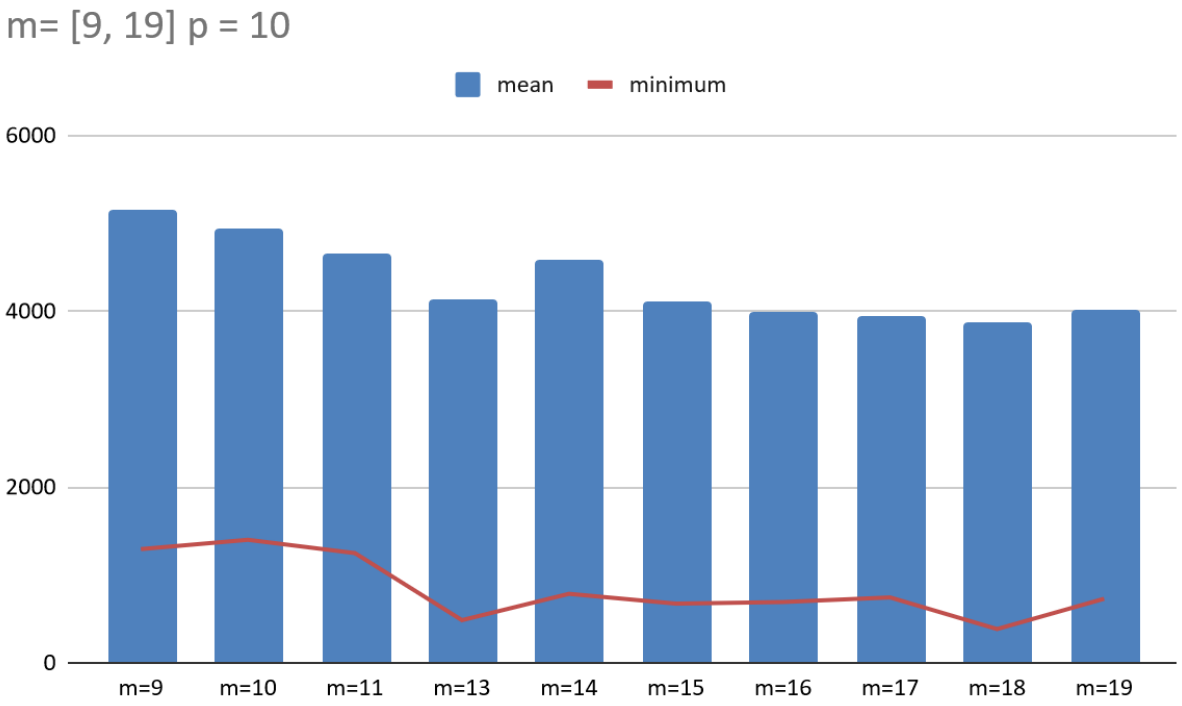
Algoritmo

El algoritmo está estructurado de forma que tenemos 3 variables que se pueden modificar con el fin de la depuración:

- **n:** Es el número de individuos de los cuales constará cada generación, por limitaciones de diseño debe ser un número par
- **m:** Es el tamaño de las selecciones para la ejecución de los torneos.
- **p (pepper):** Es el porcentaje de mutación de bits.

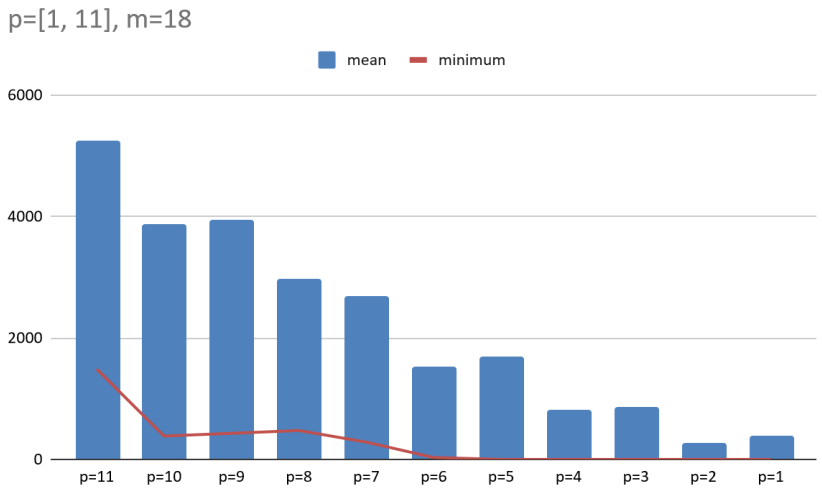
Depuración para test

Comenzando el proyecto, se utilizó experimentación con valores para determinar la dirección en la que los valores de nuestros parámetros mejorarían las soluciones que obtenemos del algoritmo. Comenzamos con valores arbitrarios de p= 10, m=10 y n=100.



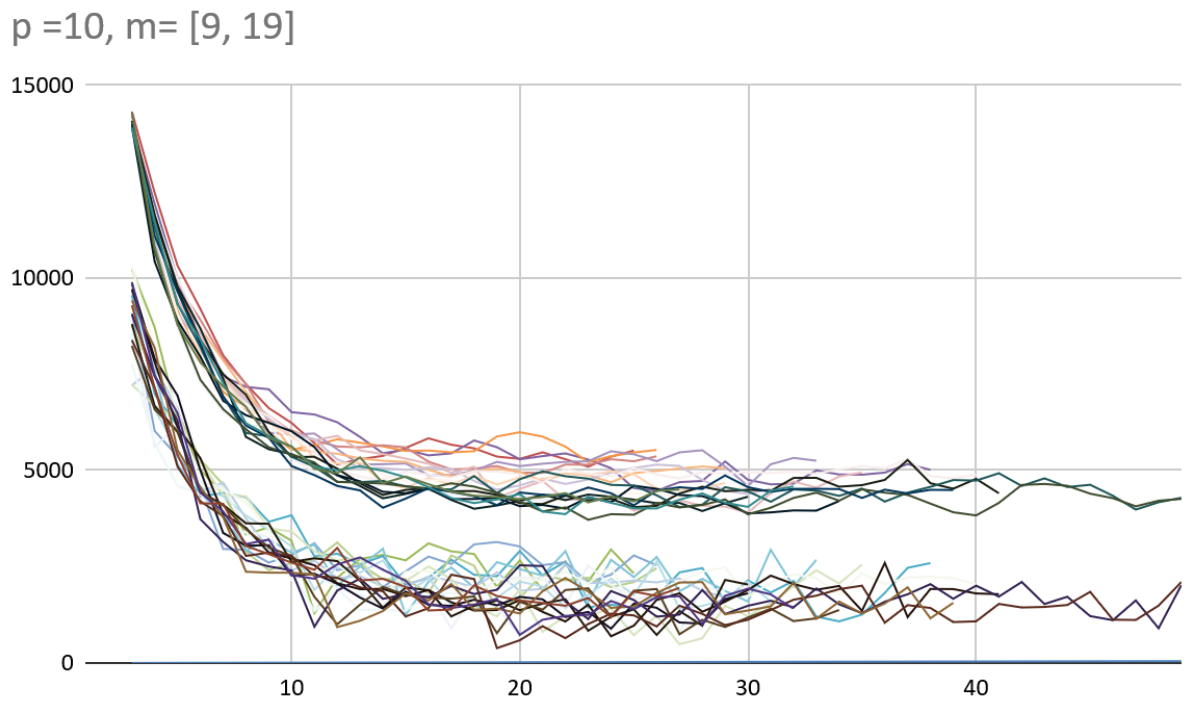
Como podemos observar en la gráfica, al aumentar el tamaño del torneo hasta llegar a 18 los valores mejoraban los resultados ligeramente. Observamos también que a partir de este valor obteníamos ligeramente peores soluciones respecto concretamente a m=18, por lo que decidimos quedarnos con este valor.

Después comenzamos a ajustar la p y aquí es donde obtenemos una mejora mucho más sustancial.

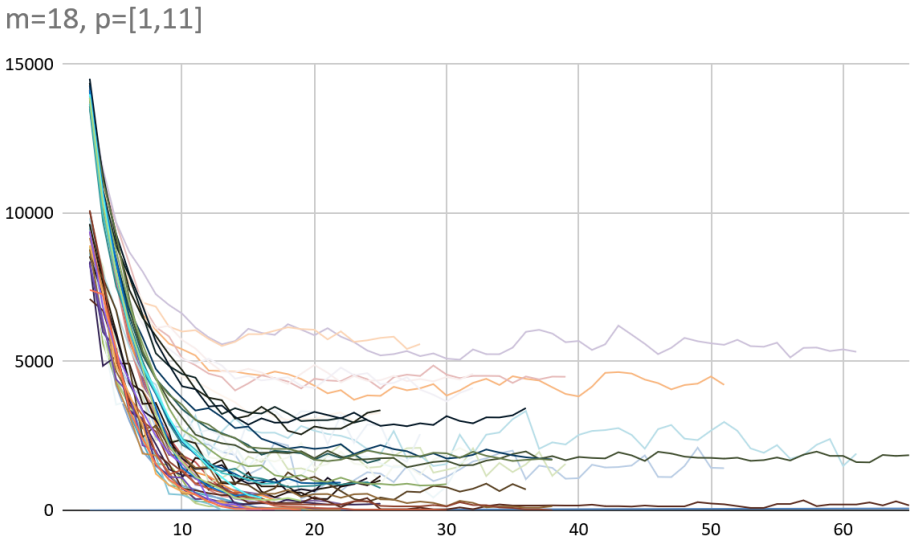


Conforme a que vamos reduciendo la p podemos ver que los resultados mejoran de una forma muy drástica. Pasando de soluciones que no llegan al valor objetivo en 30-40 generaciones a soluciones que alcanzan la solución en apenas 15.

Analizando los datos de las pruebas podemos comprobar que el algoritmo se comporta de forma esperada en referencia a otros algoritmos genéticos, teniendo una explotación inicial alta y ralentizándose pasadas unas pocas generaciones.

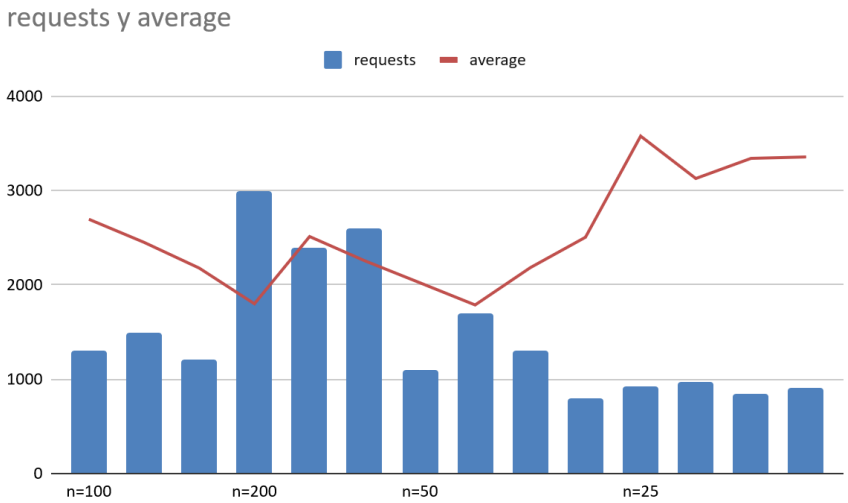


El grupo superior se corresponde con los valores medios de las generaciones mientras que el grupo inferior se corresponde con los mínimos de dichas generaciones



Aunque menos claro que en el gráfico anterior también se pueden apreciar los grupos de mínimos y medias viendo los valores iniciales

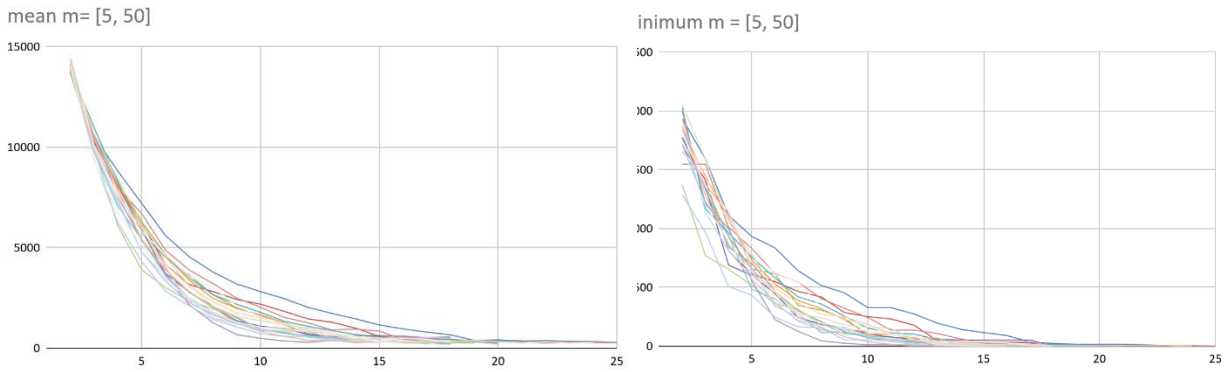
También hemos observado que tener un número grande de individuos no es estrictamente necesario si queremos optimizar el número de peticiones que se hacen al servidor para obtener la solución.

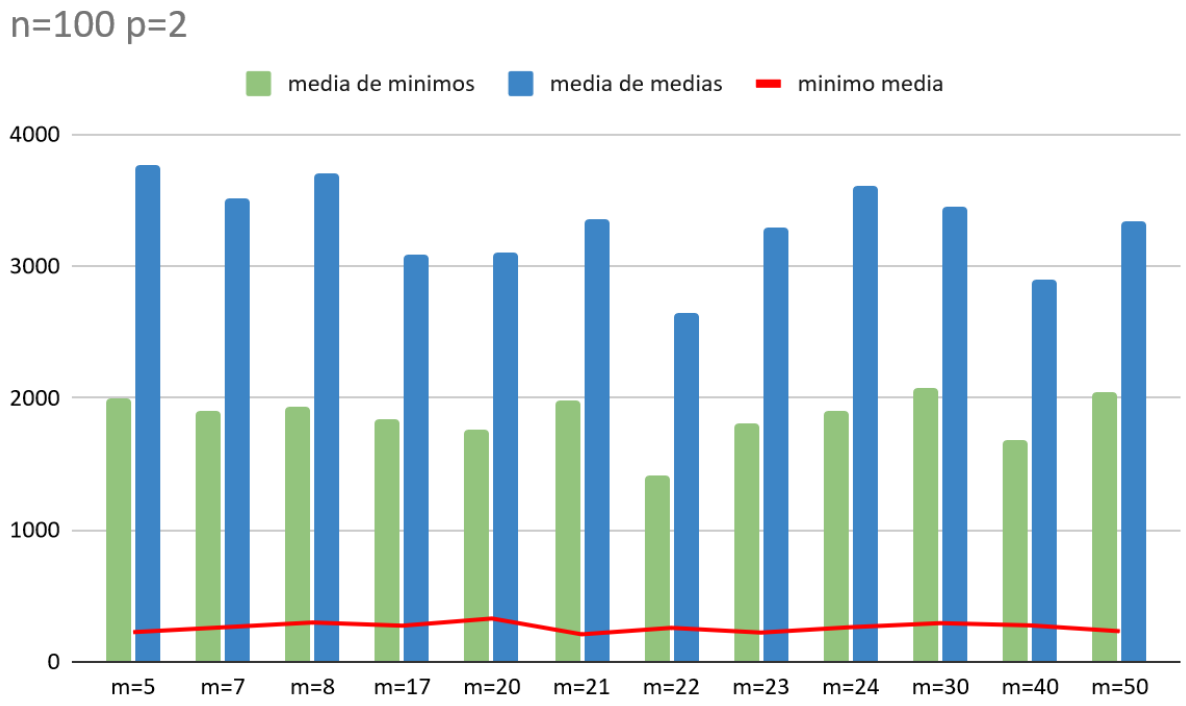


En la gráfica anterior podemos observar el número de peticiones necesarias para llegar a la solución dado un cierto tamaño de generación. Contrastado con la media de fitness de las generaciones. Podemos observar que el número de peticiones es más bajo cuando reducimos el tamaño de población. Esto no sería un resultado esperable en un caso general donde se apliquen algoritmos genéticos dada la

necesidad de tener poblaciones altas para resolver situaciones donde se estanquen poblaciones de menor tamaño. Podemos suponer que el efecto ocurre debido a la baja complejidad del problema.

Dado el análisis limitado que hicimos del parámetro m, decidimos realizar más pruebas para comprobar como se comporta el algoritmo al tener distintos valores de m.





En las gráficas anteriores se condensa la información obtenida de las pruebas. En las dos primeras gráficas podemos observar que la evolución del algoritmo se modifica a la par con el parámetro m, haciéndose la explotación más agresiva al aumentar el valor.

También pudimos observar que los valores se volvían más dispares con forme a que aumentábamos los valores del torneo. En el histograma podemos observar como los valores centrados en m=22 son los más bajos del conjunto, y que los valores medios de los mínimos comienzan a oscilar cuando aumentamos m. Esto nos indica que para este caso concreto, el valor óptimo del tamaño del torneo estaría alrededor de 20. Estos valores también no serían muy esperables, dado que en situaciones normales, donde la complejidad del problema es mayor, sería más favorable tener una mayor diversidad de individuos, lo que nos llevaría a reducir el tamaño de torneos, con el fin de reducir la presión evolutiva y permitir que haya un mayor rango de fenotipos dentro de la población.

Algoritmo pseudo-evolutivo

Anterior al desarrollo del algoritmo completo evolutivo, pudimos desarrollar un algoritmo basado principalmente en mutación y exploración en profundidad que conseguía de forma consistente alcanzar la solución en una cantidad sustancialmente baja de peticiones (1200).

Este algoritmo a veces se trababa en ciertos cromosomas. Esto probablemente se debe a su enfoque de tomar únicamente el mejor individuo de cada generación y únicamente realizar mutaciones sobre dicho individuo para crear la siguiente generación, esto causa que se atasque en ciertos casos y tarde considerablemente más en llegar a la solución.

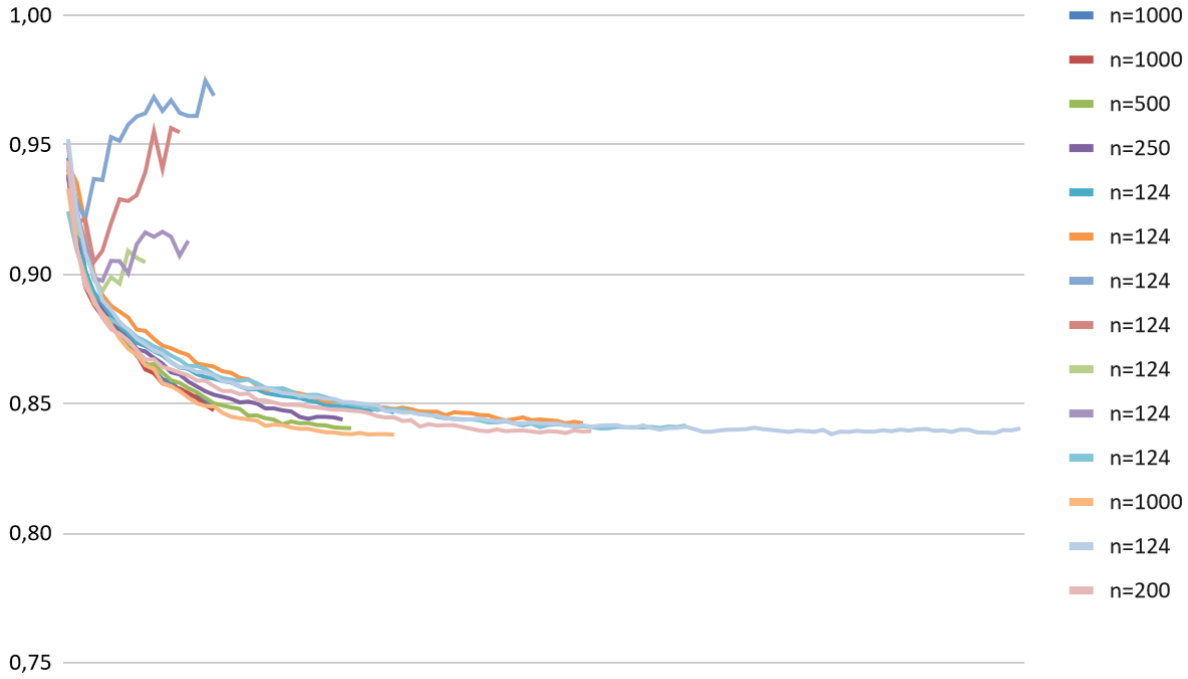
Realizamos pruebas de dicho algoritmo con la función de fitness Alpha pero no tuvimos unos resultados acordes a las expectativas dadas por su rendimiento con la función de test. Estancándose en valores mayores que 0.9.

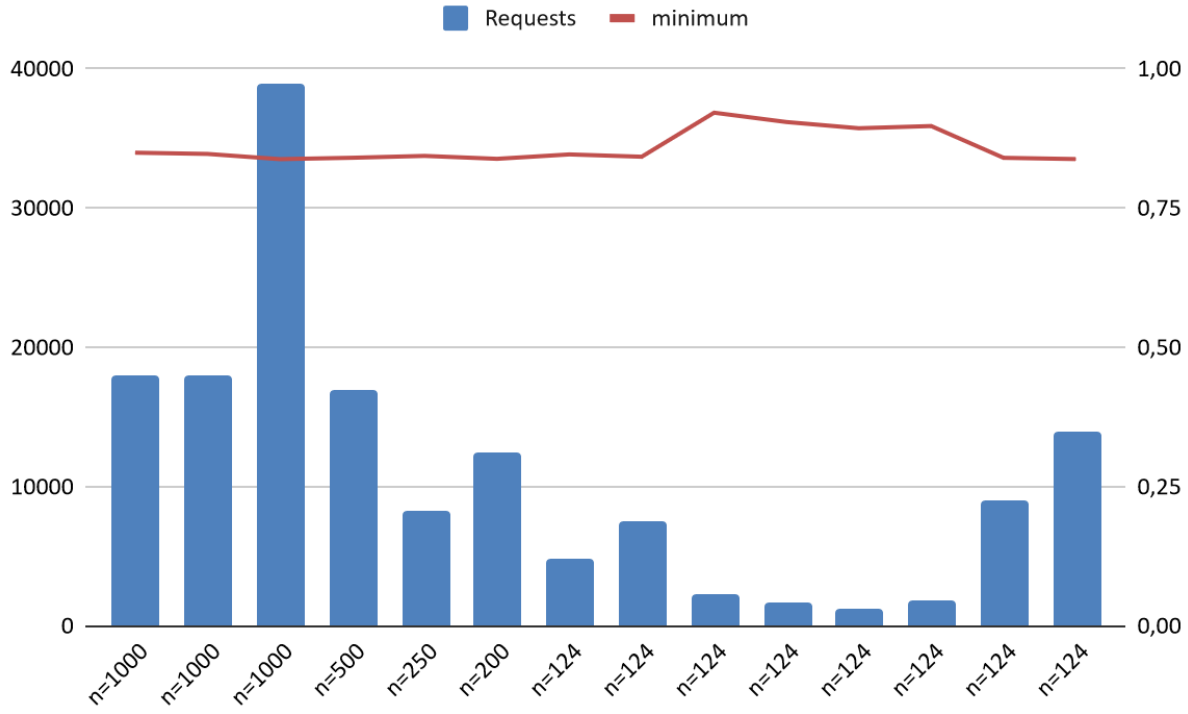
Depuración para alfa

Comenzamos probando con los valores que nos dieron los mejores resultados con la función de test. La evolución de nuestro algoritmo resulto ser mucho peor de lo esperado inicialmente, dado que se estanca de forma irremediable antes de bajar de 0.8. Concretamente nuestro mejor valor fuer **0,835561622** valor tras el cual la función comenzó a obtener valores superiores, sugiriendo que se trata de suerte.

El número máximo de generaciones que ejecutamos antes de comenzar otra prueba fueron 145.

Debido a la tardanza en la ejecución de estas pruebas (dado el incremento en magnitud del número de valores a tener en cuenta) decidimos explorar la reducción del tamaño de la población. El grueso de las pruebas fue realizado con n=1000. Curiosamente observamos una situación similar a la ocurrida con la función de test, donde podemos obtener soluciones comparables, con una cantidad mucho más reducida de individuos por generación, lo que hace que el algoritmo se ejecute de forma más rápida. Cabe comentar de todas formas que siendo los resultados valores muy alejados de la solución (0.0) podemos concluir que la ejecución de estas pruebas con menos individuos no sería capaz de llegar a la solución en contraste con la situación que se presentaba con la función anterior.





Podemos ver como reducir el número de individuos apenas empeora el resultado con un número de peticiones mucho más bajo.

Conclusiones

En esta práctica hemos aprendido primeramente la importancia de automatizar la recolección de datos en un formato fácilmente procesable, dado que sin el uso de este tipo de mecanismos, no se puede procesar la información de un volumen de pruebas suficiente para extraer ningún tipo de conclusiones en este tipo de sistemas.

También hemos aprendido la importancia de diseño modular de secciones de código, y establecimiento de variables globales con función de parámetro, permitiendo conformar una parte crucial a la hora de depurar los algoritmos genéticos, así como poder implementar diversos módulos con funcionamientos dispares (distintos tipos de selección de individuos).

Hemos podido apreciar la potencia de los algoritmos genéticos, que ajustándose correctamente pueden resolver problemas que son imprácticos de resolver mediante programación convencional. También hemos podido constatar la importancia de adecuar el algoritmo al problema que debe resolverse, dado la falta de eficacia que ha tenido este en la resolución del problema alfa, llegando apenas a una medida de 0.83, con objetivo 0.

En cuanto a resultados, concluimos que para la función de test los mejores valores de torneo era de un tamaño circa 20 y una mutación de 1, con un tamaño de población 100 resultando adecuado. En cuanto a la función alfa, no hemos encontrado ninguna solución que llegué a 0, al menos en menos de 150 generaciones. La mejor solución fue obtenida con una población de 1000, con torneo de 18 y mutación de 2 -un valor de 0,8355974553.