Memoria Práctica 2

## Implementación

El proyecto está estructurado en 3 partes: el modelo, la vista y el controlador.

La vista, llama al controlador para pasarle los parámetros del algoritmo y el controlador crea

un objeto de tipo algoritmo genético, lo ejecuta y le devuelve los resultados a la vista.

* 1. Estructura de paquetes

En los paquetes del controlador y la vista están respectivamente la clase controlador y la clase interfaz.

En el paquete útil están los tipos enumerados, la clase correspondiente a los datos que le pasamos a la interfaz para que forme la gráfica y una clase Funciones que tiene todas las funciones de la f1 a la f5, estas funciones se llaman desde el modelo para calcular la aptitud de los individuos.

En el paquete modelo están todos los tipos de cromosoma, problema, selección y la clase algoritmo genético.

* 1. Estructura de clases

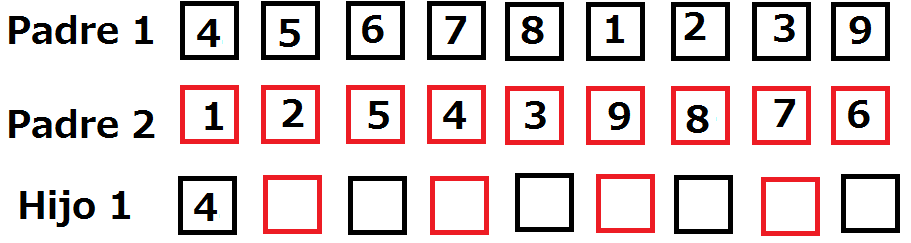
La estructura de clases está compuesta por unas clases abstractas para cromosoma, problema y selección, de estas 3 clases abstractas salen las clases para cada función en concreto y en el caso de selección para cada tipo de selección en concreto.

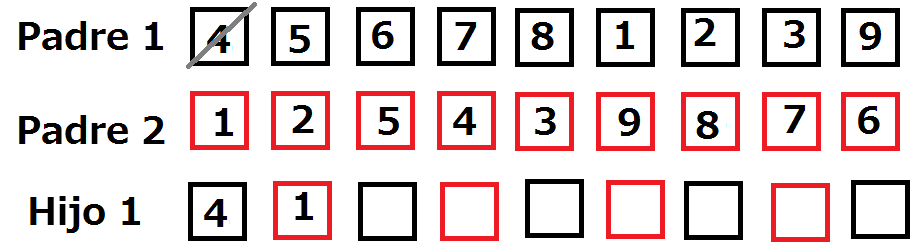
Para que el algoritmo genético pueda ser general y utilizar los métodos de las diferentes subclases se utiliza la clase Factoría que tiene 3 métodos, uno para devolver un objeto del cromosoma correspondiente a la función con la que se está trabajando, otra para devolver el problema y la última para devolver la selección.

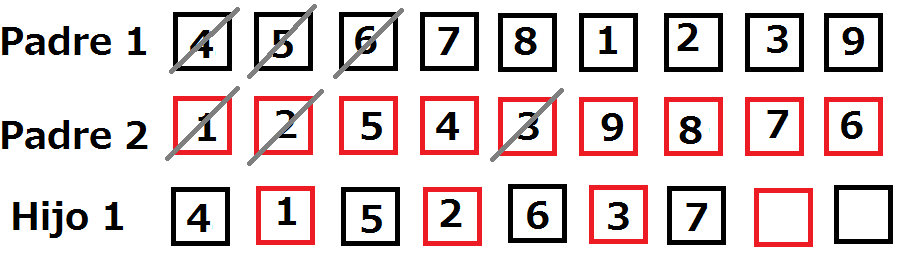
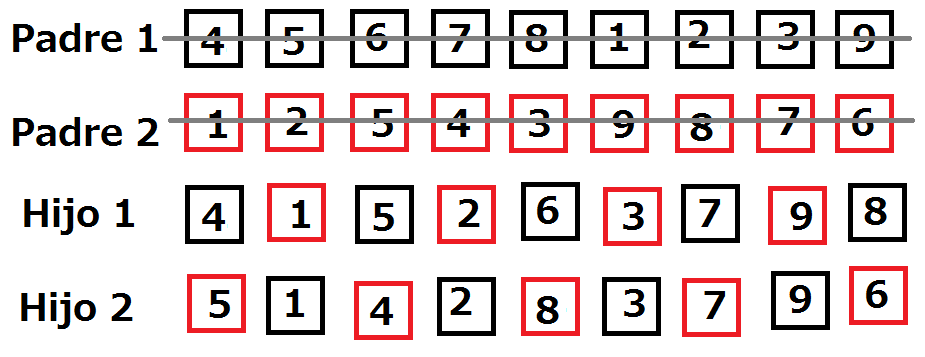
Con esta estructuración de clases conseguimos que el algoritmo genético sea general y ejecute de manera transparente las funciones correspondientes a cada tipo de función de manera transparente.

* 1. Algoritmos
  2. Cruce Propio

El cruce propio que hemos implementado consiste en intercalar (en orden) las ciudades de los cromosomas padre dentro del cromosoma hijo.

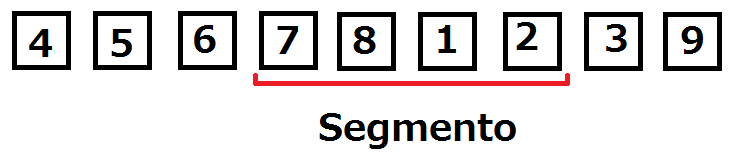
* El primer cromosoma hijo tendrá como primer elemento la primera ciudad del primer padre, y se elimina esa ciudad del padre.
* El siguiente elemento debe provenir del segundo padre, y tras este se volverá a introducir el siguiente elemento del primer padre.



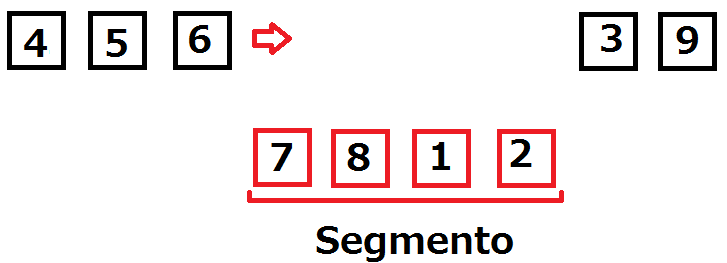
* En caso de que el elemento de un padre ya se encuentre dentro del cromosoma hijo, se saltara una posición en el cromosoma padre del que se quiere obtener el elemento y se prueba con el siguiente, y seguirá así hasta encontrar un elemento que no genere conflicto al ser introducido.
* Cada vez que se introduce una ciudad en el hijo, esta se elimina del padre del que proviene.
* Se repite el proceso hasta completar el cromosoma hijo.
* Para el segundo hijo, se empieza rellenando con las ciudades que sobren el segundo padre.
  1. Mutación Propia

Esta mutación consiste en seleccionar las ciudades ubicadas entre 2 puntos escogidos al azar, eliminarlas del original, invertir el orden de esta selección, y luego recolocar los elementos al principio del cromosoma.

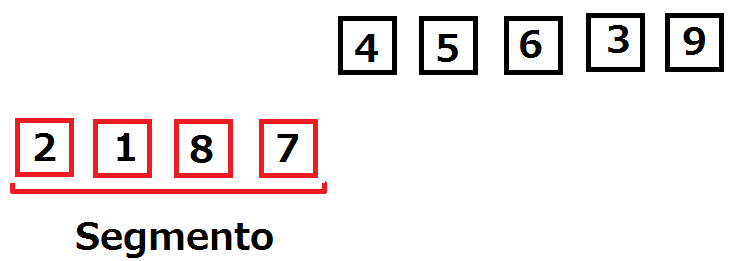
* Se deben escoger 2 puntos aleatorios del cromosoma. Estas ciudades que se encuentren entre estos 2 puntos formaran parte de nuestro segmento.



* Después de copiar las ciudades escogidas, estas deben ser eliminadas del cromosoma original.



* Lo siguiente que hay que hacer es invertir el segmento copiado



* Por último, reintroducimos las ciudades de nuestro segmento al inicio del cromosoma. Este cromosoma resultante será nuestro cromosoma mutado.

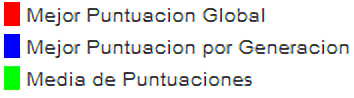


1. Gráficas de Evolución de las Distintas Mutaciones

En esta sección nos dedicaremos a realizar simulaciones de nuestro algoritmo evolutivo con distintos datos de entrada (ajuste.dat, datos12.dat, datos15.dat y datos30.dat) para conocer cuáles de los nuevos tipos de mutación introducidos genera los mejores resultados.

Todas las simulaciones de esta sección, excepto las que se indiquen, han sido generadas con los siguientes parámetros:

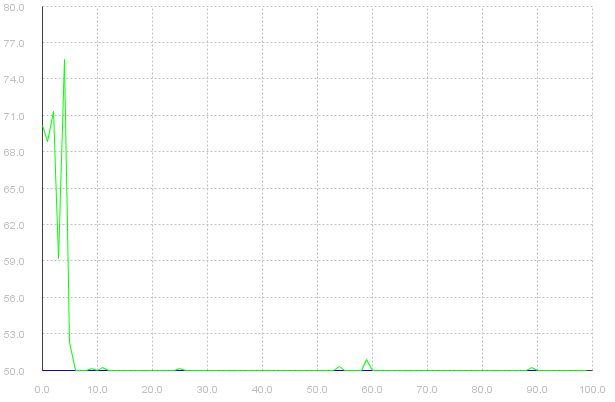
|  |  |
| --- | --- |
| **Semilla** | 2 |
| **Población Inicial** | 100 |
| **Nº. de Iteraciones** | 100 |
| **% de Cruce** | 60 |
| **% de Mutación** | 5 |
| **Precisión** | 0.001 |
| **Tipo de Selección** | Ruleta |
| **Elitismo** | Activado |
| **Tipo de Cruce** | Codificación Ordinal |
| **Inversión Especial** | Activado |

****Los colores en las gráficas representan los siguiente:

Simulación con el archivo ajuste.dat

Debido a que la entrada de prueba es muy pequeña, todas las simulaciones (independientemente del tipo de mutación usado) convergen al coste optimo esperado en las primeras iteraciones, y presentan una gráfica similar a la siguiente.

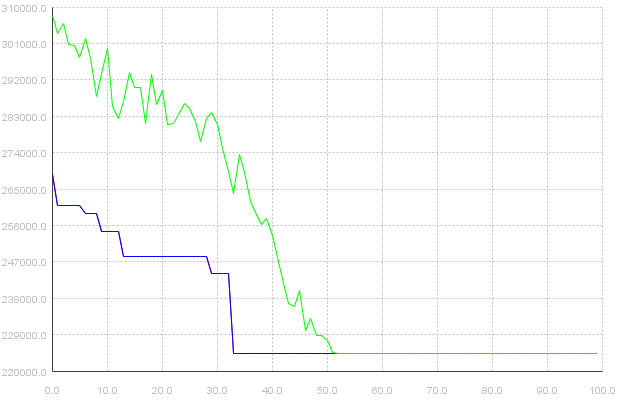
**Fenotipo(s):** [2, 3, 4, 0, 1] **--- Aptitud:** 50.0



Simulación con el archivo datos12.dat

Para este dato de entrada, los resultados de las simulaciones para distintos tipos de mutaciones empiezan a diferenciarse: Las mutaciones por inversión, inserción e intercambio arrojan resultados muy parecidos, con cromosomas que alcanzan una aptitud mínima que no baja de los 240.000, y la mutación propia solo baja hasta los 230.000.

Es la mutación heurística con la que obtenemos los mejores resultados, llegando a alcanzar el fenotipo con la aptitud que considerábamos óptima.



**Fenotipo(s):** [1, 3, 6, 11, 7, 2, 9, 0, 8, 5, 4, 10]

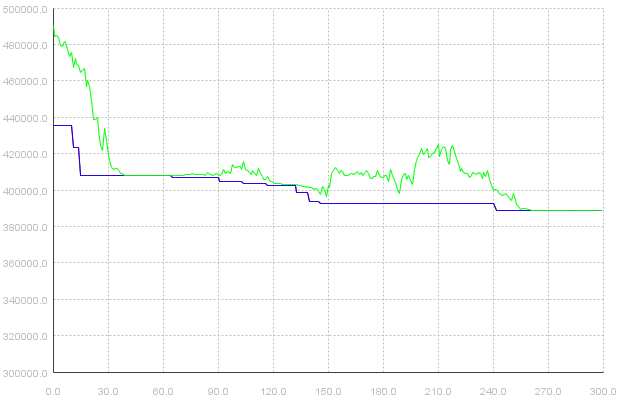
**Aptitud:** 224416.0

Simulación con el archivo datos15.dat

En estas simulaciones vemos que la diferencia entre los resultados de la mutación heurística y el resto de mutaciones se hace más grande (solo la heurística obtiene un mínimo que baja de los 400.000), pero también se vuelve más difícil conseguir cromosomas que se acerquen al coste óptimo esperado.

De hecho, todas las simulaciones que hicimos con la mutación heurística (más de 50), ninguna de ellas se acercaba lo suficiente al coste optimo, por lo que tuvimos que incrementar el número de iteraciones.

La siguiente grafica corresponde a una simulación en la que se han realizado 300 iteraciones.



**Fenotipo(s):** [10, 2, 11, 0, 14, 4, 1, 8, 7, 5, 3, 13, 9, 6, 12]

**Aptitud:** 388988.0

Simulación con el archivo datos30.dat

De todas las mutaciones, la heurística sigue siendo la que consigue los mínimos que más se acercan al coste optimo, mientras que todas las demás no son capaces de obtener aptitudes que bajen de 1.980.000.

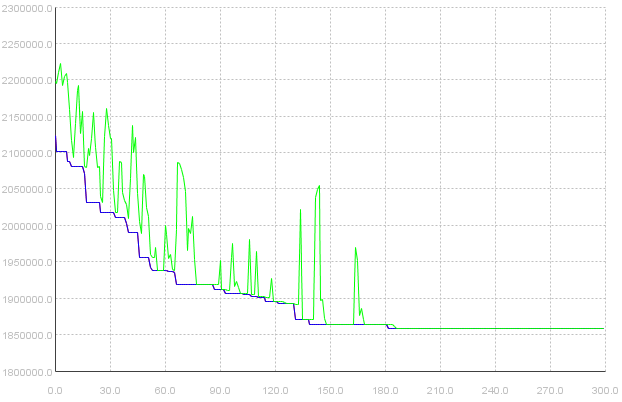
Pero al igual que en los casos anteriores, no basta con 100 iteraciones para acercarnos al coste óptimo.

La siguiente grafica corresponde a una simulación en la que se han realizado 300 iteraciones.

**Fenotipo(s):** [9, 24, 23, 15, 8, 7, 4, 0, 25, 14, 10, 13, 20, 12, 27,

19, 3, 17, 2, 22, 29, 28, 5, 11, 21, 6, 16, 18, 1, 26]

**Aptitud:** 1857696.0



También probamos a aumentar aún más el número de iteraciones (a 500) y aumentar la probabilidad de las mutaciones (de 5% a 20%), pero el mejor resultado que obtuvimos de estas simulaciones no era muy diferente del de 300 iteraciones, y tampoco se acercaba lo suficiente al coste optimo:

Resultado con 500 iteraciones y 20% probabilidad de Mutación:

**Fenotipo(s):** [0, 17, 25, 9, 6, 19, 11, 1, 20, 16, 3, 26, 27, 14, 22,

21, 15, 7, 2, 10, 18, 23, 29, 8, 4, 5, 12, 24, 13, 28]

**Aptitud:** 1855652.0

Tablas de Ejecuciones:

datos12.dat

Todas las simulaciones de esta sección han sido generadas con los siguientes parámetros:

|  |  |
| --- | --- |
| **Población Inicial** | 100 |
| **Nº. de Iteraciones** | 100 |
| **% de Cruce** | 60 |
| **% de Mutación** | 5 |
| **Precisión** | 0.001 |
| **Tipo de Selección** | Ruleta |
| **Elitismo** | Activado |
| **Tipo de Cruce** | Codificación Ordinal |
| **Inversión Especial** | Activado |

|  |  |  |
| --- | --- | --- |
| **Semilla** | **Fenotipo** | **Aptitud** |
| 265 | [6, 9, 11, 0, 4, 2, 3, 5, 1, 7, 8, 10] | 243618 |
| 10 | [6, 9, 4, 0, 11, 10, 3, 2, 5, 8, 7, 1] | 245068 |
| 46 | [5, 11, 9, 0, 8, 3, 10, 4, 1, 7, 2, 6] | 239670 |
| 56 | [6, 7, 4, 0, 10, 5, 9, 8, 3, 2, 11, 1] | 241360 |
| 68 | [11, 4, 6, 7, 8, 2, 5, 0, 10, 3, 9, 1] | 241056 |
| 55 | [6, 9, 4, 0, 11, 10, 3, 2, 5, 8, 7, 1] | 245068 |
| 40 | [2, 6, 4, 7, 3, 11, 5, 0, 10, 8, 9, 1] | 241722 |
| 44 | [11, 4, 6, 7, 8, 2, 5, 0, 10, 3, 9, 1] | 241056 |
| 66 | [10, 11, 3, 0, 1, 9, 4, 8, 7, 5, 2, 6] | 230704 |
| 321 | [4, 11, 9, 0, 1, 10, 3, 5, 8, 7, 2, 6] | 233040 |
| 666 | [7, 6, 10, 3, 4, 11, 5, 0, 2, 9, 8, 1] | 249738 |
| 600 | [5, 11, 7, 0, 6, 9, 4, 8, 3, 10, 2, 1] | 239830 |
| 354 | [11, 4, 6, 7, 8, 2, 5, 0, 10, 3, 9, 1] | 241056 |
| 565 | [11, 4, 9, 0, 6, 8, 7, 2, 3, 10, 5, 1] | 241170 |
| 355 | [3, 4, 7, 0, 10, 9, 5, 8, 6, 11, 2, 1] | 237488 |
| 345 | [3, 6, 2, 7, 10, 11, 5, 0, 4, 8, 9, 1] | 250132 |
| 365 | [9, 3, 6, 11, 5, 2, 8, 0, 1, 7, 4, 10] | 236768 |
| 532 | [7, 9, 5, 3, 10, 4, 1, 11, 6, 0, 8, 2] | 240400 |
| 362 | [5, 11, 7, 0, 6, 9, 4, 8, 3, 10, 2, 1] | 239830 |
| 323 | [7, 3, 8, 11, 1, 2, 9, 0, 6, 4, 5, 10] | 229092 |
|  |  |  |

datos15.dat

|  |  |  |
| --- | --- | --- |
| **Semilla** | **Fenotipo** | **Aptitud** |
| 12 | [6, 0, 8, 11, 12, 2, 7, 14, 4, 10, 1, 5, 13, 9, 3] | 403312 |
| 23 | [7, 13, 1, 10, 5, 6, 4, 8, 14, 2, 11, 9, 3, 12, 0] | 405956 |
| 34 | [11, 6, 1, 14, 7, 9, 5, 0, 4, 12, 13, 8, 3, 2, 10] | 404130 |
| 45 | [4, 12, 5, 1, 9, 0, 3, 14, 13, 8, 7, 2, 6, 10, 11] | 407120 |
| 56 | [0, 1, 4, 10, 8, 2, 6, 14, 13, 5, 12, 7, 9, 11, 3] | 395696 |
| 67 | [6, 1, 12, 9, 3, 0, 10, 2, 5, 13, 8, 7, 11, 4, 14] | 407396 |
| 78 | [0, 1, 8, 10, 4, 3, 7, 14, 13, 5, 12, 9, 6, 11, 2] | 391522 |
| 89 | [10, 1, 0, 3, 4, 7, 8, 2, 5, 13, 12, 9, 6, 11, 14] | 393052 |
| 100 | [0, 9, 8, 10, 2, 4, 6, 1, 5, 14, 3, 7, 11, 12, 13] | 392754 |
| 111 | [10, 1, 12, 7, 4, 2, 6, 14, 13, 5, 8, 9, 0, 11, 3] | 392318 |
| 122 | [10, 2, 14, 3, 9, 7, 6, 1, 13, 5, 12, 8, 4, 11, 0] | 398702 |
| 133 | [9, 0, 4, 11, 12, 3, 6, 7, 1, 13, 14, 5, 8, 10, 2] | 397156 |
| 144 | [6, 3, 11, 4, 7, 0, 1, 9, 2, 5, 13, 14, 8, 10, 12] | 397716 |
| 155 | [7, 9, 8, 11, 12, 2, 6, 1, 0, 4, 3, 10, 5, 14, 13] | 396216 |
| 166 | [6, 1, 14, 7, 4, 3, 10, 8, 13, 5, 2, 9, 0, 11, 12] | 393820 |
| 177 | [14, 10, 11, 5, 0, 7, 2, 4, 1, 12, 13, 8, 6, 3, 9] | 405160 |
| 188 | [0, 1, 13, 9, 4, 10, 14, 3, 5, 7, 2, 6, 11, 12, 8] | 401838 |
| 199 | [1, 5, 12, 0, 13, 9, 8, 2, 7, 3, 10, 6, 14, 4, 11] | 404018 |
| 210 | [2, 0, 11, 6, 1, 14, 13, 9, 3, 8, 7, 5, 4, 10, 12] | 406562 |
| 221 | [10, 1, 7, 14, 12, 11, 3, 2, 5, 4, 8, 6, 0, 13, 9] | 400238 |
|  |  |  |

datos30.dat

|  |  |  |
| --- | --- | --- |
| **Semilla** | **Fenotipo** | **Aptitud** |
| 1 | [3, 26, 19, 22, 5, 23, 8, 18, 9, 17, 29, 0, 13, 25, 10, 7, 1, 15, 28, 11, 21, 24, 16, 12, 6, 14, 27, 20, 2, 4] | 1870246 |
| 2 | [29, 12, 16, 26, 11, 2, 24, 13, 27, 0, 9, 19, 5, 8, 14, 10, 4, 3, 25, 15, 20, 6, 21, 1, 28, 7, 18, 22, 17, 23] | 1932106 |
| 3 | [11, 4, 7, 20, 6, 10, 22, 17, 1, 0, 21, 28, 5, 18, 27, 26, 3, 23, 29, 16, 19, 9, 25, 14, 8, 15, 13, 24, 12, 2] | 1925994 |
| 4 | [26, 22, 23, 11, 27, 28, 8, 12, 14, 7, 16, 17, 9, 1, 6, 15, 21, 25, 20, 10, 19, 13, 0, 2, 4, 18, 29, 3, 24, 5] | 1910092 |
| 5 | [9, 16, 7, 15, 11, 8, 14, 24, 29, 6, 17, 10, 0, 25, 2, 27, 28, 12, 20, 1, 5, 26, 3, 19, 4, 18, 23, 21, 13, 22] | 1911724 |
| 6 | [22, 24, 1, 29, 8, 2, 13, 10, 27, 19, 15, 4, 12, 7, 5, 21, 3, 9, 25, 20, 23, 6, 0, 14, 26, 11, 16, 17, 18, 28] | 1927076 |
| 7 | [28, 3, 25, 5, 26, 29, 23, 7, 2, 1, 10, 12, 27, 15, 16, 8, 21, 22, 0, 18, 17, 20, 9, 6, 11, 13, 19, 4, 14, 24] | 1915480 |
| 8 | [22, 11, 15, 6, 28, 13, 25, 23, 16, 1, 5, 26, 20, 19, 0, 12, 27, 4, 14, 18, 24, 9, 2, 7, 3, 17, 29, 10, 8, 21] | 1908258 |
| 9 | [24, 15, 16, 3, 29, 2, 11, 21, 20, 7, 19, 17, 18, 13, 8, 27, 9, 5, 25, 0, 23, 6, 26, 1, 4, 12, 10, 14, 22, 28] | 1910642 |
| 10 | [23, 10, 9, 15, 2, 1, 28, 24, 4, 25, 7, 19, 14, 3, 27, 29, 18, 13, 6, 17, 8, 5, 26, 11, 20, 21, 22, 12, 16, 0] | 1908656 |
| 11 | [22, 18, 16, 6, 26, 29, 9, 19, 27, 12, 15, 25, 17, 28, 10, 13, 5, 0, 2, 14, 11, 4, 20, 24, 8, 3, 1, 7, 23, 21] | 1931936 |
| 12 | [22, 23, 12, 15, 9, 8, 21, 24, 1, 10, 26, 6, 20, 17, 27, 7, 2, 16, 11, 25, 18, 3, 0, 13, 4, 29, 19, 14, 5, 28] | 1926434 |
| 13 | [24, 29, 16, 8, 2, 1, 20, 11, 5, 19, 27, 12, 0, 26, 15, 4, 14, 10, 7, 23, 22, 3, 6, 25, 9, 17, 13, 21, 18, 28] | 1896482 |
| 14 | [26, 11, 16, 6, 29, 23, 24, 9, 5, 17, 27, 19, 8, 20, 15, 4, 12, 10, 13, 22, 14, 3, 25, 7, 28, 2, 18, 1, 0, 21] | 1904946 |
| 15 | [11, 24, 17, 5, 3, 26, 7, 13, 1, 16, 6, 21, 12, 15, 29, 23, 22, 14, 28, 18, 0, 27, 2, 9, 4, 8, 10, 19, 20, 25] | 1906976 |
| 16 | [21, 6, 17, 0, 23, 9, 4, 20, 3, 19, 18, 24, 2, 11, 22, 27, 16, 29, 1, 13, 28, 10, 5, 25, 14, 7, 12, 8, 26, 15] | 1927134 |
| 17 | [13, 19, 1, 0, 14, 28, 17, 18, 9, 26, 6, 11, 25, 2, 22, 3, 24, 15, 4, 27, 16, 20, 7, 23, 21, 5, 12, 29, 10, 8] | 1915844 |
| 18 | [24, 29, 25, 5, 15, 22, 9, 26, 23, 17, 19, 14, 6, 7, 13, 2, 20, 16, 28, 1, 11, 3, 27, 12, 4, 0, 10, 8, 18, 21] | 1913238 |
| 19 | [27, 26, 19, 5, 28, 20, 25, 15, 14, 17, 11, 0, 13, 18, 3, 9, 10, 7, 24, 1, 22, 8, 6, 12, 29, 2, 23, 16, 21, 4] | 1913124 |
| 20 | [26, 5, 27, 3, 29, 11, 8, 21, 19, 1, 6, 7, 14, 2, 16, 4, 17, 9, 12, 18, 25, 24, 10, 0, 28, 13, 20, 22, 15, 23] | 1909608 |
|  |  |  |

1. Conclusiones

3.1 Tipos de Mutación

Tras haber realizado muchas simulaciones con los distintos tipos de mutación, podemos concluir con que la mejor de ellas es, sin duda, la mutación heurística, ya que no solo recombina distintas ciudades en distintas posiciones, sino que además solo se queda con aquella combinación que mejora(minimiza) la aptitud del cromosoma a mutar. La única desventaja de utilizar esta mutación es que el coste de la función es factorial [O(n!)], lo cual significa que cuanto mayor sea el número de ciudades a intercambiar entre sí, mayor será el tiempo que tardara la mutación en finalizar.

El resto de mutaciones no proporciona resultados tan buenos debido a que, independientemente de la aptitud del cromosoma mutado, este siempre sustituirá al original. Esto puede ocasionar que se pierdan muchos cromosomas con una buena aptitud tras el proceso de mutación.

El problema con estas mutaciones es que ninguna de ellas realiza modificaciones que garanticen que el cromosoma mutado sea mejor que el original. La mutación por intercambio es la que menos cambios realiza sobre el cromosoma, ya que solo intercambia la posición de valores del cromosoma. El resto puede alterar más posiciones del cromosoma, pero no garantiza que el cambio sea a mejor.

3.2 Tipos de Cruce

3.3 Parámetros