PRÁCTICA FINAL:
ALGORITMOS GENÉTICOS

Una pequeña memoria sobre cómo poder usar esta técnica para la resolución del famoso problema del viajante.

Alejandro Meza Tudela

Índice

1. Observaciones previas

El problema del viajante, o también conocida en habla inglesa *como The Travelling Salesman Problem,* es un problema de gran dimensionalidad. Concretamente, cuenta con un número posible de tours entre ciudades de (n-1)! /2, para n ciudades.

A lo largo de la historia, se ha tratado de resolver con diferentes técnicas. Entre estas, se pueden poner de ejemplo: redes neuronales o computación evolucional.

2. Algoritmos genéticos: un vistazo general

En 1975 se introdujeron los algoritmos genéticos. El espacio de búsqueda se traduce a un conjunto de individuos. Estos individuos, se denominan cromosomas. El objetivo de cualquier algoritmo genético es encontrar el mejor individuo. Se realizará esto mediante la función de fitness.

De manera general:

```
BEGIN ALGORITMO
 Iniciar de manera aleatoria población
 Mientras no condición de parada hacer
     BEGIN
        Selección de padres de la población
        Producir descendientes de los padres elegidos
        Mutar los individuos
        Añadir descendientes a población
        Reducir población
      END
  Devolver el mejor individuo
 END ALGORITMO
```

Veamos ahora, un *esquema en mayor profundidad de un algoritmo genético típico*:

1. Generar de manera aleatoria una población de n individuos. Cada uno, representa una posible solución.
2. Evaluar la adaptación de cada individuo mediante la función fitness.
3. Crear población mientras sea necesario, realizando lo siguiente:

   -Seleccionar cromosomas padres. A mayor adaptación, el individuo tendrá más

   probabilidades de ser progenitor.

   -Dada una probabilidad de cruce, cruzar los padres.

   -Dada una probabilidad de mutación, realizar ésta en caso afirmativo.

   -Colocar los cromosomas descendientes en la nueva población resultante.

4. Repetir esto hasta que se alcance condición de parada, y dar como mejor solución, la población actual.
5. Volver al paso 2.

Se ha de tener en cuenta, que cada iteración del algoritmo es referida como generación. Por otro lado, los operadores de *crossover* y mutación juegan papeles distintos, pero igual de relevantes.

Por un lado, los operadores de *crossover* o cruce -en español- han de mejorar la calidad de los individuos. Por lo que, tras cada iteración, la posibilidad de tener una solución cercana a la óptima crece.

Por otro lado, los operadores de mutación son necesarios para crear nuevos estados, y poder evitar que el algoritmo se estanque en mínimos locales.

3.El problema del viajero

El problema del viajero consiste, en que dada una lista de n ciudades, se ha de determinar la ruta más corta que visita cada una de ellas, y permite volver al punto de partida -existen otras variantes, donde solo se ha recorrer todas-. De manera más matemática, el problema del viajero se define:

“Dado un entero n >=3 y una matriz de dimensiones nxn C = (cij) , donde cij es un entero no negativo. ¿Qué permutación cíclica pi, de enteros de 1 a n, minimiza el sumatorio de i=1 a n de Cipi(i) ?”

El problema del viajero es relativamente antiguo, pues está documentado de aproximadamente el siglo XVIII por Euler (no por ese mismo nombre), que se interesó por resolver el problema del recorrido de los caballos. Dicho problema, consistía en determinar el camino óptimo que un caballo de ajedrez ha de seguir para recorrer todas las casillas, o *escaques,* del tablero.

El término ‘el problema del viajero’, fue usado por primera vez en 1932 en un libro alemán, escrito por un vendedor que recorría ciudades. Posteriormente, el problema fue introducida por la corporación RAND en 1948. La reputación de dicha empresa, y su importancia, hizo del problema del viajero, uno de los problemas más conocidos de la fecha.

Pero, aun así, surge la pregunta – que obviamente me surgió a mí también-, ¿por qué el problema del viajero es tan importante?

En primer lugar, es un problema fácil de describir, pero difícil de resolver. No hay ningún algoritmo que en tiempo polinómico sea capaz de encontrar solución. Esta peculiaridad, hace del problema del viajero un problema NP-completo característico.

En segundo lugar, es aplicable a una variedad de problemas de enrutamiento y de planificación. De los cuales, hay muchos.

En tercer lugar, el problema del viajero es un ejemplo perfecto para probar la eficiencia, o no, de métodos de optimización.

Y, por último, en una gran cantidad de problemas de campos como la inteligencia artificial donde se aplican técnicas heurísticas, interesa obtener la mejor permutación de los elementos. Como se puede intuir, diversos algoritmos que implementan heurística han sido probados con el problema del viajero.

3.1 Representación del problema

El problema del viajero o del viajante, admite diversas representaciones. Entre estas, se encuentra la representación binaria y la representación matricial. Ambas, usan alfabetos binarios para la representación del tour de ciudades.

Debido a que la representación puede no ser del todo intuitiva, y que los operadores de *crossover* y de mutación no constituyen operaciones cerradas; la elección para este proyecto ha sido la de números enteros, conocida en la literatura del problema, como la **representación del camino** (*path representation*) -o representación mediante permutación-.

En la **representación del camino**, las *n* ciudades que han de ser visitadas, se colocan en el orden de visita en una lista. De manera que, si se coloca una ciudad en la posición j, es la ciudad número j en ser visitada.

Actualmente es uno de los métodos más usados, y esto se debe a una razón fundamental: presenta una representación intuitiva, a la par que generalmente da buenos resultados.

A lo largo de los últimos años, muchos autores han abordado el problema de la misma forma. Por citar algunos:

Estrategia a seguir	Operadores	Autores
Representación del camino	Partially-mapped crossover	Goldberg and Lingle (1985)
	Order-crossover	Davis (1985)
	Order based crossover	Syswerda (1991)

	Position based crossover	Syswerda (1991)
	Heuristic crossover	Grefenstette (1987b)
	Edge recombination crossover	Whitley et al. (1989)
	Sorted match crossover	Brady (1985)
	Maximal preservative crossover	M¨uhlenbein et al. (1988)
	Voting recombination crossover	Mühlenbein (1989)
	Alternating-positions crossover	Larrañaga et al. (1996a)
	Displacement mutation	Michalewizc (1992)
	Exchange mutation	Banzhaf (1990)
	Insertion mutation	Fogel (1988)
	Simple inversion mutation	Holland (1975)
	Inversion mutation	Fogel (1990)
	Scramble mutation	Syswerda (1991)

3.2 Representación del camino, cruzamiento y mutaciones

Como se ha mencionado anteriormente, la representación del camino -o representación por permutación- es una de las más naturales. El tour es una lista de *n* ciudades. Por lo tanto, si se tiene el tour 3-1-5-6-7-8-2-4, simplemente es representado por:

(3 1 5 6 7 8 2 4)

Debido a esta representación, se puede entender que se está ante un problema que importa el orden, y, por tanto, se pueden aplicar las ideas de la codificación por permutación. Nos encontramos, ante una lista de operaciones de *crossover* y mutación particulares.

<u>Algunos tipos de cruzamiento</u>

-Cruzamiento parcialmente mapeado (PMX)

-Cruzamiento por ciclos (CX)

-Cruzamiento basado en orden (dos variantes) (OX1 / OX2)

-Cruzamiento por aristas (ER)

-Cruzamiento basado en posición (POS)

-Cruzamiento heurístico

-Cruzamiento multiparentales

Algunos tipos de mutaciones

-Mutación por inserción (ISM)

-Mutación por intercambio (EM)

-Mutación por sacudida (SM)

-Mutación por inversión (SIM)

Para el desarrollo del presente proyecto, **se ha elegido los siguientes cruzamientos**:

- **Cruzamiento parcialmente mapeado (PMX)**, pues la adyacencia es de gran importancia en el problema.

- Cruzamiento basado en orden (OX), pues en el problema del viajero, el orden en el que se visitan las ciudades es relevante.

Ambos son enfoques distintos, y es por ello, por lo que desde mi punto de vista creo que dotarán de riqueza a los resultados que se vayan a obtener. Como curiosidad, estos operadores de cruzamiento fueron los elegidos para esta misma situación, en la prestigiosa publicación *'The cronicity of self-injurious behaviour: a long-term-follow-up of a total population study'*.

Con respecto a las **mutaciones,** se ha implementado la mutación por inserción. Se ha decidido llevar esta acabo, pues influye en el orden de los elementos, y para nuestro caso en concreto, en el orden de las ciudades a visitar.

3.3 Selección de progenitores

Los métodos de selección de progenitores han sido la ruleta y el método del torneo con reemplazamiento.

Me he decantado por estos dos métodos, pues pese a que los dos favorecen a la selección como progenitor del mejor individuo, uno lo hace más que otro, y eso brinda diversidad a la hora de elaborar los experimentos.

3.4 Selección de función de fitness

La función de fitness utilizada ha sido parecida a la que se hace referencia en la publicación "Solving TSP problema with improved genetic algorithm", siendo esta, bastante intuitiva.

$$f(x) = \Sigma\, D\,(Ci, Ci+1) \rightarrow \text{límite inferior: } i=1, \text{ límite superior: } n-1$$

La función de fitness termina siendo la suma de distancias de las *n* ciudades adyacentes. El objetivo, será obtener la solución asociada al menor coste total.

En la implementación realizada en *Matlab*, el cálculo del fitness de los cromosomas sigue la forma de la siguiente operación: 1/f(x).

Penalizando de esta manera, a los individuos con mayor distancia total; y favoreciendo a los que tienen asociados una distancia menor.

3.5 Selección de supervivientes

El método de selección de supervivientes elegido, ha sido el de la selección elitista. La razón de dicha elección ha sido los buenos resultados que fueron obtenidos en la publicación "*Genetic algorithms for the travelling salesman problema: a review of representation and operators*", y naturalmente, la tendencia a quedar con la mejor población posible.

En este método, los nuevos individuos reemplazan a los peores individuos existentes, solo si su función de evaluación es mejor. Con el pasar de las iteraciones, se conserva siempre una población con el mismo número de cromosomas.

3.6 Resultados experimentales

Para las pruebas, se han considerado diverso número de poblaciones para obtener variedad en los resultados. De cara a tener un coste computacional moderado, se han realizado las simulaciones con un número de ciudades igual a diez. Se ha usado el fitness como forma de evaluación en cada experimento.

3.6.1 Mutación y selección de progenitores: métodos obtenidos con pmx

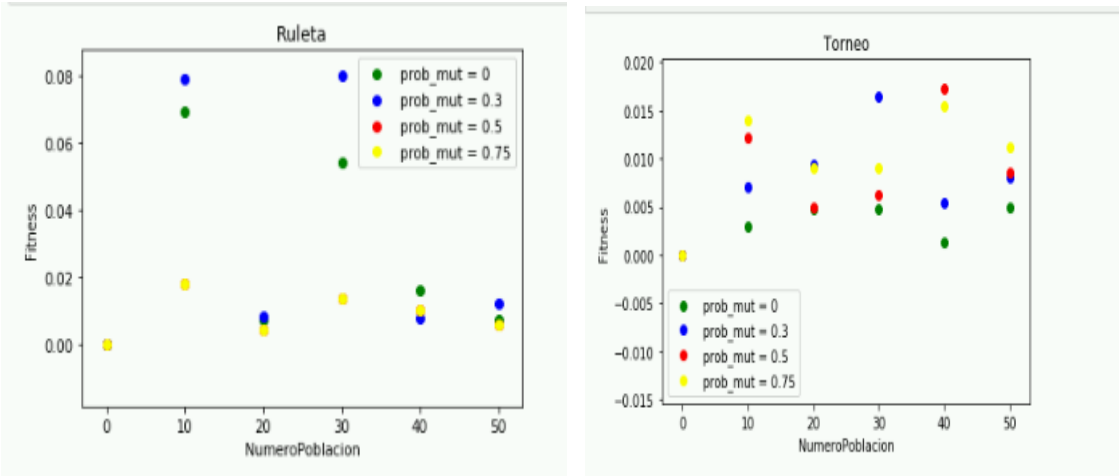
El primer estudio realizado, ha sido la comparación métodos de selección de progenitores. Por un lado, se ha experimentado con el método de la ruleta, y por el otro, el método del torneo. El cruzamiento realizado ha sido pmx. Se ha elegido sin un propósito específico, pues solo se ha buscado comparar los dos métodos ante las mismas condiciones.

Ruleta

Prob.Mutación	0 indiv.	10 indiv.	20 indiv.	30 indiv.	40 indiv.	50 indiv.
0	0	0.069	0.074	0.054	0.0159	0.0072
----------	----------	----------	----------	----------	----------	----------
0.3	0	0.079	0.081	0.08	0.075	0.0123
----------	----------	----------	----------	----------	----------	----------
0.5	0	0.0179	0.0435	0.0133	0.0101	0.0059
----------	0	----------	----------	----------	----------	----------
0.75	0	0.068	0.0106	0.009	0.0101	0.0120

Torneo

Prob.Mutación	0 indiv.	10 indiv.	20 indiv.	30 indiv.	40 indiv.	50 indiv.
0	0	0. 003	0. 0048	0. 0047	0. 0057	0. 0062
----------	----------	----------	----------	----------	----------	----------
0.3	0	0. 0071	0. 0094	0. 0164	0. 0055	0. 0081
----------	----------	----------	----------	----------	----------	----------
0.5	0	0. 0122	0. 0049	0. 0063	0. 0172	0. 0086
----------	0	----------	----------	----------	----------	----------
0.75	0	0. 0139	0. 0090	0. 0091	0. 0154	0. 0112



Gráficas realizadas en Python

Observaciones a partir de los experimentos realizados: Tras realizar la comparación de ambos métodos, se puede apreciar que el hecho de haber aplicado el método de la ruleta ha influido en un mejor resultado. Recordando que para la elaboración del algoritmo genético se ha utilizado el torneo con reemplazamiento, esto puede ser natural; pues siempre ganarán los mejores individuos, y dará lugar a una convergencia más rápida, pero quizás no llegará a un mejor resultado. Eso sí, si el número de individuos fuese mucho mayor, el método del torneo quizás alcance un mejor resultado, puesto que, en el método de la ruleta, hasta los mejores individuos no poseerían una probabilidad demasiada alta.

Por otro lado, se puede ver que aplicar una probabilidad de mutación del 30%, parece mejor que otras opciones -como lo han sido 0%, 50%, 75%-. Esto puede ser debido, a que es una probabilidad pequeña, pero suficiente para aportar variación al algoritmo, y llegar a soluciones más prometedoras. Con un porcentaje de mutación excesivo, el algoritmo se puede desviar fácilmente de la búsqueda de la solución dentro del espacio de soluciones.

3.6.2 Cruzamiento y selección de progenitores

El segundo experimento realizado, se centró en la comparativa de los métodos de cruzamiento. Como existen dos métodos de selección de progenitores, se tuvieron en cuenta las diferentes combinaciones que esto implica. El número de progenitores elegido fue de cuatro.

Basado en orden + ruleta + probabilidad_cruzamiento (0.8)

Prob.Mutación	**0 indiv.**	**10 indiv.**	**20 indiv.**	**30 indiv.**	**40 indiv.**	**50 indiv.**
0	0	0. 092	0. 0102	0. 0149	0. 0058	0. 0072
----------	----------	----------	----------	----------	----------	----------
0.3	0	0. 0120	0. 0119	0. 0141	0. 0058	0. 0103
----------	----------	----------	----------	----------	----------	----------
0.5	0	0. 0132	0. 0143	0. 0112	0. 0087	0. 0149
----------	0	----------	----------	----------	----------	----------

0.75	0	0. 0164	0. 0116	0. 0167	0. 0114	0. 0079

Basado en orden + torneo + probabilidad_cruzamiento (0.8)

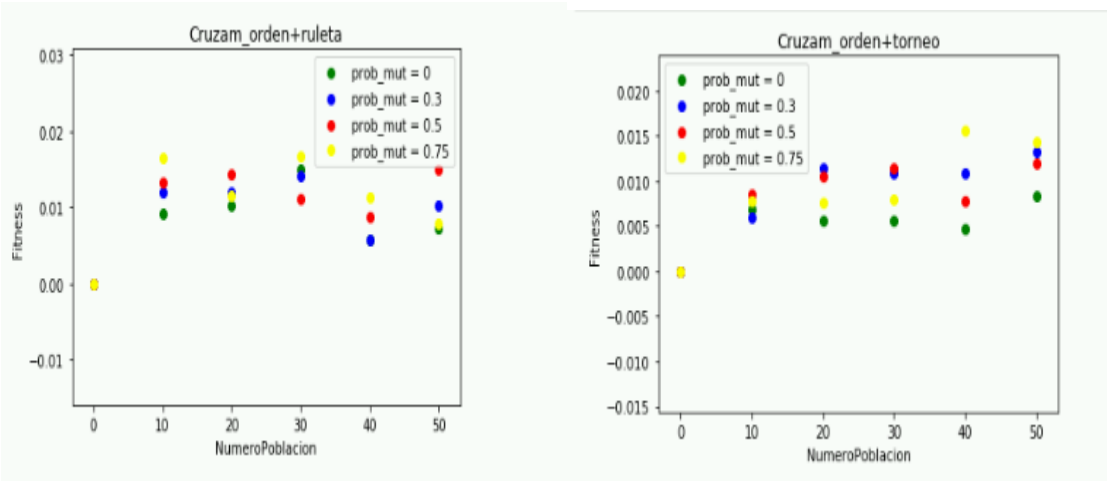
Prob.Mutación	0 indiv.	10 indiv.	20 indiv.	30 indiv.	40 indiv.	50 indiv.
0	0	0. 0068	0. 0057	0. 0057	0. 0047	0. 0083
----------	----------	----------	----------	----------	----------	----------
0.3	0	0. 0060	0. 0114	0. 0109	0. 0109	0. 0133
----------	----------	----------	----------	----------	----------	----------
0.5	0	0. 0086	0. 0105	0. 0115	0. 0078	0. 0120
----------	0	----------	----------	----------	----------	----------
0.75	0	0. 0078	0. 0077	0. 0080	0. 0156	0. 0143

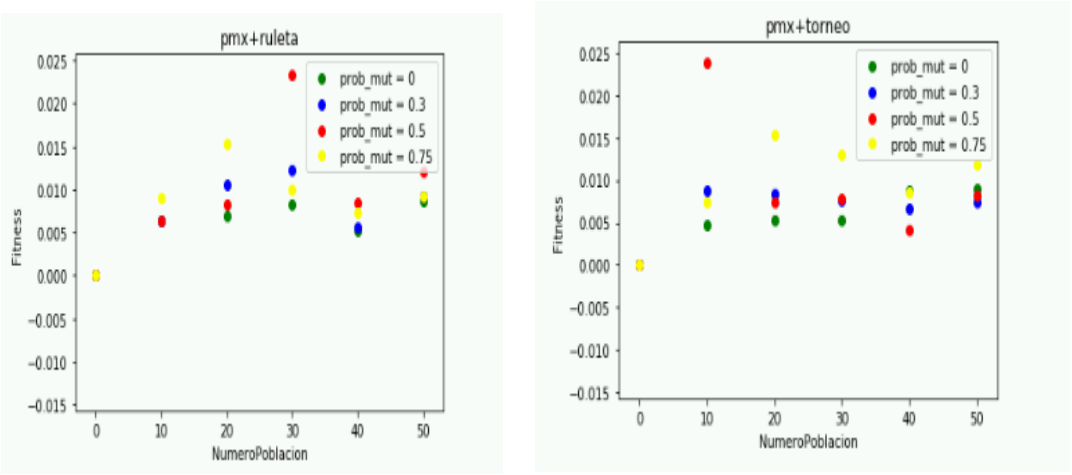
PMX + ruleta + probabilidad_cruzamiento (0.8)

Prob.Mutación	0 indiv.	10 indiv.	20 indiv.	30 indiv.	40 indiv.	50 indiv.
0	0	0. 0064	0. 0070	0. 0082	0. 0053	0. 0087
----------	----------	----------	----------	----------	----------	----------
0.3	0	0. 0064	0. 0105	0. 0123	0. 0057	0. 0093
----------	----------	----------	----------	----------	----------	----------
0.5	0	0. 0063	0. 0083	0. 0233	0. 0085	0. 0120
----------	0	----------	----------	----------	----------	----------
0.75	0	0. 0090	0. 0154	0. 0100	0. 0074	0. 0093

PMX + torneo + probabilidad_cruzamiento (0.8)

Prob.Mutación	0 indiv.	10 indiv.	20 indiv.	30 indiv.	40 indiv.	50 indiv.
0	0	0. 0047	0. 0052	0. 0053	0. 0087	0. 0090
----------	----------	----------	----------	----------	----------	----------
0.3	0	0. 0088	0. 0083	0. 0076	0. 0067	0. 0075
----------	----------	----------	----------	----------	----------	----------
0.5	0	0. 0238	0. 0075	0. 0079	0. 0042	0. 0082
----------	0	----------	----------	----------	----------	----------
0.75	0	0. 0075	0. 0154	0. 0130	0. 0085	0. 0119





Gráficas realizadas en Python

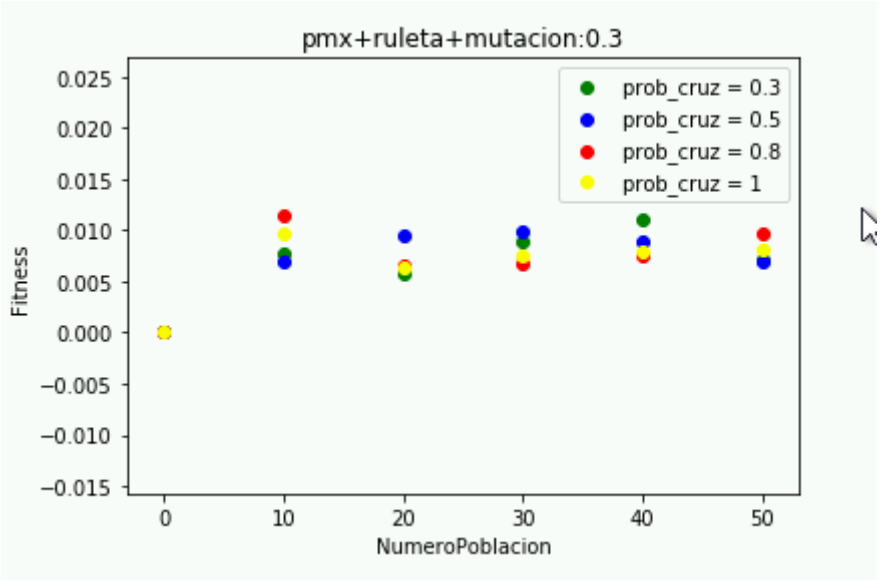
Observaciones a partir de los experimentos realizados: Después de la realización de la comparativa de los dos métodos de cruzamiento, se han obtenido mejores resultados con el método de cruzamiento parcialmente mapeado -pmx-. *¿Quizás esto puede poner en manifiesto que es más importante para el problema, la adyacencia que el orden?*

Además, de entre los dos experimentos con pmx, se han obtenido similares resultados, independientemente del método de selección de progenitores.

3.6.3 Probabilidad de cruzamiento

Tras los experimentos anteriores, se ha considerado oportuno comparar el impacto de la probabilidad de cruzamiento en el fitness. Para ello, se ha elegido una selección de progenitores mediante el método de la ruleta, y el cruzamiento parcialmente mapeado -pmx-; debido a sus buenos resultados previamente. La probabilidad de mutación elegida ha sido del 30%, de igual manera, por sus buenos resultados.

Prob.cruzamiento	0 indiv.	10 indiv.	20 indiv.	30 indiv.	40 indiv.	50 indiv.
0.3	0	0. 0077	0. 0057	0. 0088	0. 0111	0. 0072
----------	----------	----------	----------	----------	----------	----------
0.5	0	0. 0069	0. 0095	0. 0098	0. 0088	0. 0070
----------	----------	----------	----------	----------	----------	----------
0.8	0	0. 0115	0. 0066	0. 0068	0. 0075	0. 0096
----------	0	----------	----------	----------	----------	----------
1	0	0. 0096	0. 0063	0. 0076	0. 0079	0. 0082



Gráfica realizada en Python

Observaciones a partir de los experimentos realizados: Tras realizar un pequeño estudio sobre el impacto de la probabilidad de cruzamiento en el fitness total del algoritmo, se han obtenido mejores resultados con las probabilidades del 80% y del 50%. Esto pone de manifiesto que el cruzamiento es necesario, pero no realizarlo de manera esporádica, y tener hijos igual que los progenitores, puede ser favorable a la hora de buscar en el espacio de soluciones del problema.

3.6.4 Futuras experimentaciones

Pese a que se han realizado varios estudios, este trabajo deja aún muchos flecos por analizar. De cara a posteriores experimentos, se podría probar a usar un número de ciudades mayor, y ver en qué medida afecta esto a los resultados obtenidos y a la complejidad del problema.

Además, se podría probar a utilizar otros métodos de selección, cruzamiento y mutación; igual de prometedores que los que se han usado en este informe. E incluso, incorporar otras funciones de fitness probadas por la comunidad científica.

4.**Bibliografía**

- *"Solving TSP problema with improved genetic algorithm",* Chunhua Fua, Lijun Zhang, Xiaojing Wang, Liying Qiao

- *"The cronicity of self-injurious behaviour: a long-term-follow-up of a total population study", Lorne Taylor, Chris Oliver, Glynhis MU*

- *"Genetic algorithms for the travelling salesman problema: a review of representation and operators",* Pedro Larrañaga, Cindy Kuipers, R.H Murga, I.Inza, S.Dizdarevic