
PROGRAMACIÓN EVOLUTIVA

MEMORIA DE LA PRÁCTICA 3

ALEJANDRO BARRACHINA ARGUDO

ADRIÀ CARRERAS BAGUR

GRADO EN INGENIERÍA INFORMÁTICA

FACULTAD DE INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

Índice general

1. Capturas de funcionamiento	5
2. Observaciones de la práctica, arquitectura del código y ejecución	13
2.1. Observaciones	13
2.2. Arquitectura	13
2.3. Métodos Propios	13
2.4. Ejecución y compilación de la práctica	13
3. Distribución de trabajo y conclusiones	15
3.1. Distribución	15
3.2. Conclusiones	15

1 | Capturas de funcionamiento

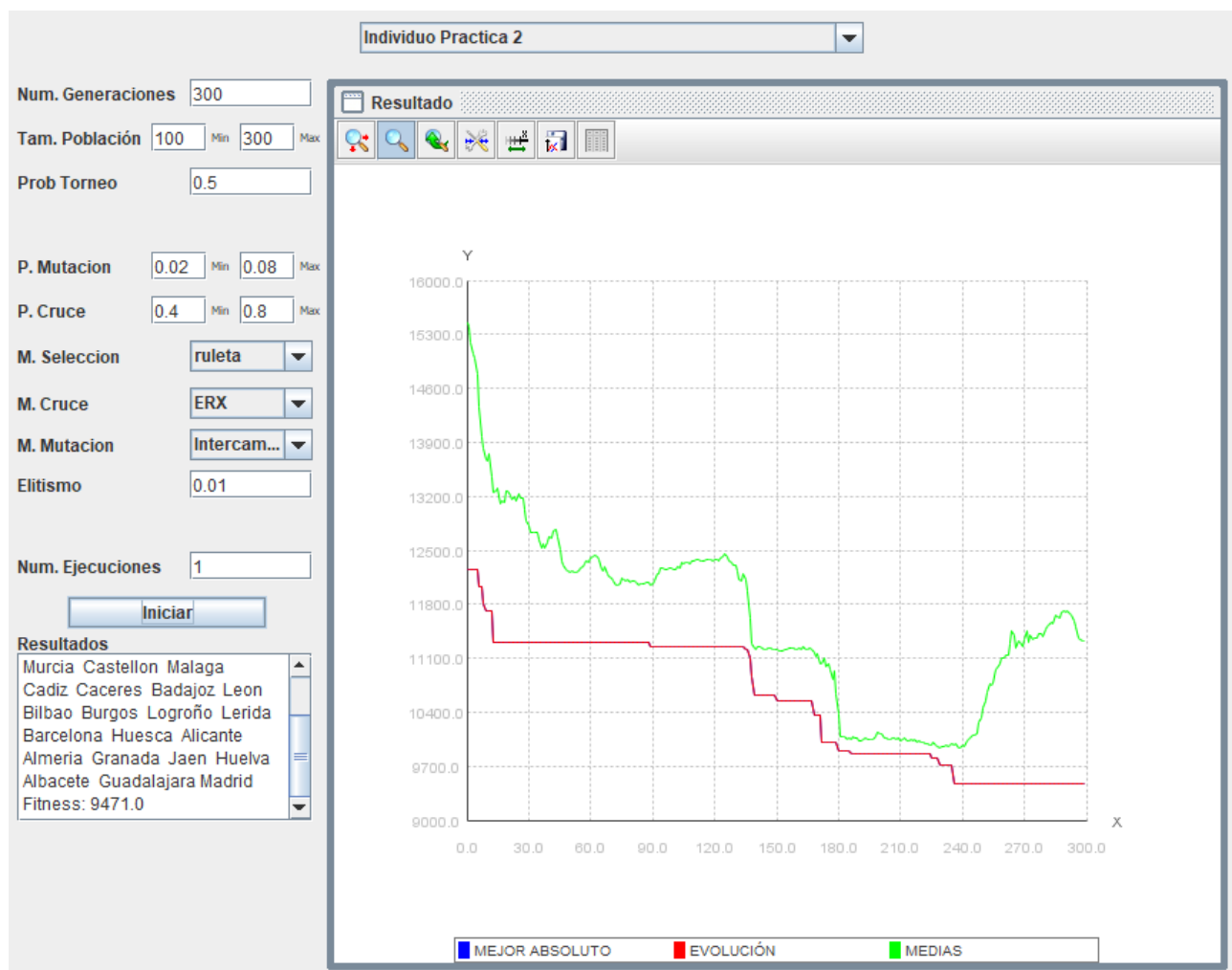


Figura 1.0.1: Captura 1

Se puede ver el funcionamiento de la práctica con los valores predeterminados, con un resultado por debajo de los 10.000.

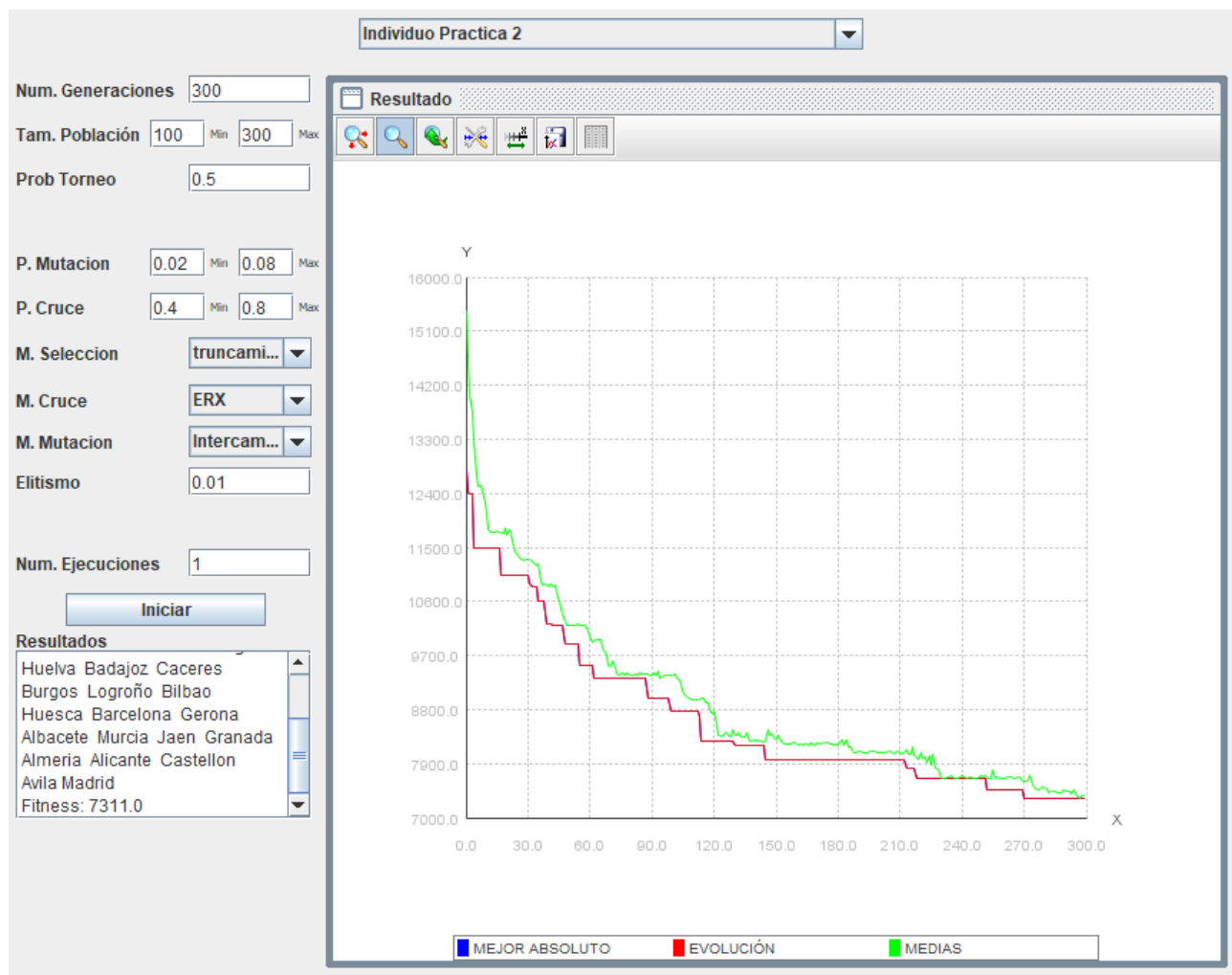


Figura 1.0.2: Captura 2

Solamente cambiando el método de selección a truncamiento se pueden notar mejoras, con una muy buena evolución.

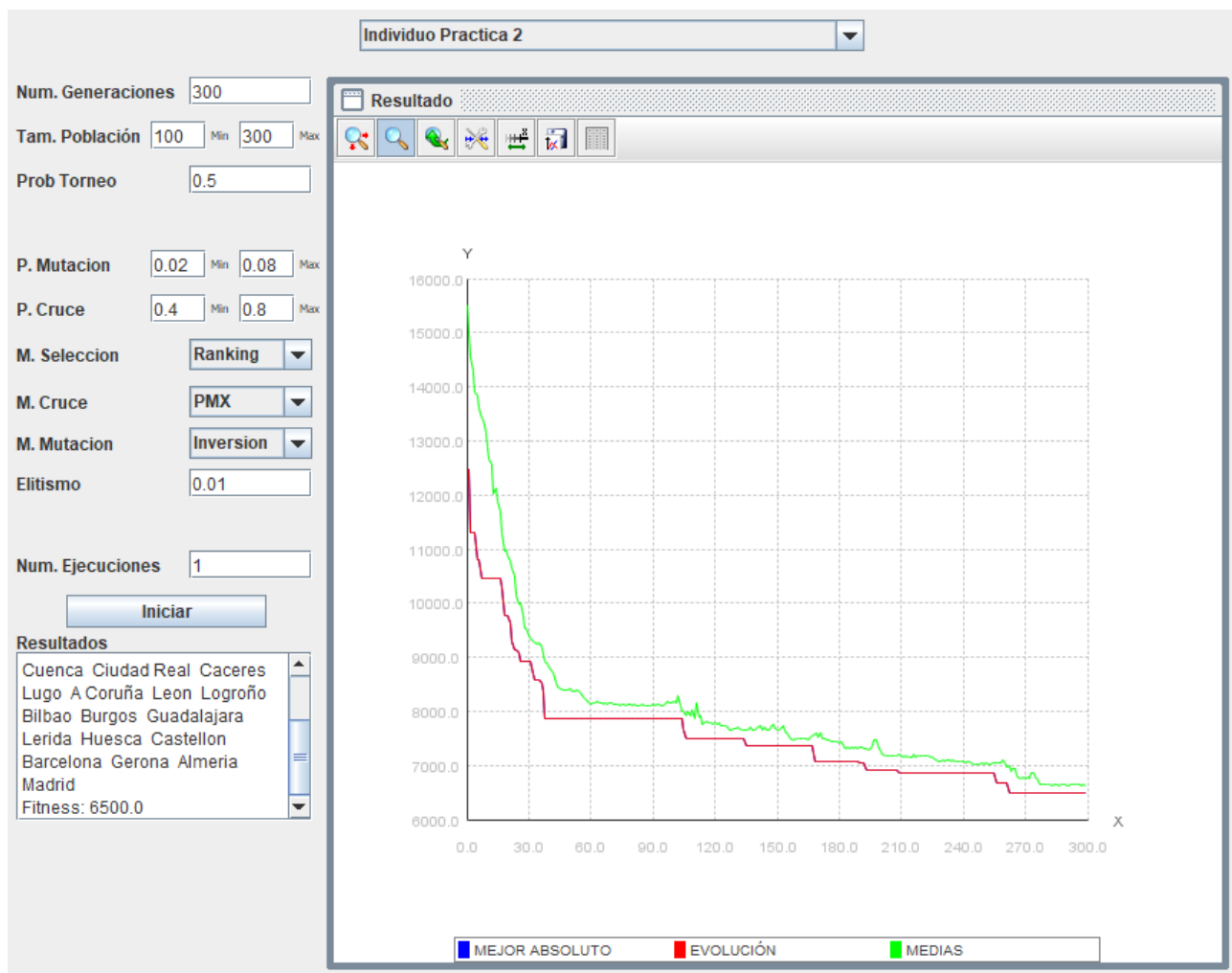


Figura 1.0.3: Captura 3

El método de Ranking consigue muy buenos resultados, juntándolo con el cruce PMX y la mutación por inversión es extremadamente eficiente y preciso.

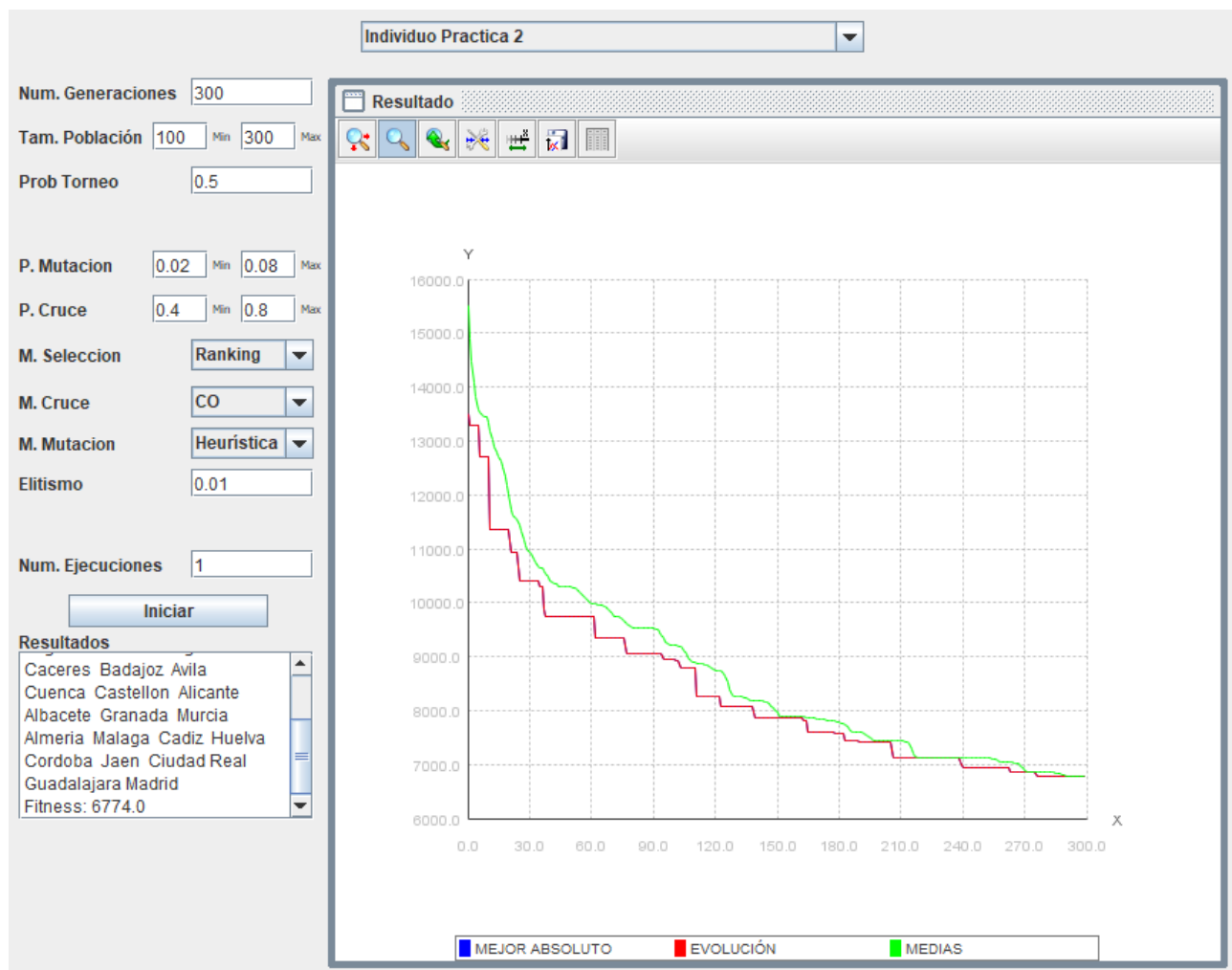


Figura 1.0.4: Captura 4

La mutación heurística da también muy buenos resultados aunque sacrificando eficiencia.

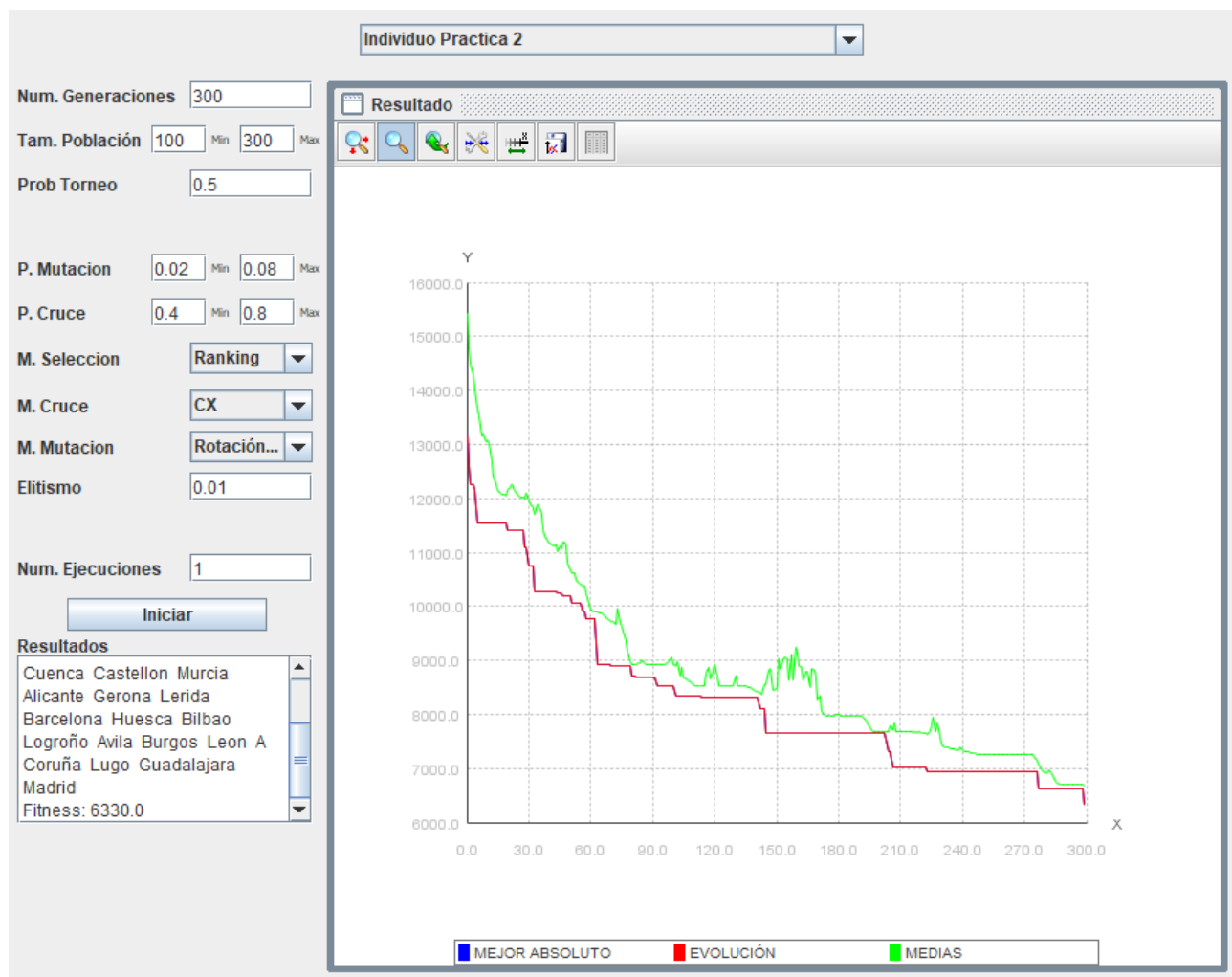


Figura 1.0.5: Captura 5

Nuestro método propio rotación heurística consigue resultados similares al anterior, también sacrificando eficiencia.

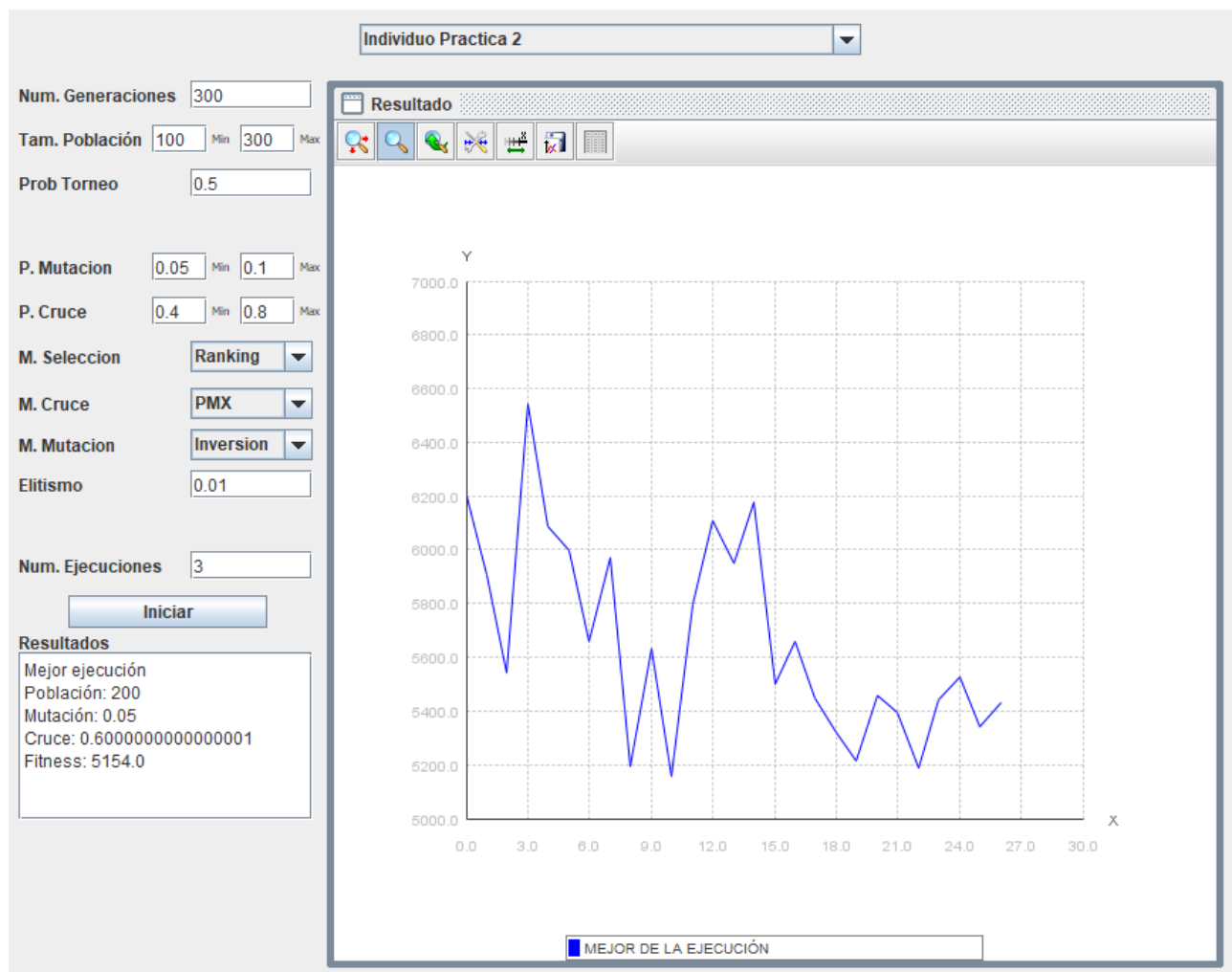


Figura 1.0.6: Captura 6

Con varias ejecuciones se consiguen resultados excepcionales.

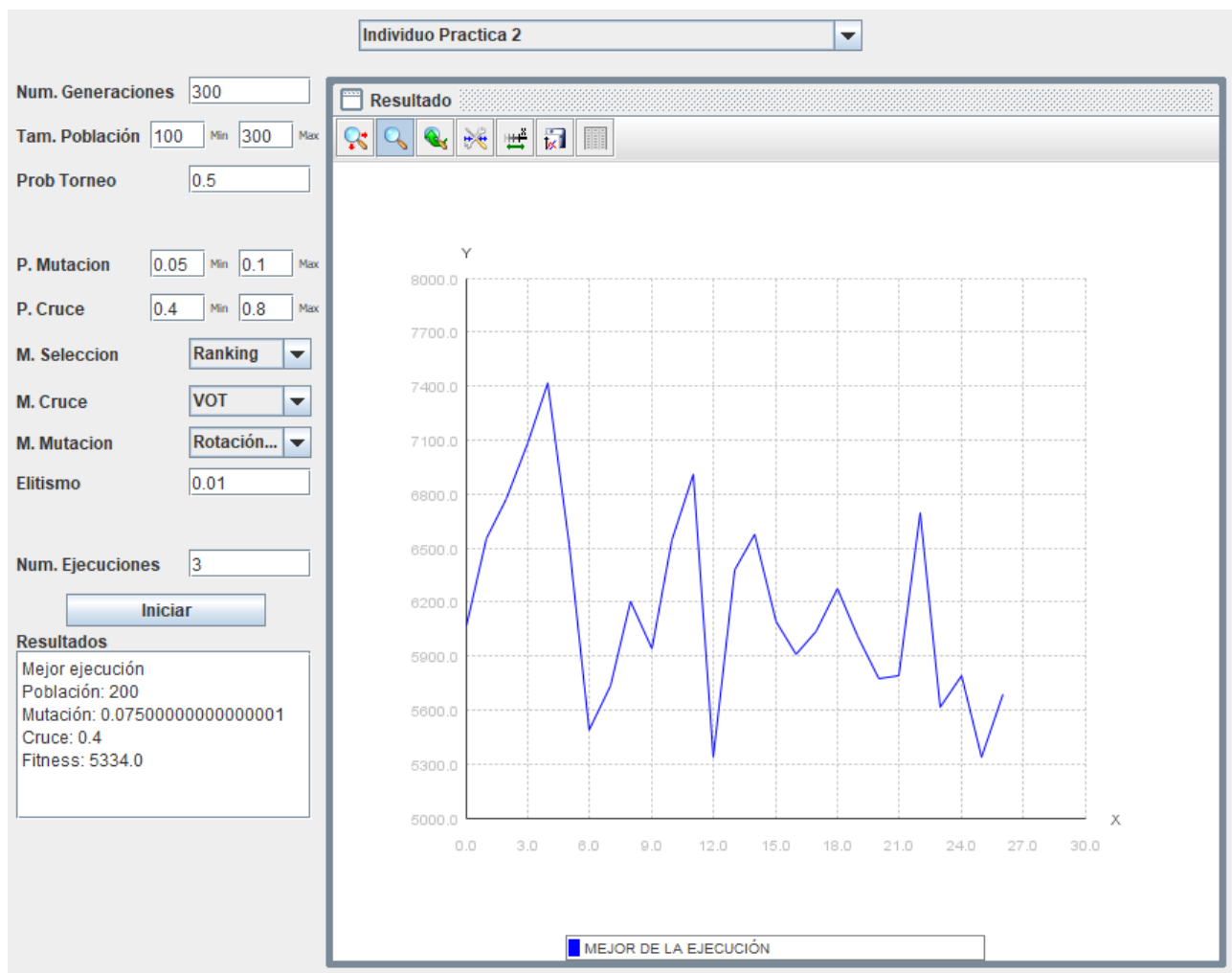


Figura 1.0.7: Captura 7

Se puede observar que no es común que el mejor resultado se dé con la máxima población, se consigue juntando unos buenos parámetros de cruce y mutación.

2 | Observaciones de la práctica, arquitectura del código y ejecución

2.1. Observaciones

Se han implementado los 2 tipos de inicializaciones distintas dentro de los árboles y Ramped and Half que combina las dos, dando muy buenos resultados.

Para el control de bloating se ha cambiado al cruce básico para que se hagan diferentes ejecuciones y así poder descartar individuos que empeorarían el fitness. Para evitar este cruce destructivo se ha añadido a la GUI el apartado Numero de Cruces, aunque hay que tener en cuenta que aumentarlo demasiado podría empeorar el tiempo de ejecución. Además para el bloating se ha utilizado el método de la penalización.

Las mutaciones Terminales y Funcionales son muy eficientes aunque pueden llevar a una convergencia prematura si no se utilizan adecuadamente. Las mutaciones de SubArbol y Expansion dan resultados muy buenos, pero si se utilizan demasiado o sin control de bloating podrían hacer que la ejecución tarde más. La mutación de Permutación es un punto medio, no alarga tanto los árboles y consigue buenas soluciones.

Se añadió la Regeneración de Población para momentos en los que los árboles pudieran quedarse atascados en resultados no óptimos, esta opción se puede elegir desde la GUI

Los métodos de selección que nos han dado mejores resultados como ya hemos visto en otras prácticas han sido el truncamiento y el ranking, siendo el truncamiento en esta practica mucho más eficiente.

2.2. Arquitectura

Para mantener todo el código organizado, se usa la siguiente estructura de paquetes:

- **go2:** paquete que engloba toda la práctica.
- **g02.cruces:** Paquete que reúne el único cruce de esta práctica bajo la interfaz Cruces.
- **g02.individual:** Paquete que contiene el individuo de la práctica 3 bajo la clase padre Individuo<T>.
- **g02.Selections:** Paquete para reunir las distintas selecciones usadas en la práctica bajo la clase padre Selection<T>.
- **g02.Ventana:** Paquete para la interfaz gráfica de la práctica y el método main.

Para una visión mas completa del proyecto, desde el navegador puede abrir el archivo `G02/target/site/index.html`

2.3. Ejecución y compilación de la práctica

Al ser un proyecto en maven se puede importar directamente desde eclipse con la opción de abrir proyectos del sistema de archivos.

Para compilar el proyecto solo hay que darle al botón de *run* y debería instalar las dependencias necesarias. Si no fuera así se puede ejecutar el comando *mvn install* para instalar todas las dependencias.

Así mismo, el proyecto se puede correr desde el propio eclipse usando el botón de *run*, usando la configuración de *aplicación java* y como clase principal `g02.Ventana.ventana`.

El archivo jar se encuentra en la carpeta `P01/target/`

3 | Distribución de trabajo y conclusiones

3.1. Distribución

Adrià ha hecho la GUI añadiendo las variables que se pueden manejar y el nuevo gráfico que compara la solución obtenida con la esperada; los métodos de mutación y ha trabajado con las clases del Árbol y el Cromosoma. Alejandro ha hecho la implementación del cruce de la práctica 3, la regeneración de la población, el control de bloating y también ha trabajado con las clases de Árbol e Individuo.

3.2. Conclusiones

Nos hemos dado cuenta de que para solucionar este problema hay que tener muy en cuenta la optimización del código en general, los árboles pueden llegar a ser muy costosos y hay que minimizar los individuos que no vayan a aportar a la solución.

Los métodos de selección que mejores resultados nos han dado han sido: Truncamiento y Ranking, siendo el Truncamiento el más efectivo, aunque más propenso a quedarse estancado.

Las mutaciones han sido muy útiles y pueden afectar mucho al resultado, cada una tiene su función y se utiliza de una manera distinta, aunque la Permutación ha dado muy buenos resultados.

El control del Bloating es esencial en este problema.