Algoritmos genético y enfriamiento simulado aplicados al problema del Viajante de comercio. Técnicas de Inteligencia Artificial (Aplicación Metaheurísticas)

Oliva Rodríguez, Alejandro

Índice

1. Introducción

El problema del viajante de comercio es uno de los problemas de optimización más estudiados. Consiste básicamente en encontrar la mínima distancia a recorrer visitando únicamente una vez cada ciudad comprendida en una lista de ciudades.

Para un problema con N ciudades existe N! posibles soluciones, sin embargo se puede reducir el número de soluciones a (N-1)! si no contamos la ciudad inicial. El viajante puede desplazarse libremente sin tener en cuenta la dirección a la que dirigirse, lo que reduce el número de soluciones a (N-1)!/2.[1]

En este trabajo se plantean dos alternativas para encontrar posibles soluciones al problema del viajante. En primer lugar, un algoritmo genético. Para la resolución se desarrollan diferentes variantes para un algoritmo genético, donde, la búsqueda y elección de parámetros para cada una de estas variantes es la clave para conseguir buenos resultados en un tiempo razonable acorde a la talla N de las ciudades del problema. En segundo lugar, se desarrolla un algoritmo de enfriamiento simulado donde se emplean algunas operaciones del algoritmo genético para la perturbación de soluciones. En el enfriamiento simulado, la temperatura inicial y final como el factor de enfriamiento serán los principales parámetros a configurar para la búsqueda de soluciones.

Para finalizar el trabajo, se presentará una comparativa y conclusión sobre los resultados obtenidos para cada uno de los algoritmos desarrollados para la resolución del problema.

2. Algoritmo Genético

Un algoritmo genético se puede describir como una serie de pasos a realizar para la consecución de una solución. Estos pasos se pueden enumerar generalmente como:

- Población inicial
- Aptitud *fitness(x)*
- Selección
- Cruce
- Mutación
- Reemplazo

El primer paso para la generación de la población se realiza únicamente al inicio del algoritmo. El resto de pasos se realizan en diversas iteraciones o más bien generaciones. El número de generaciones que realiza el algoritmo está sujeto a una función de parada que evaluará el comportamiento del algoritmo durante las generaciones y si este está mejorando o convergiendo a lo largo de estas.

Para la resolución del problema se desarrolla un algoritmo genético desde cero, donde se implementan cada una de las operaciones mencionados. Para las operaciones que se repiten en cada una de las generaciones(selección, cruce, mutación y reemplazo) se han implementado diferentes variantes de cada uno de ellos para poder elaborar una comparativa de resultados. El lenguaje utilizado para la implementación es Python.

2.1. Individuo

Antes de explicar cada una de las operaciones del algoritmos, es necesario explicar como se representa un individuo de la población para el problema planteado. El individuo

estará formado por un genotipo que tendrá tantos genes como ciudades tenga el problema. Se ha de recordar que no podrá haber mas de un alelo repetido. En la siguiente figura se puede ver como se representa un individuo para un ejemplo de 7 ciudades:

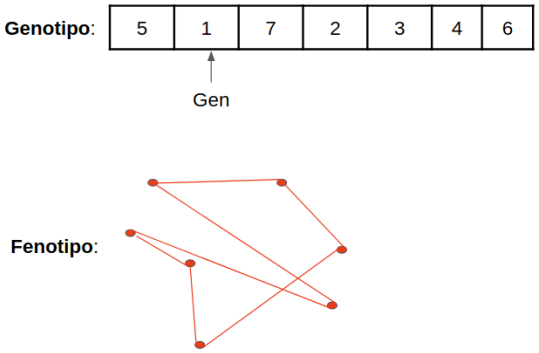

Figura 1. Representación de un individuo.

También se observa el fenotipo que se correspondería con la representación del individuo de forma externa.

2.2. Población inicial

En la población inicial está formada por *n* individuos aleatorios. Para la generación de cada individuo de la población se tiene en cuenta que cada una de las soluciones sea válida. Para que una solución sea válida se ha de respetar la restricción de no visitar una ciudad más de una vez.

Este primer paso tiene importancia en cuanto a parámetros del algoritmo. Dependiendo el número de individuos a crear, *n* puede agilizar la búsqueda o ralentizar este proceso. Si se genera una población con un valor de *n* elevado y el problema tiene una talla elevada de *N* ciudades, el tiempo de ejecución para cada generación puede ser elevado. Por esto motivo se le asigna importancia a la búsqueda de un valor óptimo al parámetro *n*, teniendo en cuenta el tamaño de N ciudades, el tiempo de ejecución y la mejora de las soluciones en cada generación.

2.3. Aptitud

La aptitud de cada uno de los individuos o también llamada función *fitness*, es la que indica la calidad de cada uno de estos. Para el cálculo de dicha función en el problema, se tiene en cuenta cual es la función objetivo de este, que se trata de minimizar el recorrido a realizar. Una buena forma de representar el *fitness* de cada uno de los individuos es realizar la inversa de la distancia del recorrido que ha realizado. Cuanto menor sea el coste del recorrido realizado más elevado será el valor de su *fitness* que se utilizará para determinar los cambios en la población en los siguientes puntos.

$$f(x) = \frac{1}{CosteRecorrido} \tag{1}$$

Donde:

$$\textit{CosteReccorrido} = \sum_{n}^{N-1} ruta(n, n+1) + (1, N-1) \tag{2}$$

2.4. Selección

En el proceso de selección se eligen a ciertos individuos (padres) que serán los encargados de reproducirse y generar nueva población a partir de su carga genética. El número de padres a elegir en el proceso de selección es un parámetro importante a tener en cuenta. En el algoritmo realizado, se ha implementado dos técnicas diferentes de selección que se describen en los siguientes subapartados. En ambas técnicas, se ha asegurado que uno de los padres seleccionados sea el mejor individuo de la población aunque esto no significa que necesariamente vaya a reproducirse.

2.4.1. Selección proporcional (Rueda de ruleta)

En la selección proporcional los individuos con un alto *fitness* serán aquellos que más probabilidad tenga de ser seleccionados. Para el problema se ha implementado la rueda de ruleta, donde a cada individuo se le asigna una probabilidad de ser elegido acorde a su *fitness*. Los mejores individuos son aquellos que más probabilidad tienen de ser elegidos, aunque esto no significa que otro padre peor no pueda ser seleccionado.

En las primeras pruebas realizadas con el algoritmo se observa que existe una baja 'Presión Selectiva' ya que el valor de *fitness* es similar en todos los individuos. Esto ofrece una evolución lenta o prácticamente nula. Para mitigar este efecto, se ha comprobado que seleccionar al mejor padre sin participar en el proceso(el resto se seleccionan con la ruleta) ofrece buenos resultados. Esto es una técnica donde podría hacer que la población siempre mantuviera la misma carga genética, lo que podría llevar al algoritmo a converger en un óptimo local. Sin embargo, como se ha mencionado anteriormente, es importante establecer el parámetro de número de padres a seleccionar y se ha de tener en cuenta de que no todos los padres seleccionados tienen la certeza de reproducirse.

2.4.2. Torneo

Está segunda técnica a diferencia de la primera, da más oportunidades a los peores individuos a que sean seleccionados como padres. En el algoritmo se ha establecido como parámetro el número de padres *P* que van a ser seleccionados. A partir de este parámetro se generarán un total de *P* grupos. En cada uno de los grupos se elige al individuos con mejor *fitness* y esté pasará a ser padre. El número de individuos es igual en cada uno de los grupos, por ejemplo, si la población es de 50 individuos y se quieren seleccionar 5 padres, entonces, se forman 5 grupos y cada uno de estos grupos estará formada por 10 individuos.

La composición de los grupos se realiza de forma aleatoria y no se rige por ningún parámetro. Esto puede ayudar a que padres con peor *fitness* sean seleccionados. También a diferencia de la técnica anterior donde se quiere asegurar la elección del mejor padre, en este siempre se asegura la elección de este ya que será el ganador del grupo donde se encuentre.

Está técnica es la que se ha utilizado generalmente para la realización de pruebas del algoritmo.

2.5. Cruce

La operación de cruce es la encargada de generar hijos a partir de los padres seleccionados en la operación de selección. Para este algoritmo implementados se habla de una talla fija de individuos en la generación de la población, es decir, que los nuevos hijos no ampliarán el número de individuos de la población. Para realizar esto, en la generación de la población, se establecen unos parámetros que son el número de hijos a generar h y que a su vez serán las posiciones que quedarán vacías en la población. Estas posiciones vacías serán reemplazadas por los nuevos hijos. En cada generación del algoritmo estas posiciones vacías se generan en la operación de reemplazo que se explicara en los próximos apartados.

Para este algoritmo se ha querido implementar diferentes operaciones de cruce, para comprobar cómo se comporta el algoritmo a lo largo de las generaciones y poder realizar una comparativa.

2.5.1. Cruce ordenado

En el algoritmo implementado el cruce ordenado se realiza a partir de dos padres. Los dos padres se seleccionan de forma aleatoria entre todos los padres seleccionados en la operación de selección.

Está operación esta compuesta por diferentes parámetros a configurar, donde, el primer parámetro es el tamaño de corte de genes del primer padre. El segundo parámetro es por donde se realiza el corte, el cual se obtiene de forma aleatoria. Para entender cómo se realiza la operación se van a ilustrar diferentes imágenes del proceso. En la Figura 2 se puede observar el primer paso realizado a partir de los parámetros mencionados.

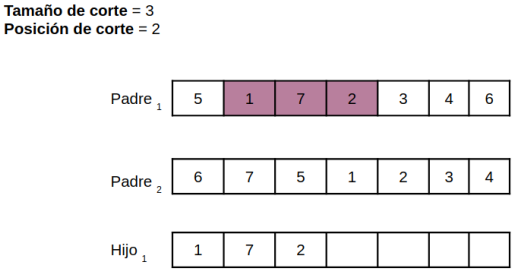

Figura 2. Cruce ordenado

El siguiente paso a realizar en la operación es completar al hijo con los genes del segundo padre. Para ello se sitúa en el índice siguiente a la última posición de corte realizada en el primer padre. Seguidamente se recorren cada una de las posiciones del segundo padre hasta completar al hijo. Es importante respetar la restricción de no repetir elementos en el hijo. En la Figura 3 se observa cómo se completa un hijo.

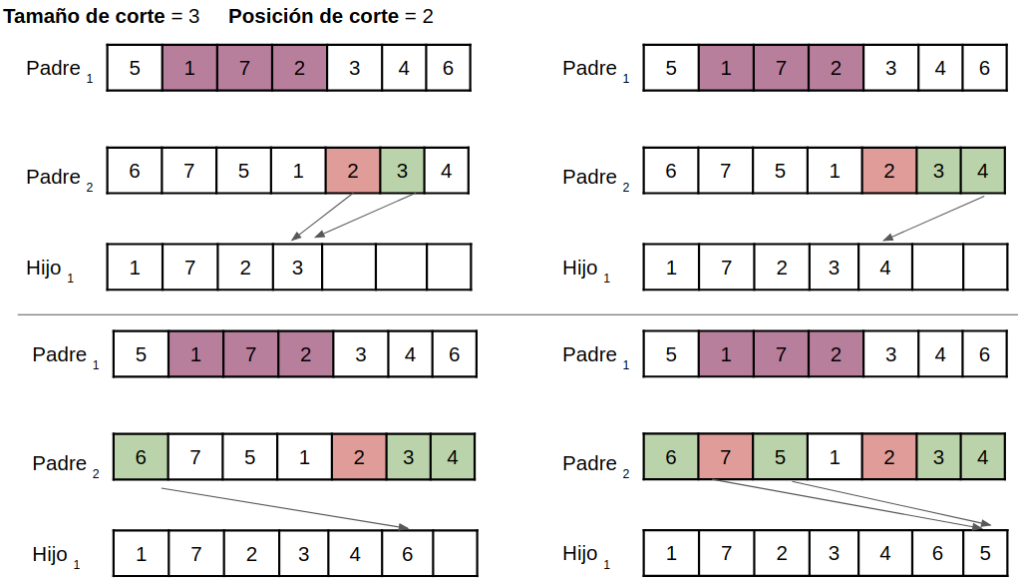

Figura 3. Cruce ordenado

El proceso mostrado para la generación de un hijo se repite tantas veces como posiciones vacías se hayan establecido para la generación de nuevos hijos.

2.5.2. Cruce por parejas

Esta operación de cruce, como en la primera, se seleccionan dos padres de forma aleatoria que serán los encargados de reproducirse para la generación de un nuevo hijo. La operación consiste en seleccionar bloques de genes de cada uno de los padres de forma consecutiva e insertarlos en el nuevo hijo. Como parámetro se ha de establecer el tamaño de bloque. También es importante respetar la restricción de elementos repetidos. En la figura 4 se observan los primeros pasos:

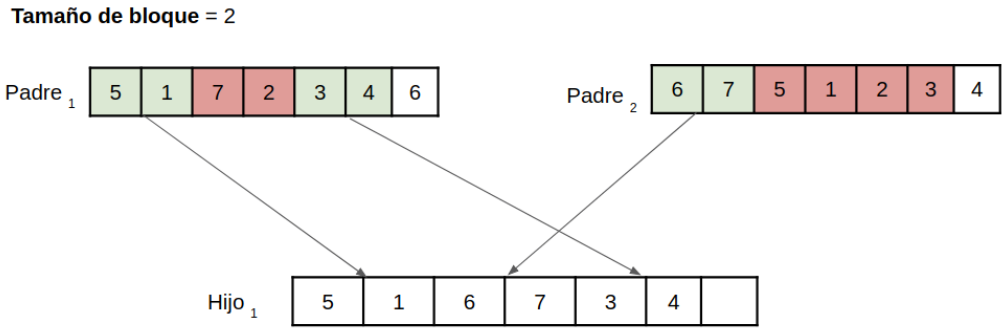

Figura 4. Cruce por parejas

Si se recorren todos los bloques disponibles en en los dos padres y no se ha conseguido generar de forma completa al hijo, entonces se van recorriendo cada una de las posiciones de los padres e insertando el gen correspondiente si no se encuentra entre los genes del hijo (ver la figura 5).

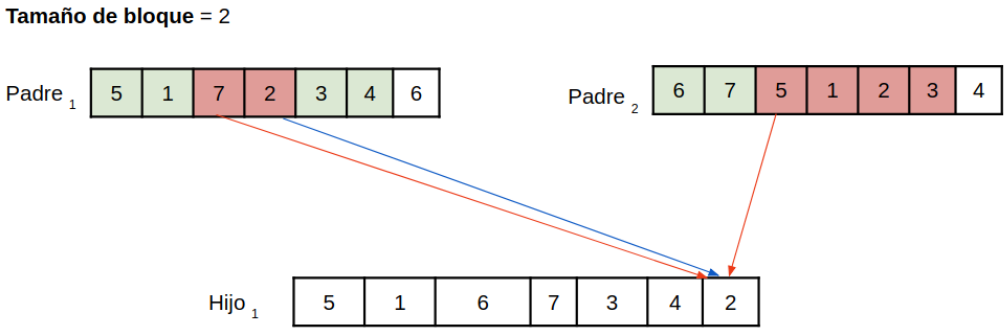

Figura 5. Cruce por parejas

2.5.3. Cruce en ciclo

El cruce en ciclo es la última operación implementada. A diferencia de las otras dos operaciones en esta no se establece ningún parámetro a configurar, únicamente se han de seleccionar dos padre de forma aleatoria de las lista de padres seleccionados anteriormente.

Una vez seleccionados los padres, se baja el primer gen del padre al hijo. Una vez elegidos el primer elemento del padre nos situamos en la primera posición del segundo padre y el valor de este gen será la posición del gen a bajar al hijo del primer padre. Está operación se repite hasta que el padre 1 vuelve a la posición inicial [2]. En la siguiente figura 6 se puede observar parte del proceso.

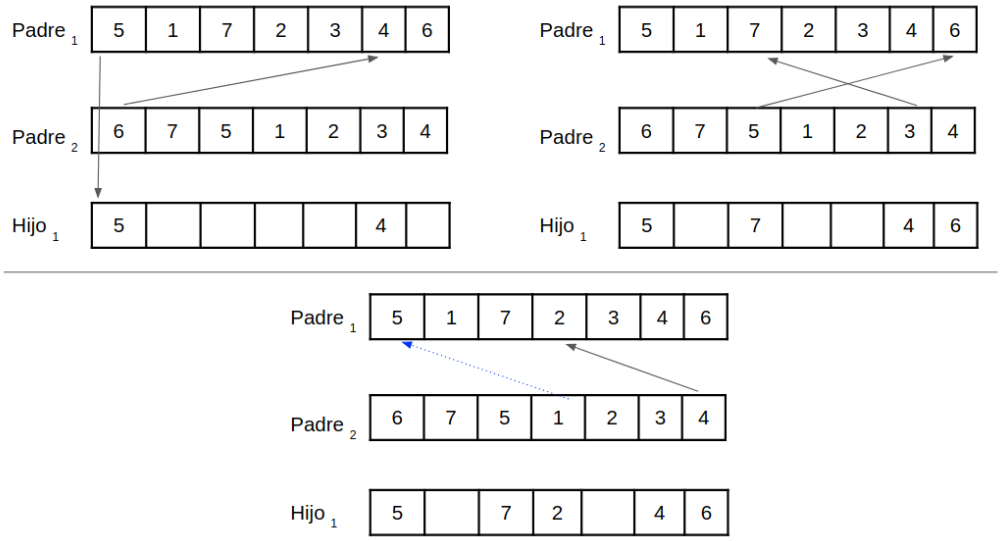

Figura 6. Cruce por ciclo

Como se observa en la anterior figura no todas las posiciones del hijo se completaran necesariamente, pudiendo quedar incluso una una única posición ocupada en caso de que el primer valor del padre 2 fuera igual a 1. Para finalizar la operación de cruce se recorren los elementos del padre 2 de forma ordenada, comprobando si el gen no se encuentra en el hijo y se inserta hasta que el hijo se encuentre completo.

Los dos tipos de cruces anteriores suelen ofrecer mejores resultados. Esto se debe a que el hijo está formado por bloques de más de un gen de los padres. Para el problema del viajante esto suele ser favorable ya que un bloque puede estar formado ya por un pequeño camino bueno. Sin embargo, este último cruce no mantiene ningún camino y se pueden obtener hijos con un valor peor. En el apartado de resultado se muestra de forma gráfica como se ha comportado cada una de estas operaciones.

2.6. Mutación

El proceso de mutación consiste en la modificación de los genes de un individuo, el cual puede mejorar el valor del *fitness* o puede ayudar a salir de un óptimo local. Generalmente se establece una probabilidad de mutación para cada gen de un individuo, pero en el algoritmo desarrollado se ha establecido una probabilidad de mutación diferente que se explica en los siguientes subapartados.

2.6.1. Intercambio

Para esta primera operación de intercambio se han establecido dos parámetros a configurar previamente. La probabilidad de mutación y el número de posiciones a intercambiar en un individuo. La probabilidad establecida se asigna a cada una de las posiciones a intercambiar. Las posiciones a intercambiar se eligen aleatoriamente (ver la figura 7, donde hijo se refiere a un individuo de la población a excepción de los padres seleccionados en el cruce).

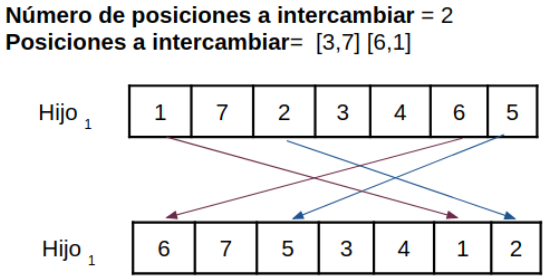

Figura 7. Mutación por intercambio

El número de posiciones a intercambiar es un parámetro que se puede ir modificando a lo largo de las generaciones, generalmente disminuyendo el número de intercambios a realizar en un individuo. Esto se realiza porque cuando el algoritmo está convergiendo hasta un óptimo es posible que la alteración de un tamaño elevado de genes genere individuos peores.

2.6.2. Intersección

En la mutación por intersección se selecciona un valor(ciudad) y posición de forma aleatoria. El valor elegido se insertará en la posición correspondiente y el resto de posiciones se desplazan un elemento a la izquierda o un elemento a la derecha dependiendo donde se encuentre la posición a insertar (ver figura 8). El único parámetro a configurar es la probabilidad de mutación de un individuo.

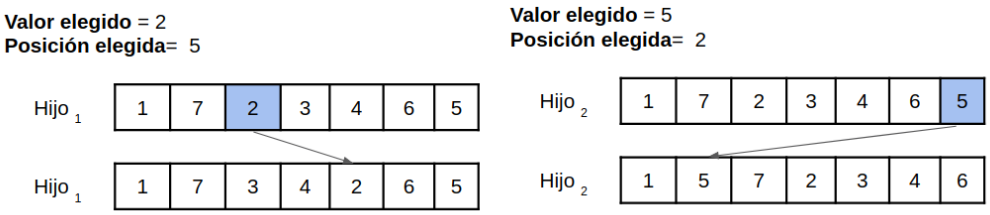

Figura 8. Mutación por intersección

2.6.3. Orden de secuencia inversa

La última operación de mutación por orden de secuencia inversa también está sujeta al parámetro de probabilidad de mutación para cada uno de los individuos. Consiste en elegir dos puntos *a* y *b* en el individuo donde *a menor que b* . Cuando las dos posiciones son seleccionadas entonces se hace un cambio de orden a la inversa como se puede observar en la siguiente Figura 9 [3].

Esta operación ofrece buenos resultados para el problema del viajante debido a que no se modifica el valor de la secuencia elegida sino que solo cambiará el valor entre las conexiones.

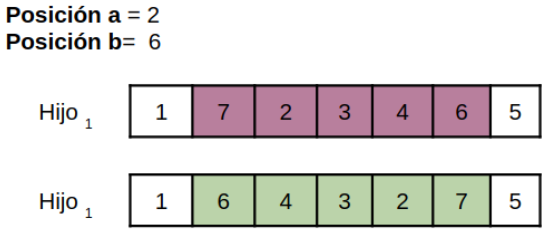

Figura 9. Mutación por orden inverso

2.7. Reemplazo

Esta es la operación que se realizará al final de cada una de las generaciones. La finalidad de esta es añadir nuevos hijos con nueva carga genética a la población. La operación está sujeta a un parámetro que definirá el número de elementos de la población que van a ser reemplazados. Todos los individuos de la población pueden ser reemplazados a excepción del mejor individuo que se mantiene siempre para no empeorar la calidad de la solución.

En esta parte se generan la mitad de individuos de los reemplazados de forma aleatoria. El resto de población que ha quedado vacía es la población que se completará a partir de la operación de cruce. Como se ha mencionado anteriormente se deja momentáneamente parte de la población con sitios vacíos con la finalidad de mantener una talla fija de individuos.

Para finalizar este apartado, hay que decir que previamente a implementar esta parte no se efectuaba reemplazo como tal. En el proceso de selección se seleccionaba a un número de padres, donde, los más fuertes tenían más probabilidades de subsistir y el resto de población se eliminaba. De esta forma se obtuvieron resultados buenos, pero después de comentarlo y discutirlo se llegó a la conclusión de que con este método se podía llegar a converger en un óptimo local ya que a lo largo de todas las generaciones siempre se mantenía la misma carga genética.

2.8. Función de parada

La función de parada es la que a partir de unos parámetros decidirá cuando el algoritmo finalizará su ejecución. Se establecen diferentes parámetros que pueden llevar a la finalización que son:

- **Iteraciones:** se establece un número de iteraciones y una vez alcanzado el algoritmo se detendrá.
- **Convergencia:** si el algoritmo está devolviendo el mismo resultado durante un número de iteraciones el algoritmo se detendrá.

Todos estos parámetros están sujetos a la talla del problema.

2.9. Resultados

En este apartado se aportan diferentes experimentos realizados con el algoritmo descrito. Se han utilizado todas las operaciones así como las diferentes variantes de cada una de ellas con diferentes configuraciones para sus parámetros.

Para la realización del experimento se ha trabajado con un mapa de 96 ciudades (gr96.tzp) donde el valor óptimo encontrado se encuentra en una distancia de 512. La finalidad de los experimentos es encontrar una parametrización óptima del algoritmo que lleve a una solución cercana al óptimo.

En primer lugar se han realizado experimentos con una población pequeña para observar cómo se comporta cada variante y poder extraer unas conclusiones con el fin de realizar cambios en la parametrización. En las dos siguientes tablas se especifican los resultados obtenidos y la parametrización utilizada.

Para los siguientes resultados los parámetros empleados son:

- **Tamaño población:** 50 individuos
- **Iteraciones:** 10000
- **Reemplazo:** 20 individuos
- **Padres seleccionados:** 5 individuos

Selección	**Cruce**	**Mutación**	**Iteración** 1000	**Iteración** 5000	**Iteración** 10000
Torneo	Ordenado	Orden Inverso	1300	1000	660
Torneo	Ordenado	Intercambio	1600	1250	980
Torneo	Ordenado	Intersección	1500	1200	800

Selección	**Cruce**	**Mutación**	**Iteración** 1000	**Iteración** 5000	**Iteración** 10000
Rueda Ruleta	Ordenado	Orden Inverso	2600	1750	1300
Rueda Ruleta	Ordenado	Intercambio	2590	2100	1700
Rueda Ruleta	Ordenado	Intersección	2700	1800	1490

En esta dos primeras pruebas realizadas se ha cambiado únicamente el tipo de selección, donde se observa que la selección por torneo ofrece unos mejores resultados respecto a la selección por rueda de ruleta. La rueda de ruleta no ofrece un mal resultado ya que a lo largo de las generaciones va mejorando su solución, pero los resultados respecto a iteraciones es peor respecto a la selección por torneo. Las siguientes pruebas se van a realizar con los diferentes tipos de cruce explicados y manteniendo en ellos la mutación por orden inverso, la cual es la que mejores resultados a ofrecido en ambos experimentos anteriores.

Selección	**Cruce**	**Mutación**	**Iteración** 1000	**Iteración** 5000	**Iteración** 10000
Torneo	Ordenado	Orden Inverso	1300	1000	660
Torneo	Ciclo	Orden Inverso	1800	1478	1200
Torneo	Parejas	Orden Inverso	2050	2020	1950

Con este nuevo experimento se observa que la mejor variante es el cruce ordenado. El cruce por ciclo también va mejorando pero a una velocidad más lenta. Sin embargo el cruce por parejas es el que peor resultados devuelve, convergiendo en una solución de 1950, muy lejos de una solución como 660 que ofrece el cruce ordenado. En las siguientes figuras (10, 11, 12) se puede observar cómo se comporta el algoritmo para este último

experimento. *(El eje 'y' se refiere a la calidad de la solución y el eje 'x' al número de iteraciones. Igual para el resto de gráficas.)*

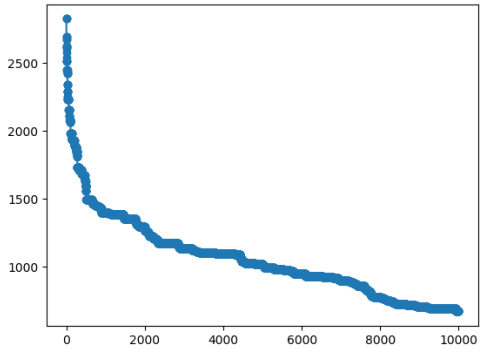

Figura 10. Resultados cruce ordenado

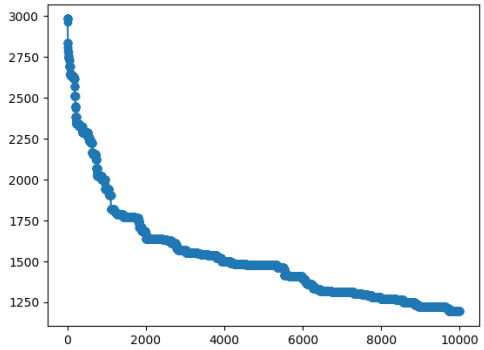

Figura 11. Resultados cruce en ciclo.

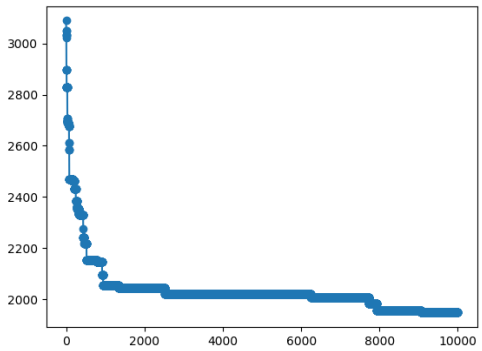

Figura 12. Resultados cruce por parejas.

Es importante mencionar que el cruce ordenado tiene un parámetro a configurar que es el tamaño de bloque. Este se establece a un tamaño de 45 genes y cada 500 generaciones se va decrementando en 5 hasta llegar a un bloque de 5. Si no se realiza esta alteración del parámetro el algoritmo convergerá a una solución muy lejana al óptimo global.

A partir de los resultados obtenidos se van a realizar unos cambios en los parámetros para tratar de alcanzar una solución más cercana al óptimo global. Para ello los parámetro son los siguientes:

- **Tamaño población:** 300 individuos
- **Iteraciones:** 10000
- **Reemplazo:** 150 individuos
- **Padres seleccionados:** 10 individuos

La siguiente configuración de parámetros se va a emplear para las variantes del cruce ordenado y el cruce en ciclo. La selección se regirá por torneo y la mutación por el orden de secuencia inversa.

Selección	**Cruce**	**Mutación**	**Iteración** 1000	**Iteración** 5000	**Iteración** 10000
Torneo	Ordenado	Orden Inverso	1085	582	547
Torneo	Ciclo	Orden Inverso	1450	750	656

Como se puede observar en la tabla y en las dos figuras anteriores, se han obtenidos buenos resultados para ambos experimentos. Sin embargo se observa que el cruce ordenado ofrece mejores resultados, donde en la iteración 5000 ya empieza a converger hacia un resultado muy próximo al óptimo de 512. Por otro lado se observa que el cruce en ciclo también ha ofrecido buenos resultados pero no ha convergido a un valor cercano al óptimo sino a un valor cercano a 660. Hay que nombrar que para los experimento no se ha establecido la función de parada para la convergencia, donde el primer experimento hubiera obtenido un resultado de 560 aproximadamente y el segundo de 660. Con estos experimento se ha concluido que el cruce ordenado junto con la mutación por secuencia inversa es la que mejores resultados ofrece. Si no se establece una función de parada y se añaden más iteraciones es posible que el resultado hubiera sido más próximo al óptimo global.

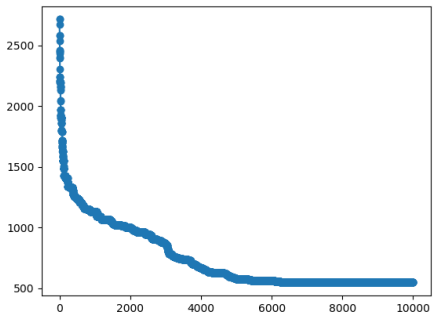

Figura 13. Resultados cruce ordenado.

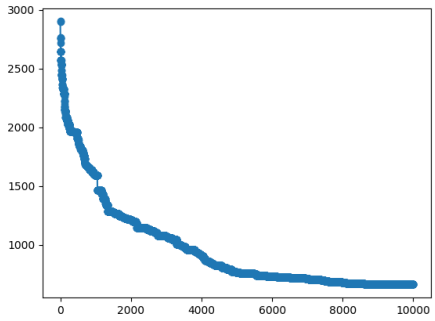

Figura 14. Resultados cruce en ciclo.

Como conclusión a los experimentos realizados, se ha comprobado que para las operaciones de cruce y mutación, si se realiza la alteración de genes de forma aleatoria el resultado obtenido no es el mejor. En cambio si se utilizan operaciones que realizan cambios de forma ordenada los resultados mejoran de forma más rápida y se converge en una solución cercana a la solución óptima. Es posible que en otros problemas estas operaciones de cruce ordenado y mutación por orden inverso de secuencia no ofrezcan los resultados que han ofrecido para el problema del viajante, donde el orden de los genes es importante para la evaluación de su *fitness*.

3. Enfriamiento simulado

Enfriamiento simulado se trata de un algoritmo de búsqueda meta-heurística para problemas de optimización. La finalidad es encontrar una solución que se aproxime al óptimo global. En este tipo de problemas se busca una buena solución y no el óptimo global debido a que el espacio de búsqueda puede ser grande y el tiempo de búsqueda inalcanzable hasta para los ordenadores con más potencia de cálculo.

El nombre de enfriamiento simulado viene de una técnica en la que en primer lugar se calienta el material (metal) hasta una temperatura y después se va enfriando lentamente para poder cambiar sus propiedades físicas. Cuando el metal está caliente los átomos aumentan su energía y pueden desplazarse de sus posiciones iniciales. Al enfriarse da la posibilidad a configuraciones con una energía menor a la inicial.

En un algoritmo de enfriamiento simulado, en primer lugar se establece una solución actual. Seguidamente se definen los parámetros de temperatura inicial, temperatura final y enfriamiento. El parámetro de enfriamiento es el que marcará cuantos vecinos (solución nueva) se van a explorar durante la búsqueda. No todos los vecinos van a mejorar la solución actual, pero cuando la temperatura es elevada estos tendrán mayor probabilidad de ser aceptados y reemplazar la solución actual. A medida que se van explorando los vecinos la temperatura va disminuyendo y a su vez la probabilidad de aceptar un vecino peor es menor. El algoritmo finaliza cuando se alcanza la temperatura final establecida como parámetro. En los siguientes apartados se explica detalladamente el algoritmo.

3.1. Inicialización

En este apartado se habla de los parámetros de inicialización así como de la inicializan las soluciones. La solución actual se inicializa de forma aleatoria, pero siempre cumpliendo con la restricción de no visitar una ciudad más de una vez. Además se inicializa otra variable donde se almacenará el mejor resultado encontrado durante la búsqueda y que en la inicialización tomará el valor de la solución actual.

Como en el anterior algoritmo (genético) las soluciones tienen un valor que está sujeto a la distancia recorrida. En esta caso no se ha implementado la misma función de *fitness* y se ha tenido en cuenta como mejor solución aquella de menor recorrido.

Una vez se ha definido una solución actual, es necesario establecer los parámetros relativos a la temperatura como se ha explicado previamente, donde se han establecido la temperatura inicial y final y por último un parámetro de enfriamiento que llevará al final del proceso. Es posible que durante este proceso de calentamiento y enfriamiento el algoritmo encuentre una solución cercana al óptimo global (óptimo local). Para tratar de salir de este óptimo y explorar otros óptimos, se ha de volver a realizar todo el proceso (reiniciar) y buscar nuevas vecindades que puedan encontrar un óptimo local mejor que el anterior. Para esto se ha establecido un parámetro que definirá el número de veces (iteraciones) que se repetirá el proceso. En el siguiente apartado se explicará cómo se exploran las vecindades y la aceptación de cada una de las soluciones dependiendo de la temperatura.

3.2. Vecindades

Durante el proceso de calentamiento se realiza una búsqueda de vecinos de la solución actual. Para la búsqueda de estas vecindades se han utilizado operaciones imple-

mentadas en el algoritmo genético. Las operaciones utilizadas son la operación de cruce ordenado y la mutación por secuencia inversa.

Como ya se explicó anteriormente, para el cruce ordenado se necesita de dos soluciones para crear una nueva solución. Una de estas soluciones es la solución actual y la segunda (solución nueva o vecina) en la primera búsqueda, será una solución aleatoria. El resultado del el cruce será una solución nueva a la que se le aplicará la operación de mutación ya explicada.

Cuando se establece una solución vecina en la cual se realizan las operaciones, solo la primera solución vecina será aleatoria. Para el resto de soluciones vecinas estarán inicializadas con el valor de la última solución vecina que haya mejorado a la solución actual. Para entender mejor esto último en el próximo apartado se aporta un pseudocódigo con todo el algoritmo.

3.3. Enfriamiento/Aceptación

Una vez se ha buscado un vecino hay que comprobar si este cumple la condición de ser mejor que la solución actual. En caso de ser mejor (distancia de recorrido es menor), la solución nueva pasa a ser la solución actual. En caso de ser peor este vecino no es descartado y se le aplica una probabilidad de ser aceptado acorde a la temperatura actual y a la calidad de la solución. Cada vez que la temperatura va descendiendo la probabilidad de que un vecino peor sea aceptado también disminuye. La fórmula con la que se calcula la probabilidad de ser aceptado es la siguiente:

prob_aceptacion= $e^{(-diferencia/temperatura\_actual)}$

Donde diferencia es:

diferencia= *solucion_nueva-solucion_actual*

Si una solución peor es aceptada por su probabilidad entonces esta reemplaza a la solución actual. Sin embargo, está solución actual buscará un vecino a partir de esta y de una solución anterior mejor como se menciona en el último párrafo del apartado anterior.

Una vez se han realizado los pasos anteriores se actualiza la temperatura con el fin de que esta se vaya acercando a la temperatura final y que la probabilidad de aceptar vecinos peores disminuya. Para conseguir esto se aplica el parámetro de enfriamiento definido en la inicialización y que dependiendo el valor de este, el proceso explorará más o menos vecinos. En la siguiente Figura 15 se adjunta el pseudocódigo del algoritmo completo con la finalidad de que se entiendan todos los pasos explicados.

```
Inicializacion:

      sol_actual=genear_aleatorio()
      sol_mejor=sol_actual;
      temp_inicial;
      temp_final;
      enfriamiento;
      iteraciones;

Para cada iteracion en el rango de (0,iteraciones)
      sol_nueva_m=generar_aleatorio()
      Mientras temp_inicial>temp_final:
            sol_nueva=cruce_ordenado(sol_actual,sol_nueva_m)
            sol_nueva=mutacion(sol_nueva)
            diferencia=sol_nueva-sol_actual
            si diferencia<0 o e^(-diferencia/temp_inicial) > random(0,1):
                  si diferencia<0:
                        sol_nueva_m=sol_nueva
                  si sol_nueva<sol_mejor:
                        sol_mejor=sol_nueva
                  sol_actual=sol_nueva
            temp_inicial=temp_inicial*enfriamiento
sol_mejor
```

Figura 15. Pseudocódigo del algoritmo de enfriamiento simulado implementado.

3.4. Resultados

Con el algoritmo presentado se van a realizar diferentes experimentos sobre el problema de 96 ciudades empleado también en el algoritmo genético. Los experimentos se realizarán a base de modificar los parámetros referente a la temperatura y al enfriamiento. En cada uno de los experimentos se tendrá en cuenta la mejor solución encontrada y el tiempo de cálculo que se ha necesitado para llegar a esta.

En la siguiente tabla se presentan unos primeros resultados donde se altera el parámetro de la temperatura inicial y final y donde solo se realiza una iteración.

Iteraciones	Temperatura inicial	Temperatura final	Enfriamiento	Solución	Tiempo
1	10e5	0.1	0.9	2673.13	0.028 s
1	10e5	1e-5	0.9	2150.06	0.046
1	10e5	1e-20	0.99	708.82	1.160 s

En estos primeros experimentos la temperatura inicial se ha mantenido igual. El primer resultado ha ofrecido una mala solución y por esto se ha decidido cambiar los parámetros de de temperatura final, con el fin de que el proceso de enfriamiento se alargará y así hacer una búsqueda de más vecinos cercanos al óptimo. A causa de esto se ha observado que el segundo experimento mejora el resultado de la primera pero sigue siendo una mala solución. En el último experimento, se ha reducido más la temperatura final y además se ha ampliado el valor de enfriamiento, lo que permite al igual que en la anterior alargar el estado de enfriamiento donde prácticamente solo aceptará soluciones que mejoren a la actual, por lo que se estará explorando un óptimo local. Con las modi-

ficaciones realizadas se ha mejorado ampliamente la solución pero a su vez el tiempo de cálculo también ha aumentado.

Con los experimentos anteriores, se ha comprobado que si se alarga el proceso de enfriamiento se consiguen buenos resultados. En los siguientes experimentos se va a ampliar el la temperatura inicial que alargará el proceso de calentamiento, donde se aceptarán muchos vecinos peores.

Iteraciones	Temperatura inicial	Temperatura final	Enfriamiento	Solución	Tiempo
1	10e10	1e-20	0.99	715.84	1.32 s
1	10e30	1e-20	0.99	642.09	2.25 s
1	10e50	1e-20	0.99	681.82	3.27 s

Se ha comprobado que modificando la temperatura inicial se logra mejorar la solución. Pero también se observa que en el último experimento con una temperatura inicial muy elevada se conseguido un resultado peor respecto al valor del segundo resultado. En las siguientes figuras se observa de forma gráfica como se ha aumentado el tiempo en el que la temperatura era alta y como el tiempo de enfriamiento era menor respecto a esta.

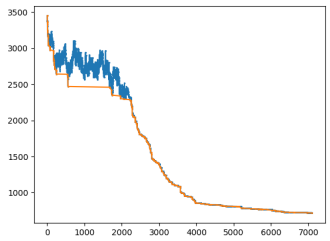

Figura 16. Resultados. Temperatura inicial=10e10 .

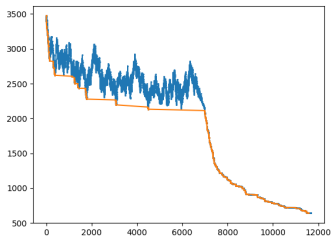

Figura 17. Resultados. Temperatura inicial=10e30 .

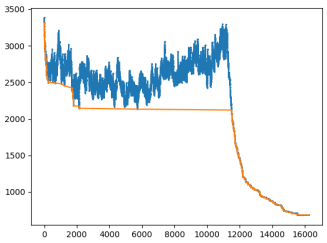

Figura 18. Resultados. Temperatura inicial=10e50 .

En las figuras anteriores se observa que, si la temperatura inicial es muy elevada el tiempo de enfriamiento es menor y a su vez los resultados han sido peores. Con esta conclusión se realizan unos experimentos donde se aumentará el valor de la temperatura final y el valor de enfriamiento para observar si los resultados se acercan al óptimo global de 512. Resultados en la siguiente tabla.

Iteraciones	Temperatura inicial	Temperatura final	Enfriamiento	Solución	Tiempo
1	10e20	1e-30	0.99	642.668	2.24 s
1	10e10	1e-20	0.995	552.80	2.83 s
1	10e10	1e-30	0.995	622.82	3.77 s

Con estos resultados se ha mostrado que aumentado el valor para el parámetro del factor de enfriamiento se ha conseguido mejores resultados. Donde se ha conseguido un valor muy cercano al óptimo global. Por último se va a realizar un experimento donde se aumentarán de una a cinco iteraciones para las dos últimas configuraciones empleadas.

Iteraciones	Temperatura inicial	Temperatura final	Enfriamiento	Solución	Tiempo
5	10e10	1e-30	0.995	547.25	22.57 s
5	10e10	1e-20	0.995	552.80	13.76 s

Con este último resultado se ha conseguido el mejor resultado pero su tiempo de ejecución ha sido mayor debido al aumento de las iteraciones. Respecto a resultados ambas configuraciones han devuelto resultados muy similares pero el último en menos tiempo. Para intentar mejorar el resultado podría resultar conveniente aumentar la iteraciones con la última configuración. En las Figuras 20 y 19 se observan los resultados.

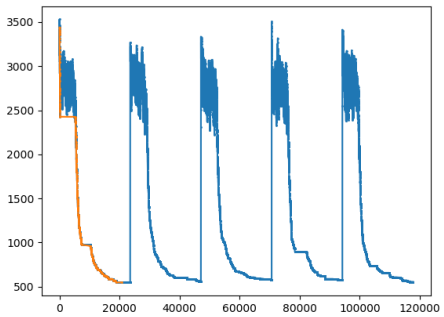

Figura 19. Resultados con 5 iteraciones. Temperatura final=10e30

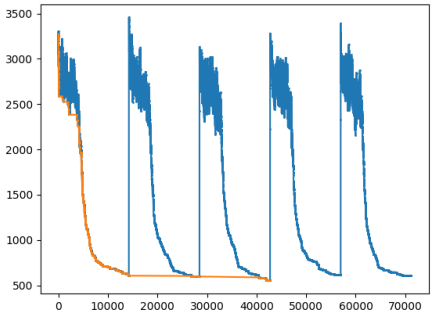

Figura 20. Resultados con 5 iteraciones. Temperatura final=10e20

En los experimentos realizados se ha observado que el algoritmo de enfriamiento simulado para el problema del viajante con una talla de 96 ciudades se comporta mejor cuando se disminuye la temperatura final y el factor de enfriamiento. Con estos parámetros el tiempo aumenta pero se encuentra una solución cercana al óptimo global en un tiempo de 13 segundos. También se ha observado que la utilización de las operaciones de los algoritmos genéticos para la búsqueda de vecinos ha ofrecido buenos resultados.

4. Comparativa

A partir de los diferentes resultados obtenidos se va a realizar una comparativa entre el algoritmo genético y el de enfriamiento simulado. Donde en ambos resultados se ha conseguido llegar a una solución de 547 sobre 512 que es el óptimo global. Las configuraciones utilizadas para encontrar estas soluciones han sido:

Algoritmo genético

- **Tamaño población:** 300 individuos
- **Iteraciones:** 10000
- **Reemplazo:** 150 individuos
- **Padres seleccionados:** 10 individuos

Selección	Cruce	Mutación	Iteración 1000	Iteración 5000	Iteración 10000
Torneo	Ordenado	Orden Inverso	1085	582	547

Tiempo de ejecución: 319.061 segundos.

Enfriamiento simulado

Iteraciones	Temperatura inicial	Temperatura final	Enfriamiento	Solución	Tiempo
5	10e10	1e-30	0.995	547.25	22.57 s

Ambos algoritmos han encontrado la misma solución pero en este caso la diferencia se encuentra en el tiempo de ejecución de cada uno de estos, donde, el algoritmo genético ha demorado mucho más tiempo para encontrar una buena solución. Es posible, que con el mismo tiempo empleado por el algoritmo genético el algoritmo de enfriamiento simulado hubiese mejorado la solución.

Otra punto importante a destacar entre ambos algoritmos es la configuración de parámetros. Para el algoritmo genético desarrollado se han de configurar diferentes parámetros para cada una de sus operaciones, algo que puede llevar un tiempo hasta encontrar una configuración óptima. Por parte del enfriamiento simulado, solo hay que configurar los parámetros respecto a la temperatura, el factor de enfriamiento y la operación de cruce dependiendo la talla del problema.

5. Conclusiones

En este trabajo se ha comprendido como trabajan dos tipos de algoritmos para la búsqueda de soluciones de optimización. Ha sido interesante desarrollar un algoritmo genético y poder realizar configuraciones en sus parámetros para mejorar las soluciones en las diferentes pruebas. Con la implementación de diferentes tipos de cruce y mutación se ha podido comprobar la efectividad de cada uno de estos y que para cada problema en concreto, en este caso el del viajante, no todas las operaciones ofrecen el mismo resultado. Sin embargo, las diferentes operaciones que se han de realizar en un problema con una talla de problema elevada hace que el tiempo de búsqueda para una solución se haga largo.

Para el algoritmo de enfriamiento simulado, el trabajo de implementación ha sido menor ya que el algoritmo es más sencillo de implementar. Además, se han comprobado que las mejores operaciones de cruce y mutación utilizadas en algoritmos genéticos también ofrecen buenos resultado si se añaden al algoritmo de enfriamiento. Ha sido interesante comprobar que con unos simples cambios en los parámetros se han conseguidos resultados muy buenos en muy poco tiempo.

Personalmente, con la implementación del algoritmo genético he aprendido a desarrollar poblaciones que con ciertas operaciones van evolucionando a lo largo del tiempo. La búsqueda de diferentes operaciones, su implementación y la búsqueda de parámetros óptimos me han servido para entender el funcionamiento completo de este tipo de algoritmos. Por su parte el enfriamiento simulado ha sido más fácil de desarrollar y por contra consigue mejores resultados.

Referencias

[1] Wikipedia. Problema del viajante. Consultado en `https://es.wikipedia.org/wiki/Problema_del_viajante` [Último acceso: Septiembre 2018].

[2] Rubicite.Cycle Crossover Operator. Consultado en http://www.rubicite.com/Tutorials/GeneticAlgorithms/CrossoverOperators/CycleCrossoverOperator.aspx [Último acceso: Septiembre 2018].

[3] Otman Abdoun, Chakir Tajani, Jaafar Abouchabka. (2012). Hybridizing PSM and RSM Operator for Solving NP-Complete Problems: Application to Travelling Salesman Problem. *arXiv:1203.5028*