Práctica Final:
Resolución al problema del Viajante
Mediante Algoritmos Genéticos

Computación

Segundo Semestre. Curso Académico 2021/2022

Ignacio Marcos Serrano

Índice

1 Planteamiento del Problema:

El objetivo de esta práctica es desarrollar, mediante algoritmos genéticos, una solución o soluciones al problema del viajante, uno de los problemas más famosos en la computación y algoritmos de optimización. Para este problema partimos de un conjunto de N ciudades y disponemos a partir de dicho conjunto, una matriz de coste M ($N \times N$) y de números reales positivos, tal que la entrada (i, j) es una medida del coste que supone ir de la ciudad i a la ciudad j, suponiendo que el coste de ir de la ciudad j a la ciudad i es exactamente la misma. Además, suponemos que es posible ir de cualquier ciudad a otra diferente.

Dicho esto, el objetivo del problema será el de determinar una ruta que permita pasar exactamente una vez por cada ciudad volviendo a la ciudad inicial en el menor coste posible. Teniendo esto en cuenta, el objetivo de la práctica será estudiar el comportamiento del algoritmo ante la variación de los diferentes parámetros.

Para la resolución de este problema deberemos tener claras tanto la función de coste a optimizar, como la codificación de los cromosomas. En nuestro caso particular, al tratarse del coste de ir de una ciudad a otra, utilizaremos **como métrica la distancia entre ciudades**, por comodidad de código, **como métrica de *fitness* invertiremos el valor de la distancia** con el objetivo de maximizar este valor, en particular;

$$\textbf{fitness} = 1 \,/\, \sum_{i=1}^{N} dist(i,j) \quad siendo\ j = \begin{cases} i+1 & si\ i < N \\ 1 & si\ no \end{cases}$$

Por otra parte, como tenemos que pasar exactamente una vez por cada ciudad, utilizaremos la **codificación por permutaciones de los cromosomas**, de esta forma nos aseguramos de que cada cromosoma tenga un tamaño N en el que cada valor aparece solo una única vez. Teniendo esto en cuenta, nuestra población será una matriz ***P* de tamaño *N_invividuos x N_ciudades*.**

Por último, cabe comentar que para este algoritmo de ha implementado una **clase *Genetic.m***, la cual contendrá los parámetros y métodos necesarios para la implementación del algoritmo genético y un **archivo *MAIN.m*** en el que llamaremos a las instancias correspondientes de dicha clase, con los métodos adecuados para la realización de las pruebas.

2 Parámetros del Problema:

Para tener una visión más clara de cómo se han realizado los experimentos de este problema, pasaremos a describir brevemente los **parámetros que podremos variar entre experimentos**:

- Número de ciudades N y Matriz de distancias

El número de ciudades lo establecerá el usuario al ejecutar el archivo *MAIN.m,* este valor debe ser un número entero positivo mayor que 3.

Una vez establecido en número de ciudades, el algoritmo calculará automáticamente y **de forma aleatoria** la matriz de distancias *M,* para la creación de esta matriz se podrán ajustar los valores mínimos y máximos que pueden obtener estas distancias, por defecto, está fijado a valores entre ***1 y 100*** **siendo** ***0*** **en la diagonal principal** (el coste de ir de una ciudad a sí misma). Para una menor ambigüedad entre los experimentos estableceremos la *seed* al mismo valor entre experimentos para comparar los resultados con las mismas distancias.

- Método de asignación de probabilidades para la selección de progenitores

Este valor lo estableceremos únicamente cuando estemos ejecutando el algoritmo con el *método de selección de progenitores* del ranking.

Se ha implementado el método ***lineal*** **y** ***exponencial***, para el método lineal se establecerá la probabilidad a;

$$P(i) = \frac{2-s}{\mu} + \frac{2i(s-1)}{\mu(\mu-1)} \quad \text{siendo } \mu\text{: Tamaño de la población y } s \in [1,2]$$

podrá establecer el valor de *S* el cual, si no se indica, tendrá como **valor por defecto a 1.5**. Para el método exponencial estableceremos el valor de *C* a;

$$P(i) = \frac{1-e^{-i}}{c} \qquad \boldsymbol{siendo\ c = \frac{2n(n-1)}{6(n-1)+n}}$$

- Método de selección de progenitores

Para la selección se han implementado los métodos de ***la ruleta, del ranking* y *del torneo***. Los k contrincantes podrán establecerse, en caso de no hacerse, competirán la **mitad de la población**. Para el método del *ranking* se podrán establecer los métodos explicados anteriormente y para el del *torneo* se podrá especificar si es **con repetición o sin ella.**

- Método de cruzamiento

Para el cruzamiento se han implementado los métodos del ***Cruzamiento Parcialmente Mapeado* y *Cruzamiento por ciclos***. Para ambos métodos, siempre obtenemos dos hijos para cada dos progenitores.

- Método de selección de supervivientes

Para la selección se han implementado los métodos del ***Reemplazamiento basados en la edad* y *Reemplazamiento basados en el fitness***, para este último caso, específicamente se ha implementado el ***Genitor***.

- Presencia de mutaciones

Para las mutaciones se han implementado los métodos del ***intercambio* y *sacudida***. Dando la opción de **no realizar mutaciones**.

Cabe destacar, que, además de estos parámetros, se ha dado la opción de realizar, si se desea, una condición de parada conocida como ***Early Stopping*** en la que, dada una determinada paciencia, establecida por el usuario, **si pasan ese número de generaciones sin obtener mejoría**, el algoritmo se detendrá, habiéndose alcanzado o no una solución.

3 Experimentación:

3.1 GridSearch:

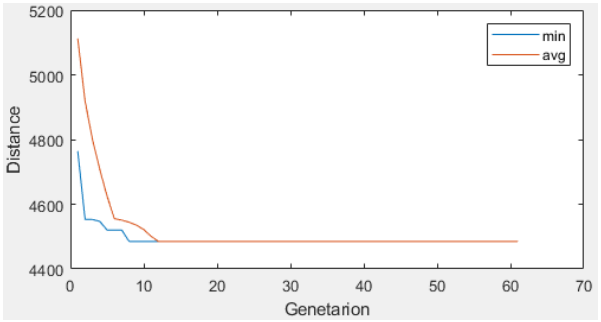
En primer lugar, como primer experimento, se ha realizado un *GridSearch* en el que se realiza una ejecución del algoritmo por cada posible combinación de todos los parámetros mencionados en el apartado anterior y así ver cuáles son las combinaciones que mejor resuelven el problema.

En nuestro caso, obtenemos un total de 60 posibles combinaciones. Para todas ellas trabajaremos con *100 Ciudades*, *20 individuos*, una *probabilidad de mutación del* 60% (Para los casos en los que se realicen mutaciones) y *600 generaciones* estableciendo la paciencia del *Early Stopping* mencionado anteriormente, a *30 generaciones.*

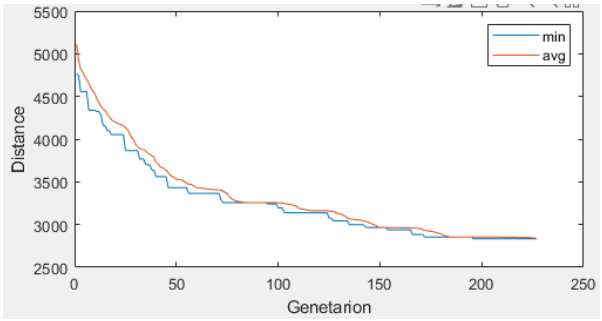
Después de ejecutar el *GridSearch* podemos observar varios tipos de gráficas:

(Todas las gráficas del *GridSearch* están disponibles en la carpeta *Figures/GridSearch*)

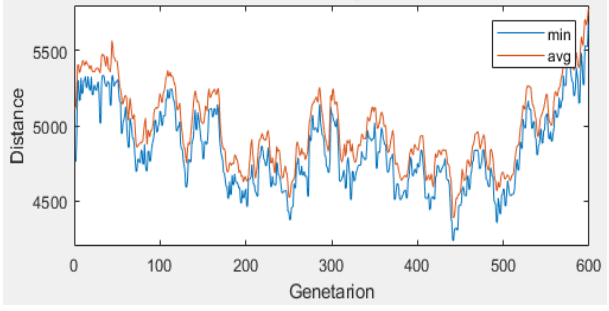
- <u>Convergencia rápida sin llegar a una solución óptima</u>



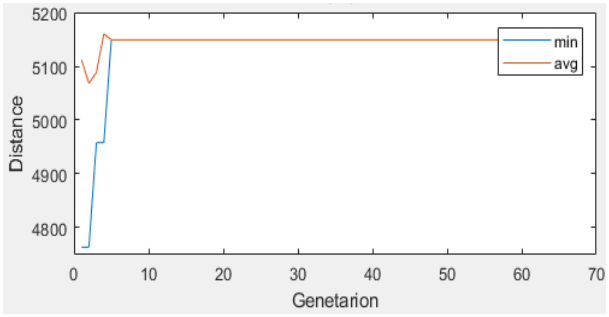
- <u>Convergencia lenta sin llegar a una solución óptima</u>



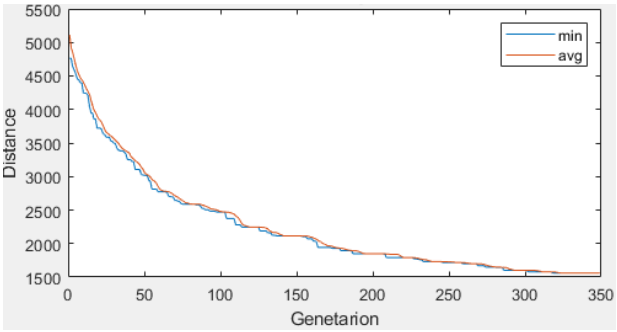
- <u>Sin convergencia ni solución óptima</u>



- Empeora rápidamente y sin solución óptima



- **Convergencia lenta que SÍ llega a una solución óptima**



En total, para esta configuración tenemos que 9 modelos llegan a una solución óptima. Algo que tienen en todas es que todas, el **método de selección de supervivientes** es de *Genitor* y todas **utilizan mutaciones** de tipo *Exchange.* Podemos asumir pues que las mutaciones de tipo *sacudida* quizá sean demasiado agresivas para este problema y la ausencia de mutaciones tampoco nos lleva a soluciones óptimas.

Con relación al **método de cruzamiento,** no hay distinción que nos lleve a decir cual de los dos funciona mejor, la mitad son de tipo *PMX* y la otra mitad por *Ciclos.* Lo mismo sucede con relación al **método de selección de progenitores,** nos encontramos de todos los tipos entre los modelos que convergen con solución.

Teniendo esto en cuenta vamos a realizar otra tanda de experimentos variando los parámetros del modelo con mejores resultados, en este caso con la siguiente configuración:

- Número de ciudades: *100*
- Número de individuos: *20*
- Método de selección de progenitores: *Ruleta*
- Método de cruzamiento: *PMX (Cruzamiento Parcialmente Mapeado)*
- Método de selección de supervivientes: *Genitor*
- Mutaciones: *Intercambio*

Coste: 1450

Para los siguientes experimentos seguiremos utilizando la misma *seed* para seguir experimentado con la misma matriz de distancias.

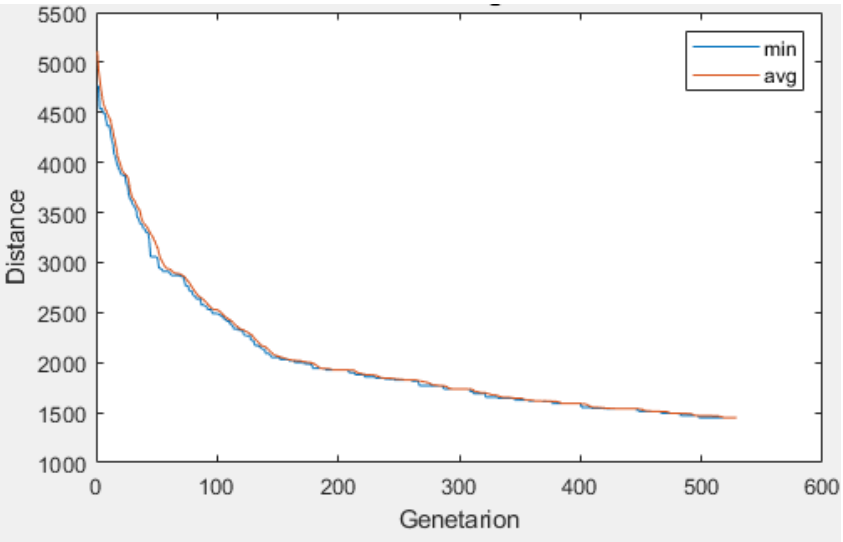
3.2 Por parámetros:

Como se ha explicado anteriormente, esta siguiente tanda de experimentos se realizará variando los parámetros de la mejor configuración obtenida en el *GridSearch*, en particular, variaremos los siguientes parámetros:

Individuos	Prob. Mutación	Select. Progenitores	Cruzamiento	Select. Supervivientes
20	0.6	Ruleta	PMX	Genitor

Mutación	Generaciones	NO. Padres selección	EarlyStopping Patience
Intercambio	529 (Con EarlyStopping)	Individuos/2	30 Generaciones

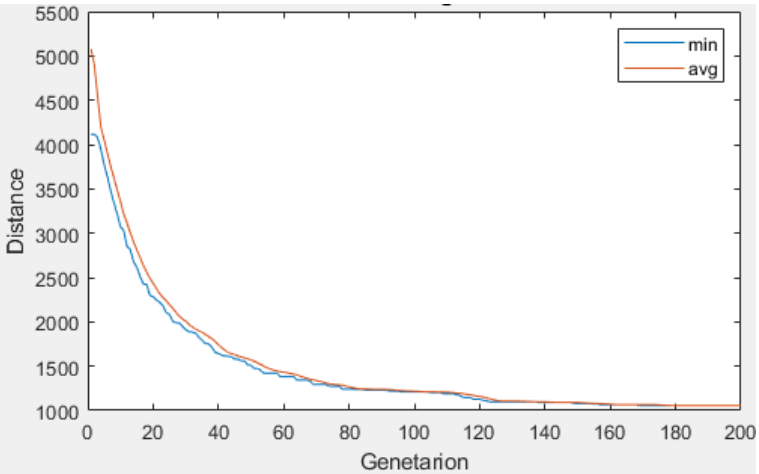
Nota: *El número de generaciones está siempre fijado a 600, pero a partir de este momento hablaremos de generaciones como el número de generaciones a las que llega hasta dejar de mejorar el modelo. Medido con el EarlyStopping Patience.*



Gráfica del modelo a variar. Coste: 1450

- **Número de individuos**

Individuos	*Coste*	*Tiempo [s]*	*Generaciones*
20	1450	0.18	451
60	1259	0.44	529
100	1303	0.43	192
260	1056	0.98	275
500	1029	2.17	238
800	1055	2.56	274



Grafica 800 individuos

Podemos observar cómo, en general, cuantos más **individuos** introducimos en nuestro modelo, mejores **resultados** obtenemos, algo que, a priori, era intuible. Sin embargo, vemos como a partir de los 250 individuos no obtenemos diferencias especialmente notables. De la mano con lo anterior viene el **tiempo** que necesita cada modelo para llegar a la solución, cuanta más población, más tiempo de ejecución, sin embargo, entre 20 individuos y 800 el tiempo de ejecución apenas varía en 2 segundos y medio. Con relación al número de **generaciones** que toma cada modelo en estancarse en una solución no se puede sacar una relación a priori con el aumento de individuos.

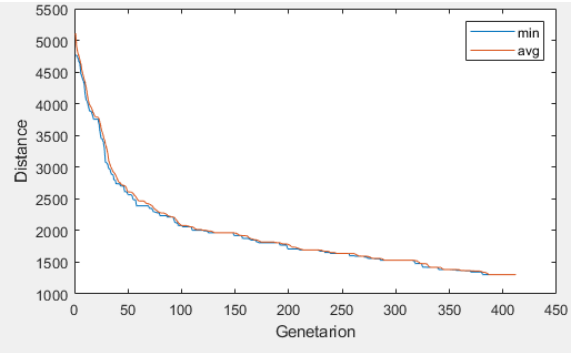
Debido a que a partir de los 260 no mejoran significativamente los resultados, que con ese número de individuos seguimos manteniendo un modelo con un tiempo de ejecución por debajo del segundo y además, no necesita un número especialmente alto de generaciones para converger, elegiremos ese valor para el modelo final.

- **Selección de progenitores**

Para este experimento, aparte de variar entre los 4 métodos de selección de progenitores, para el método del *ranking* con *ranking lineal*, **variaremos el parámetro s**. Y para todos los métodos, además **variaremos también el número de progenitores** que seleccionaremos en cada generación.

1. Ruleta

NO. Padres Seleccionados	***Coste***	***Tiempo [s]***	***Generaciones***
4	2163	0.14	324
8	1799	0.12	330
10	1450	0.19	529
16	1305	0.19	412
20	1352	0.24	482



Gráfica 16 progenitores

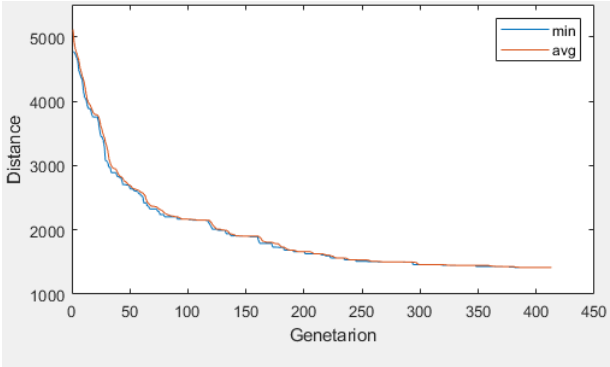
2. Ranking

2.1 Ranking lineal

Para este apartado en el que variamos, no solo el **número de progenitores** que se seleccionan en cada generación, sino que también variamos el **parámetro s**, se mostrará solo la tabla con la variación de progenitores por dos motivos. En primero es por simplificación y el segundo porque en todas las iteraciones, el único valor de s que reduce significativamente el coste es **1.2**.

S = 1.2

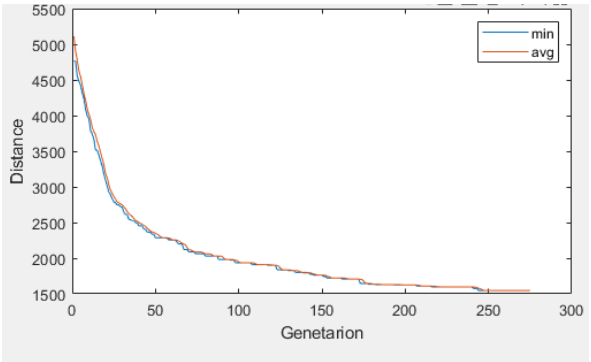
NO. Padres Seleccionados	***Coste***	***Tiempo [s]***	***Generaciones***
4	2068	0.13	400
8	1686	0.14	353
10	1587	0.18	437
16	1414	0.19	413
20	1475	0.18	350



Gráfica 16 progenitores

2.2 Ranking exponencial

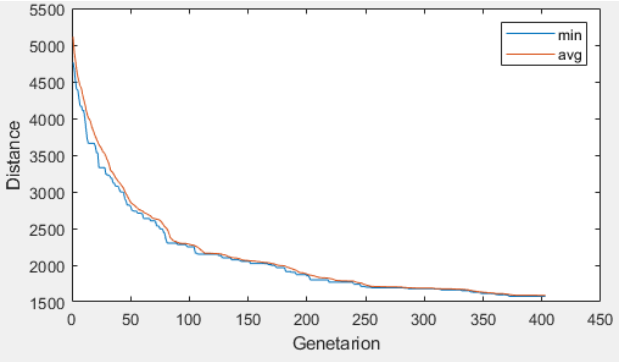
NO. Padres Seleccionados	***Coste***	***Tiempo [s]***	***Generaciones***
4	2178	0.09	285
8	1689	0.11	310
10	1737	0.13	304
16	1614	0.13	268
20	1547	0.15	275



Gráfica 20 progenitores

3. Torneo con repetición

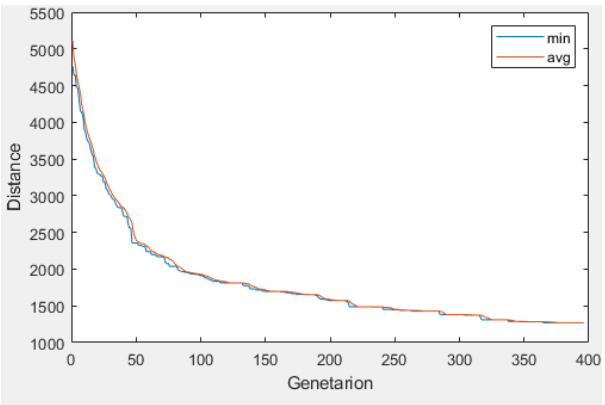
NO. Padres Seleccionados	***Coste***	***Tiempo [s]***	***Generaciones***
4	2134	0.1	311
8	1411	0.22	548
10	1622	0.17	391
16	1576	0.24	403
20	1669	0.2	289



Gráfica 16 progenitores

4. Torneo sin repetición

NO. Padres Seleccionados	***Coste***	***Tiempo [s]***	***Generaciones***
4	1776	0.16	485
8	1969	0.11	267
10	1495	0.18	400
16	1390	0.26	447
20	1270	0.23	365



Gráfica 20 progenitores

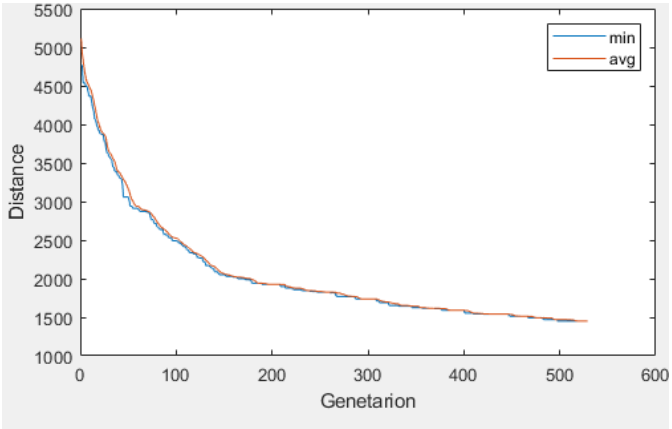
Los resultados obtenidos por los distintos métodos son bastante similares entre sí, sin embargo, los métodos de la *ruleta* y del *torneo sin repetición* son algo mejores que el resto. Para todos los modelos, en general, a excepción del *torneo sin repetición* el número de padres seleccionados que obtiene mejores resultados es de 16, en proporción al número de individuos esto supondría alrededor del 80%. Nos quedaremos pues con un porcentaje de entre el 60% y 80% para el modelo final.

Con relación al **tiempo de ejecución** entre el número de progenitores no hay diferencia notable, por lo que no es un parámetro que afecte al tiempo de ejecución.

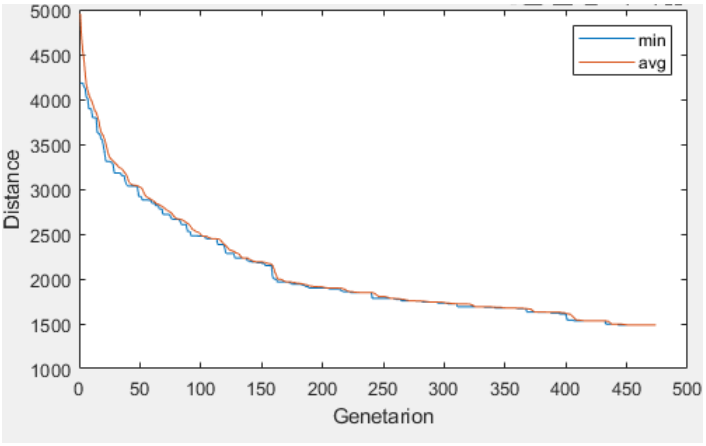
- **Cruzamiento**

En este caso comprobaremos simplemente la diferencia entre el método del *Cruzamiento Parcialmente Mapeado* y el *Cruzamiento por Ciclos.*

Método	***Coste***	***Tiempo [s]***	***Generaciones***
PMX	1450	0.2	529
Ciclos	1559	0.47	474



Gráfica PMX



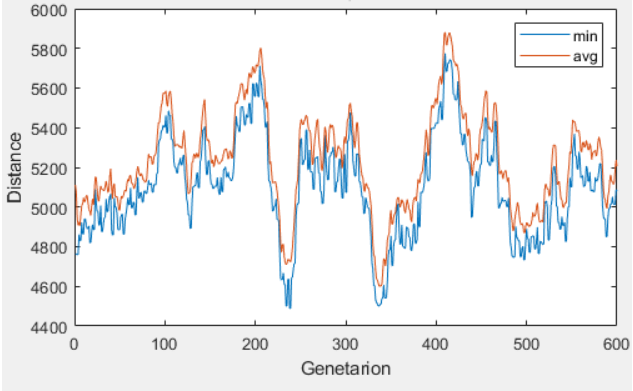
Gráfica Ciclos

En este caso tenemos conclusiones bastante parecidas al experimento anterior, que no hay realmente mucha diferencia, aunque mínimamente notable, entre el coste obtenido por ambos. Sabiendo esto, elegiremos el método de *Cruzamiento Parcialmente Mapeado* debido a que tarda menos de la mitad en ser ejecutado, aún tardando unas pocas generaciones más en converger.

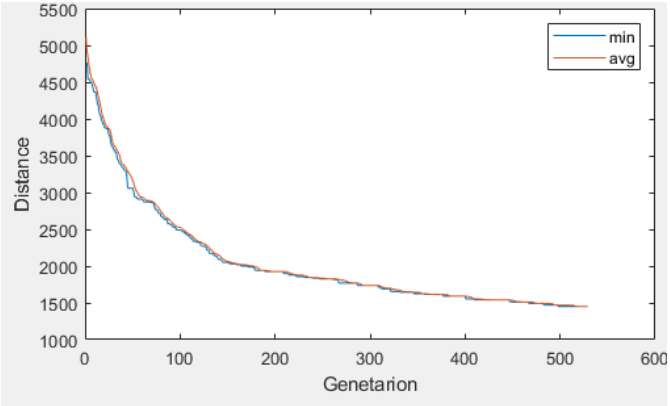
- **Selección de supervivientes**

Nuevamente compararemos entre dos métodos, la selección por *edad* y la selección *genitora.*

Método	***Coste***	***Tiempo [s]***	***Generaciones***
Edad	4488	0.16	600
Genitor	1450	0.2	529



Gráfica Edad



Gráfica Genitora

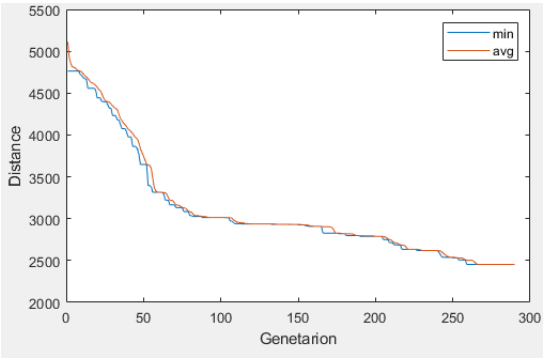
En este caso, vemos una clara diferencia entre los dos métodos, en el caso de la *edad* vemos como el modelo no es capaz ni de converger, ni de hallar una solución óptima, así que para nuestro problema es inviable utilizar ese método, elegiremos pues la *Selección genitora* para nuestro modelo final.

- **Mutaciones**

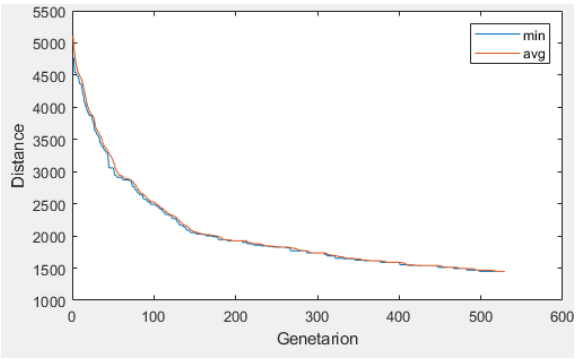
Para este último experimento, variaremos entre los dos tipos de mutaciones, por *Intercambio* y *Sacudida*, y a su vez variaremos la probabilidad de mutaciones. Compararemos a su vez estos dos métodos con un modelo sin mutaciones.

1. Por Intercambio

Probabilidad De Mutación	***Coste***	***Tiempo [s]***	***Generaciones***
0.2	2454	0.14	290
0.4	1997	0.08	233
0.5	1521	0.14	388
0.6	1450	0.19	529
0.8	1610	0.14	402
1	1598	0.11	285



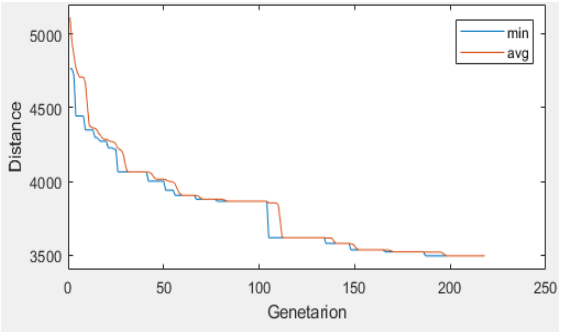
Gráfica probabilidad 0.2



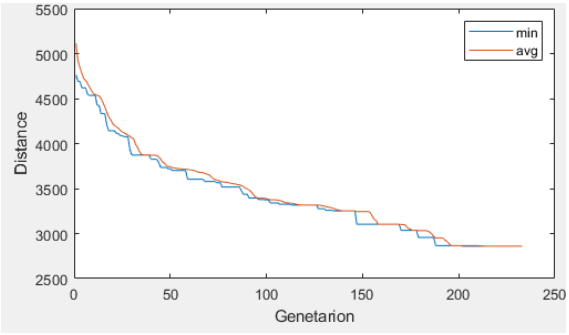
Gráfica probabilidad 0.6

2. Por Sacudida

Probabilidad De Mutación	***Coste***	***Tiempo [s]***	***Generaciones***
0.2	3494	0.08	218
0.4	3510	0.06	172
0.5	3082	0.08	213
0.6	2860	0.09	233
0.8	2620	0.08	201
1	2978	0.07	175



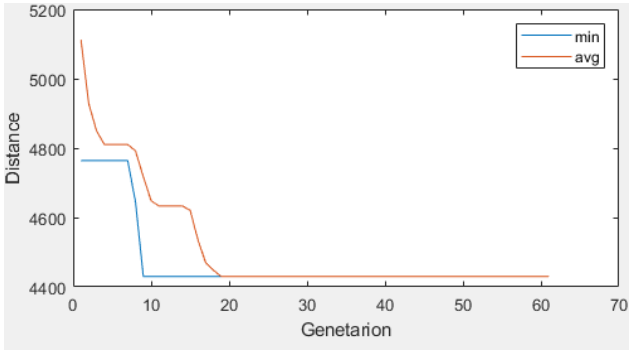
Gráfica probabilidad 0.2



Gráfica probabilidad 0.6

3. Sin mutaciones

Probabilidad De Mutación	***Coste***	***Tiempo [s]***	***Generaciones***
0	4430	0.02	61

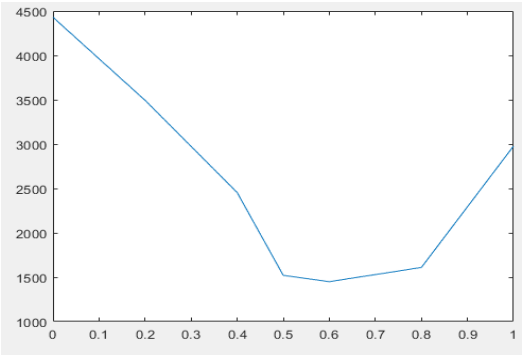


Gráfica Sin Mutaciones

De este experimento podemos sacar varias conclusiones. En primer lugar, vemos el gran **peso que tienen las mutaciones** en nuestro problema, algo a tener en cuenta y por el cual **descartar la ausencia de mutaciones**.

Por otra parte, podemos ver como los valores para la probabilidad de mutar más próximos a los extremos (0-0.3 y 0.7-1) tienen los peores resultados de todos, **centrándose las mejores probabilidades entorno a la mitad**.

Se podría ver de la siguiente manera:



Por último, cabe destacar que el método con los mejores resultados es la mutación por *Intercambio*. La mutación por *Sacudida* es demasiado agresiva para nuestro problema y tiende a converger muy rápidamente sin llegar a soluciones óptimas.

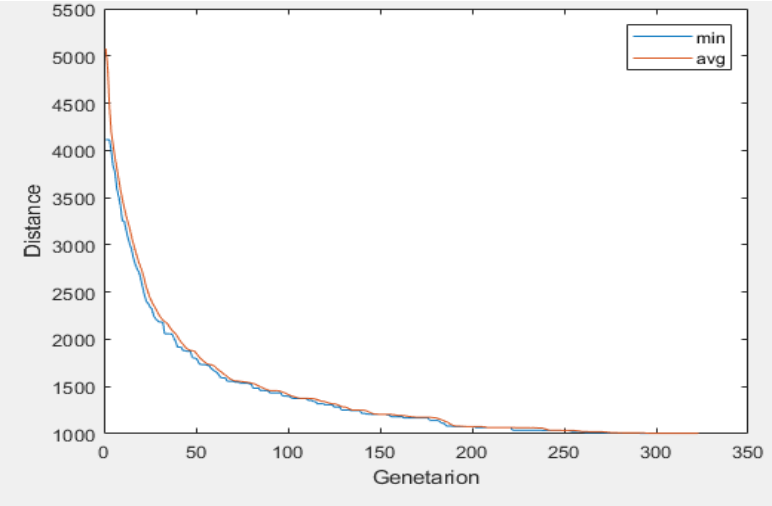
4 Modelo final:

Tras haber realizado los distintos experimentos, a base de las conclusiones obtenidas, crearemos el modelo final con los parámetros, que, a priori, parecen los más adecuados para nuestro problema, estos parámetros son los siguientes:

Individuos	Prob. Mutación	Select. Progenitores	Cruzamiento	Select. Supervivientes
260	0.6	Ruleta	PMX	Genitor

Mutación	NO. Padres selección	EarlyStopping Patience
Intercambio	70%	30 Generaciones

Con esta configuración conseguimos un **coste de 856** en **329 generaciones** tardando **1.58 segundos**, una mejora significativamente grande frente al coste mínimo de **1450** que conseguimos al principio.



Gráfica final

5 Conclusiones generales:

Con estos experimentos hemos visto la gran utilidad de los algoritmos genéticos para problemas de optimización, ya sean de menor o mayor escala, con los parámetros adecuados. Haciendo un análisis en profundidad de cuáles pueden ser los mejores métodos y parámetros podemos llegar a soluciones que funcionen incluso mejor que las que, a priori, parecen ser las mejores que podemos alcanzar. Sumado a lo anterior, hemos visto que, tengamos soluciones que converjan antes o después, estos algoritmos funcionan a una gran velocidad para nuestro problema, especialmente seleccionando los parámetros adecuados.

Hemos comprobado también que, para los algoritmos genéticos, es imprescindible, no solo realizar mutaciones, sino que tiene que haber un buen balance entre ellas, ni ningún cromosoma debe no mutar ni todos los cromosomas tienen que hacerlo. Llevando a ambos casos a una convergencia prematura que no llega a ninguna solución óptima. Algo parecido sucede haciendo que sobrevivan los individuos a la siguiente generación basándonos exclusivamente en la edad que tienen. En este caso, como en el anterior, no se llega a una solución, pero a demás no llega a converger en ningún momento.

Por otro lado, hemos visto también que, a mayor número de individuos tengamos interviniendo en cada generación, mejores resultados obtendremos, pero llega un punto que por mucho que incrementemos ese valor, no seguirá mejorando el resultado.

Por el resto de los métodos estudiados, como los de la *selección de progenitores* o el *cruzamiento*, en general, aun habiendo métodos que funcionen un poco mejor que otros, no difieren demasiado entre sus resultados, en especial entre el método del *torneo*, *ruleta* y *rango* y entre los métodos de *PMX* y *ciclos*.

En conclusión, se trata de algoritmos que merece la pena abordarlos con métodos y parámetros que vayan convergiendo más lentamente, ya que por su funcionamiento no se puede esperar una solución óptima inmediata.