Programación Evolutiva
Práctica 2

Roberto Plaza Hermira

Gorka Silva Ramón

Detalles de implementación

1. Asignación de pista

Lo más destacable es la función *asignarPista* que se encarga de ir asignando a los vuelos una pista y un *TLA*. Esta función asigna los vuelos según el orden en que se encuentran en el *Gen*. Cada elemento busca su pista óptima dependiendo del valor de *retardo*, donde:

$$retardo = (TLA - \text{mínimo } TEL)^2$$

El mínimo *TEL*, es el menor valor de *TEL* del vuelo en cualquiera de las pistas. Lo interesante de esta función de asignación es que devuelve un valor mínimo de 11.25, cuando en el enunciado se indica que puede llegar a un valor de 6.25.

Se podría suponer que esta función no encontraba ese mínimo ya que en caso de tener dos opciones con el mismo *retardo* siempre se quedaba con la primera opción elegida. Pero hemos descubierto que no es así porque hemos probado un método que elegía, con un valor aleatorio, entre cualquiera de las opciones que tenían el mismo *retardo*. Lo cuál hacía que con mismos *Genes* generasen diferentes asignaciones de pista y por lo tanto diferentes valores función. Aun así el mínimo de todas las pruebas hechas ha sido 11.25.

2. GenPr2

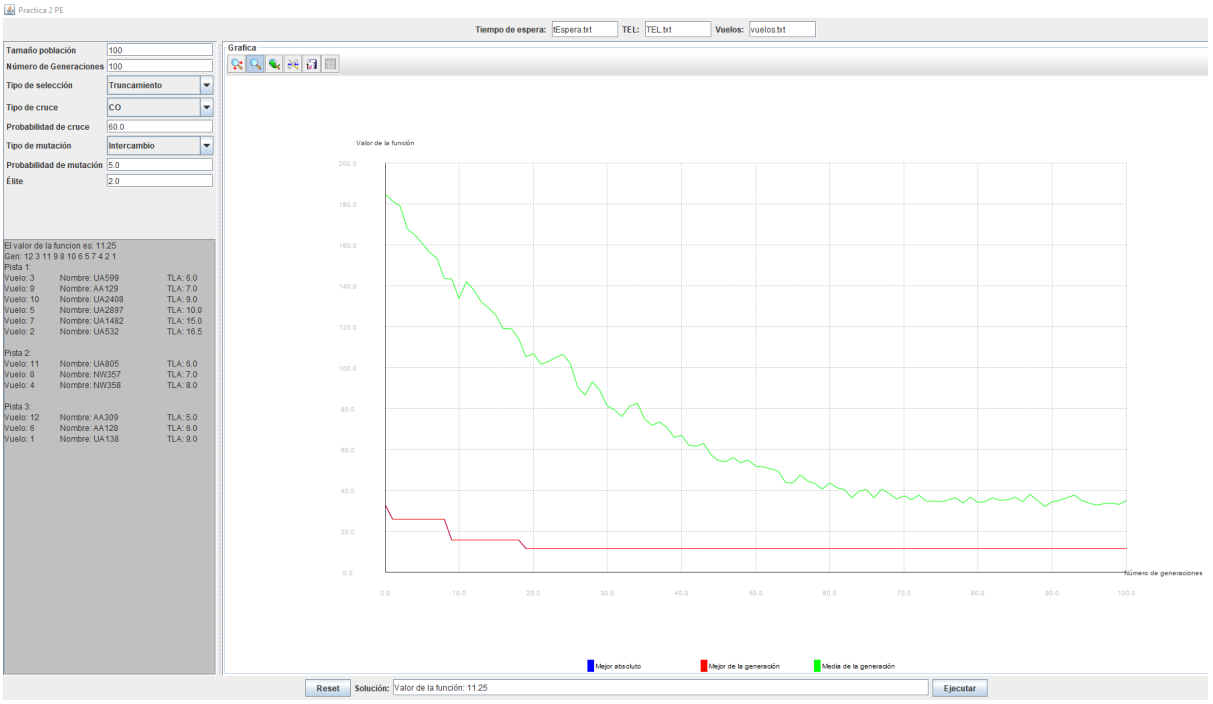
El gen consiste en una lista de alelos cuyo tamaño es el número de vuelos. Cada alelo es un int que corresponde al índice de la lista <vuelos> en la cual está contenido su nombre y tipo. Cuando se inicializa, se da a cada posición el valor del índice, posteriormente se realizan un número aleatorio de intercambios entre posiciones aleatorias del gen. De esta manera se obtienen diferentes permutaciones de los vuelos.

Comportamiento de las diferentes combinaciones

1. Ejemplo guía práctica.

Archivos: tEspera.txt, TEL.txt, vuelos.txt

Todas las combinaciones de cruces, selecciones y mutaciones llegan al óptimo (11.25) muy rápido. La mayoría de las combinaciones tardan alrededor de 20-30 generaciones con casos excepcionales en los que tardan como mucho unas 40 generaciones.



-Estos resultados también se obtienen sin elitismo

2. Ejemplo 1: 50 vuelos, 5 pistas.

Archivos: tEspera.txt, TEL1.txt, vuelos1.txt

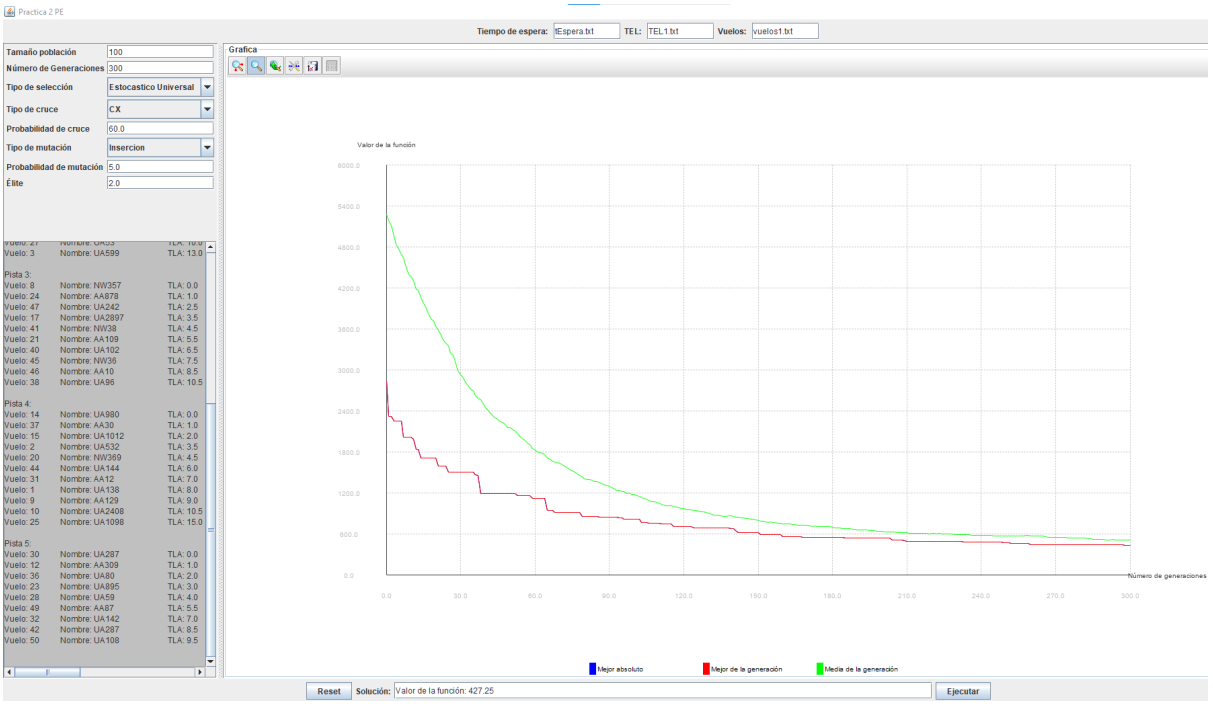
Este caso de prueba tiene un alto número de vuelos (50) y pocas pistas (5), por lo que el resultado crece enormemente. Se debe tener en cuenta que el archivo TEL para este caso ha sido generado aleatoriamente con números en el intervalo 0-25, por lo que habrá vuelos con una gran diferencia entre su TLA y su menor TEL. Al usar como función de fitness (TLA -TEL pista)^2 en lugar de (TLA - menor TEL)^2, los resultados cambian enormemente. Debido al alto número de permutaciones, este caso de prueba lo estamos ejecutando con 300 generaciones.

Guía de uso: para cambiar el tipo de fitness a utilizar se hace cambiando la línea comentada en la función “recalcularFenotipo()” de ”Individuo”.

- (TLA - menor TEL)^2

-Los cruces OX y OX-PP, independientemente de la combinación de selección y mutación se mantienen en el intervalo (900 - 1200).

-El cruce CX se mantiene en el intervalo (400 - 650), siendo 427 el mínimo obtenido a lo largo de múltiples ejecuciones.



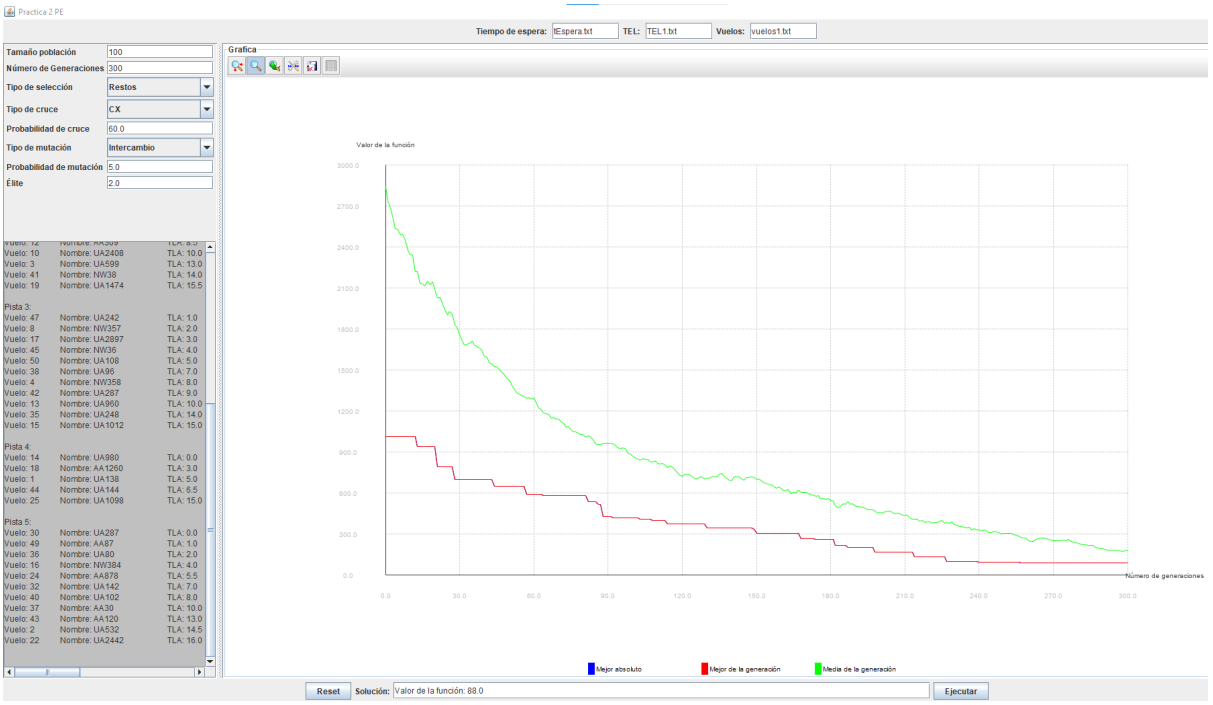
-El resto de combinaciones se mantienen en el intervalo (600 - 900) de manera aproximada.

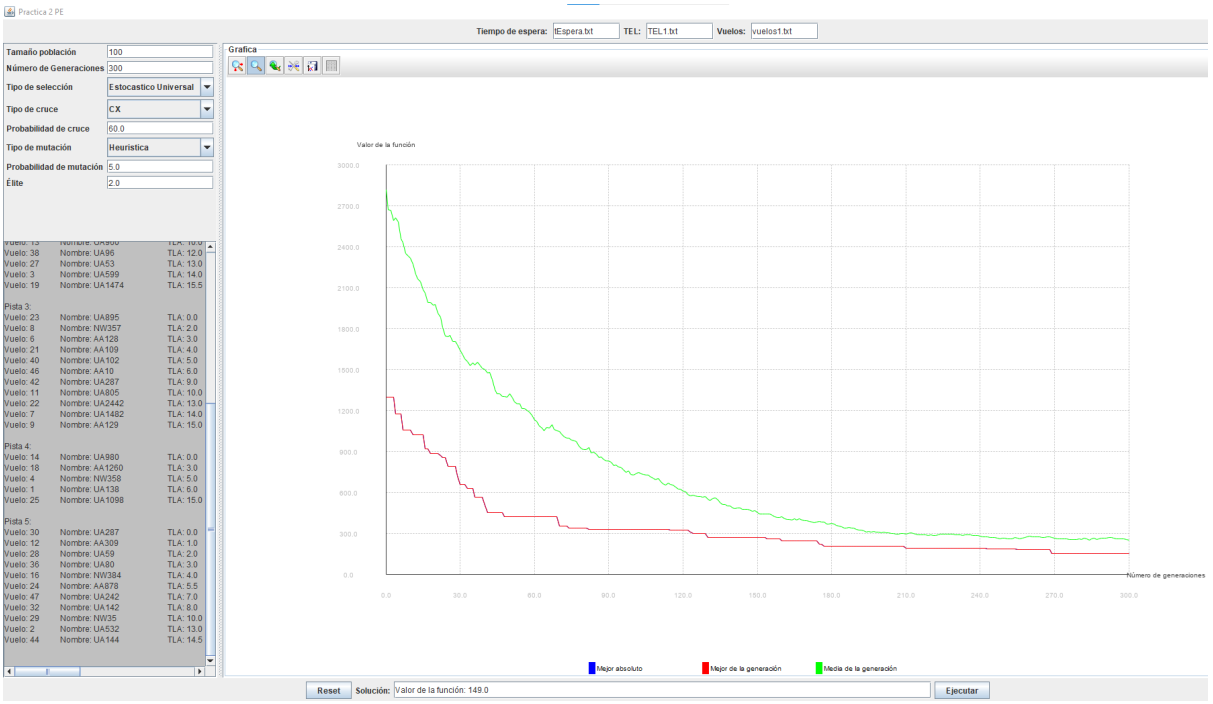
- (TLA - menor TEL pista)^2

Si bien el valor es obviamente menor, el rendimiento de los diferentes cruces, mutaciones y selecciones es muy similar al caso anterior, sino idéntico.

-Cruces OX y OX-PP:

-Cruce CX: se mantiene en el intervalo (100 - 250) con un mínimo de 88.



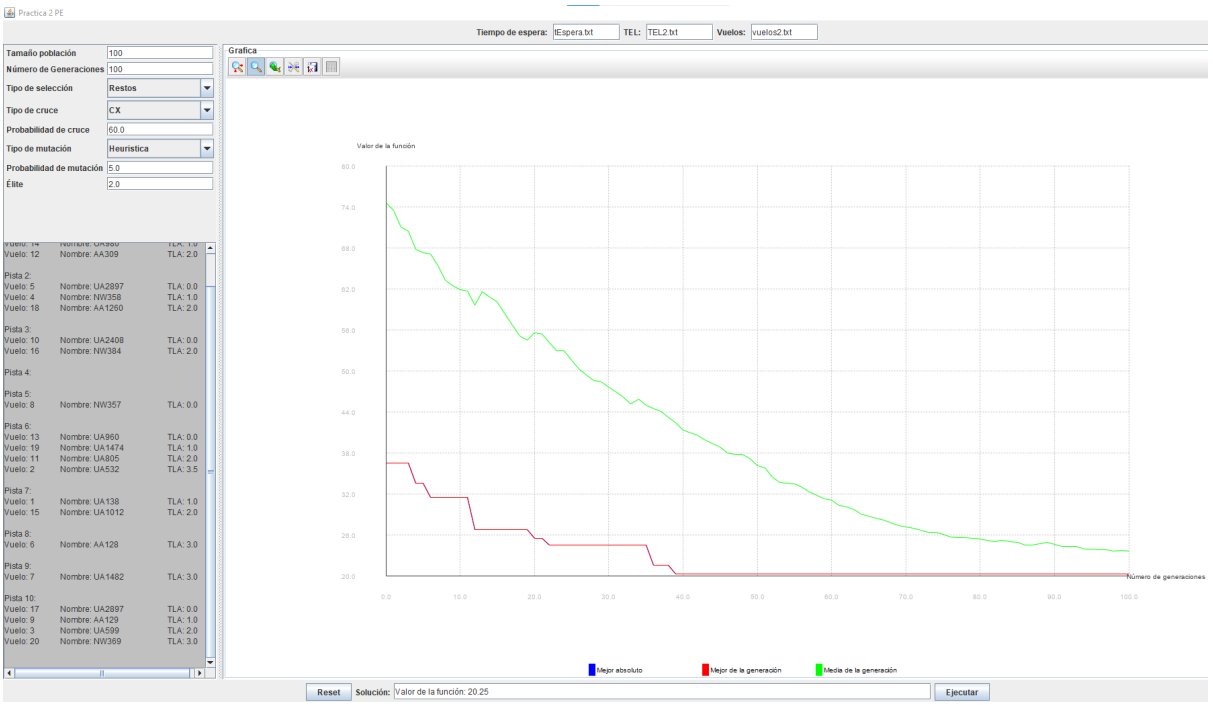


-Resto de combinaciones: se mantienen en el entorno de (250 - 450) salvo algunas ejecuciones con resultados tanto mayores como menores.

3. <u>Ejemplo 2: 20 vuelos, 10 pistas.</u>

Archivos: tEspera.txt, TEL2.txt, vuelos2.txt

Todas las combinaciones llegan al resultado de forma rápida. Generalmente se llega al óptimo (20,25), en torno a las 10-40 generaciones (varía mucho dependiendo de cada ejecución), con la excepción de algunas ejecuciones en las que no se llega al óptimo. En estos casos el resultado se queda bastante cerca del óptimo, no superando los 25.



Como se puede observar en las imágenes anteriores, los cruces CO y CX producen una gran reducción en la media, llegando a estar muy cerca del mejor de la generación. Esto es obvio en los ejemplos con mayor número de generaciones, ya que se puede ver claramente el descenso de la media y como prácticamente llega al mínimo.

Quitar el elitismo hace que desaparezca la cualidad que tenían los cruces CX y CO de converger con la media de las generaciones.

Cruce CX sin elitismo:

Cruce OX sin elitismo:

Reparto de tareas

La parte que hemos realizado en conjunto ha sido:

- Función Asignar Pista: esta función ha sido la que más problemas nos ha dado para encontrar el mínimo y por lo tanto ambos hemos realizado una versión de ella para intentar obtener resultados diferentes.
- GUI: las adaptaciones correspondientes para esta práctica.
- Adaptación de alguna función de algoritmoGenetico.

Roberto se ha encargado principalmente de:

- IndividuoPr2: la estructura del individuo con todo lo que eso conlleva: representación de la solución, la generación de la población, el Gen y la función de fitness entre otros.
- Selección por Ranking.
- Cruce OX.
- Cruce OX-PP.
- Mutación por Inversión.
- Mutación por Intercambio.

Gorka se ha encargado principalmente de:

- Cruce CO.
- Cruce CX.
- Cruce PMX.
- Mutación Heurística
- Mutación por Inserción.
- Depuración del código corrigiendo errores.
- Función fitness de menor TEL de pista