UNIVERSIDAD TECNOLÓGICA NACIONAL

FACULTAD REGIONAL ROSARIO

Cátedra: Algoritmos Genéticos

Trabajo Práctico Número 3:

Aplicación de diversos métodos para la resolución del problema del viajante de comercio (TSP)

Integrantes del grupo:

Nombre	**Legajo**	**Correo**
Bassi, Danilo	43725	danilo-bassi@hotmail.com
Bella, Sebastián	43784	sebastian.o.bella.it@gmail.com
Garello, Iván	43804	ivangarello@gmail.com
Vacchino, Lucas Nehuén	43870	lnvacchino@gmail.com
Zilli, Joel Matías	44107	joelz97@outlook.com

Ciclo lectivo: 2020

Profesores:
Daniela Diaz y Victor Lombardo

Índice

Aplicación de diversos métodos para la resolución del problema del viajante de comercio

Resumen

Este documento se embarca en la resolución de un problema conocido como Problema del Viajante de Comercio (TSP). Para esto, se describen y desarrollan tres técnicas distintas basadas en la búsqueda de soluciones óptimas: búsqueda exhaustiva, búsqueda heurística y algoritmo genético, con el fin de encontrar la ruta más corta que conecte las 24 capitales de Argentina. Dichas técnicas (excepto la búsqueda exhaustiva) fueron programadas en lenguaje Python y en este trabajo se puede encontrar tanto el código como los resultados obtenidos de las distintas ejecuciones realizadas.

Palabras Clave: Problema del viajante, Heurística, Solución óptima, Algoritmos Genéticos

1. Problema del viajante

El problema del viajante (también conocido como problema del viajante de comercio o por sus siglas en inglés: TSP (Traveling Salesman Problem), es uno de los problemas más famosos (y quizás el mejor estudiado) en el campo de la optimización combinatoria computacional.

El problema del viajante se basa en la hipótesis de la existencia de un comerciante (u otra persona o cosa) con el deseo de recorrer una determinada cantidad de ciudades distanciadas entre sí en un solo viaje pasando por todas ellas. El objetivo del problema es encontrar una ruta que, comenzando y terminando en una ciudad en particular, pase una sola vez por cada una de las ciudades y minimice la distancia recorrida por el viajante.

A pesar de la aparente sencillez de su planteamiento, el TSP es uno de los problemas más complejos de resolver. Este se encuentra entre los problemas denominados NP-Complejos, esto es, los problemas que no se pueden resolverse en tiempo polinomial en función del tamaño de la entrada (en este caso el número N de ciudades que el viajante debe recorrer).

La solución más directa es la que aplica la fuerza bruta: evaluar todas las posibles permutaciones y quedarse con la mejor. No obstante, el número de posibles ciclos viene dado por el factorial del número de ciudades (N!) y esto hace que la solución por fuerza bruta sea impracticable para valores de N incluso moderados

Por ende, para resolver el problema del viajante se deben aplicar otros métodos que logran conseguir una respuesta en un menor tiempo, aunque muy probablemente no sea la solución óptima. Estas formas de resolución tienen por objetivo conseguir una solución calificada como ”muy buena”.

2. Solución aplicando Método Exhaustivo

La búsqueda exhaustiva es una técnica general de resolución de problemas que consiste en recorrer todas las combinaciones posibles de los elementos de la solución, dando como resultado el mejor de todos ellos. Aunque esta metodología logra obtener la solución óptima, no es aplicable para nuestro trabajo. Esto se debe a que la cantidad de combinaciones total de las soluciones de nuestro problema es demasiado grande. Para el problema del viajante, el método exhaustivo se basa en generar todas las rutas que se pueden realizar pasando por cada una de las n ciudades. Si este problema tuviera 3 ciudades: A, B y C, entonces habría en total 6 rutas distintas a analizar: A-B-C, A-C-B, B-A-C, B-C-A, C-A-B, C-B-A.

En lenguaje matemático, se puede definir esta operación como una permutación: 3! = 3.2.1 = 6. Si el problema tuviera 4 ciudades, la cantidad de rutas posibles que tendría como solución es 4! = 4.3.2.1 = 24. De esta forma, podemos saber cuantas rutas posibles existen para este tipo de problemas a partir de una cantidad de ciudades antes determinada.

Cant. de ciudades (n)	Recorridos posibles (n!)
2	2
3	6
4	24
5	120
6	720
7	5.040
8	40.320
9	362.880
10	362.8800
11	39.916.800
12	479.001.600
13	6.227.020.800
14	87.178.291.200
15	1.307.674.368.000
...	...
20	$2{,}433 \times 10^{18}$
...	...
24	$6{,}204 \times 10^{23}$

Tabla 1: Cantidad de Soluciones Posibles para el problema del viajante con distintas cantidad de ciudades

Esto nos demuestra que para nuestro planteo del problema del viajante con 24 ciudades, el método exhaustivo debería calcular la distancia de $6{,}204 \times 10^{23}$ recorridos distintos, siendo esta tarea prácticamente imposible de llevar a cabo; si utilizáramos una computadora que calcule la distancia de 100.000.000

de rutas por segundo, el tiempo que se tardaría en buscar la distancia de todas las combinaciones es:

$$\frac{24!}{100{,}000{,}000} = 6{,}204{,}484{,}017{,}332{,}394 \ segundos$$

También, podemos calcular resultados equivalentes para distintas unidades de tiempo:

Medida	Unidad
6.204.484.017.332.394	Segundos
123.467.782.592	Horas
196.742.897	Años
196.742	Siglos

Tabla 2: Resultados equivalentes para distintas unidades de tiempo

Esto nos demuestra la imposibilidad de aplicar el método exhaustivo para resolver el problema planteado en este documento.

3. Solución aplicando Método Heurístico

En general, los métodos heurísticos son preferibles en la solución de problemas difíciles donde una búsqueda exhaustiva necesitaría un tiempo demasiado grande para encontrar el resultado óptimo. Las soluciones encontradas a través de métodos heurísticos tienen la desventaja de no ser soluciones óptimas, pero pueden obtenerse de manera rápida y en la mayoría de los casos son buenas y/o aceptables.

En nuestro caso, se programó la siguiente heurística para resolver el trabajo practico: ”Desde cada ciudad ir a la ciudad más cercana no visitada”.

Como los datos de las distancias entre ciudades fueron proveídas por la cátedra en una tabla de Excel, el algoritmo funciona de la siguiente manera (ver fig. 1):

Se comienza asignando la variable *recorrido* como un arreglo donde solamente se posee la provincia inicial enviada por parámetro. También se asigna a la variable *distTotal* 0, dado que es donde se acumulará la distancia total recorrida.

Luego, la variable *provActual* indica la provincia donde se está actualmente en el recorrido del while. Inicialmente posee el valor de la ciudad enviada por parámetro.

Una vez ya iniciado el ciclo, se llama al método CalculaProvMasCercana() y se le envía por parámetro la provincia actual y el recorrido hecho hasta ahora. Este método básicamente recorre toda la fila de Excel correspondiente a la provincia enviada como primer parámetro y devuelve la provincia más próxima que no se encuentre en el recorrido enviado como segundo parámetro, junto con la distancia entre provincias.

Luego a esto, solo nos queda sumar la distancia que se obtuvo a la distancia total, agregar la provincia que se retornó al arreglo *recorrido* y actualizar la variable que indica la provincia actual por la provincia devuelta por el método CalculaProvMasCercana().

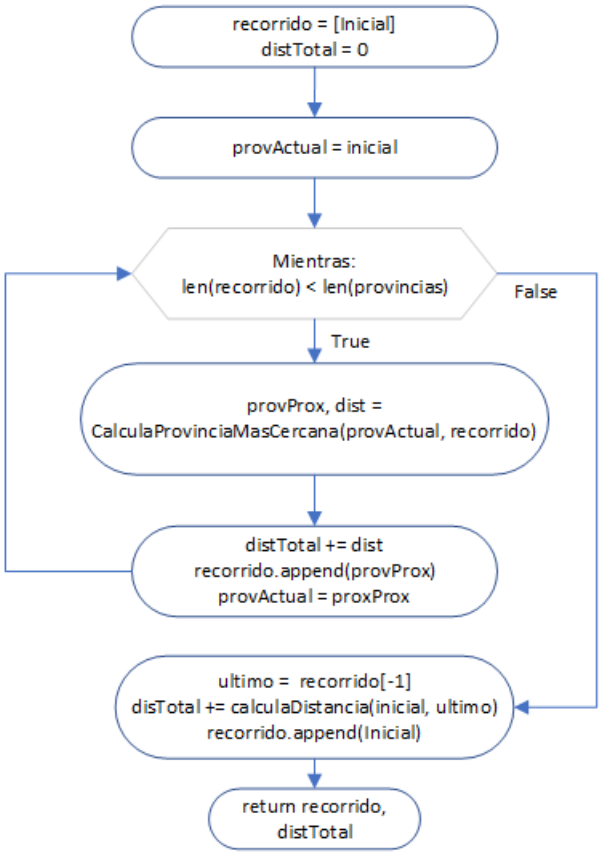

Figura 1: Diagrama de flujo de la heurística programada en Python.

Una vez que el recorrido tenga todas las provincias dentro del mismo, se terminará el ciclo while. En este punto es necesario agregar el primer elemento al final del recorrido, dado que el problema requiere que se vuelva a la ciudad de inicio. Antes de hacer esto, se calcula la distancia entre la provincia inicial y el ultimo del recorrido (en Python el ultimo elemento de una lista se puede acceder con el índice -1). Finalmente se devuelve el recorrido, con la misma ciudad de inicio y fin, junto con la distancia total del recorrido.

El código de esta función puede encontrase en el Anexo, en el archivo Main.py, función: ”Greedy(Inicial)”.

Finalmente, para encontrar el mejor recorrido utilizando la heurística, solo es necesario invocar el método mencionado para cada una de las provincias, y quedarnos solo con el recorrido que tenga la menor distancia total.

Realizando lo anterior mencionado, el resultado obtenido es el siguiente recorrido (Ver fig. 2):

1. Neuquén
2. Santa Rosa
3. San Luis
4. Mendoza
5. San Juan
6. La Rioja
7. S.F.d.V.d. Catamarca
8. Sgo. Del Estero
9. S.M. de Tucumán

10. Salta
11. S.S. de Jujuy
12. Resistencia
13. Corrientes
14. Formosa
15. Posadas
16. Paraná
17. Santa Fe
18. Córdoba
19. Cdad. de Bs. As.
20. La Plata
21. Viedma
22. Rawson
23. Río Gallegos
24. Ushuaia
25. Neuquén

Cantidad de Km recorridos: 9335

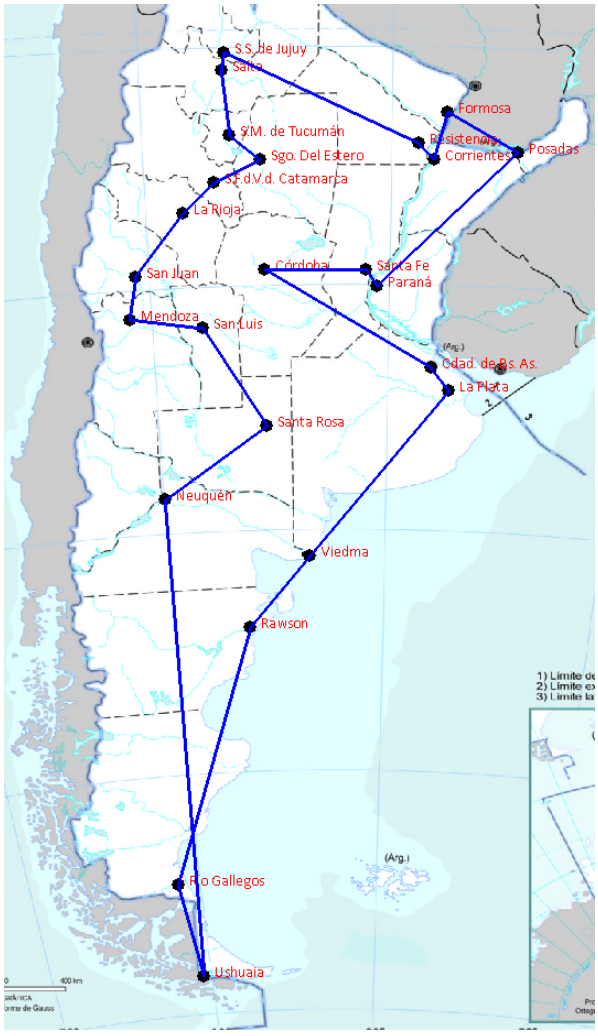

Figura 2: Visualización del recorrido realizado utilizando la heurística.

4. Solución aplicando Algoritmos Genéticos

Una manera de poder resolver el problema del viajante es aplicar los conocimientos teóricos y prácticos que nos aportan los Algoritmos Genéticos. En este caso, se parte de un conjunto de rutas generadas aleatoriamente. A partir de allí, se observan cuáles de estos podría llegar a representar una ruta óptima y se los prioriza por sobre los demás. Luego se van seleccionando pares de soluciones que tienden a ser los mejores de todos los obtenidos para así generar dos rutas nuevas, que pueden tener las mismas características o pueden ser resultado de una mezcla de ambas soluciones parciales. Este proceso se itera hasta una cierta cantidad de veces. La ruta con la menor distancia obtenida en toda la ejecución se la puede considerar como la más cercana a la solución óptima del problema.

El valor de la distancia recorrida por el viajante en una ruta determinada la llamaremos con el nombre de "función objetivo" y será definida de la siguiente manera:

$$f_{obj}(x) = \sum_{r=1}^{m-1} d_{r,r+1} + d_{m,1}$$

Donde m es la cantidad de ciudades a recorrer y $d_{i,j}$ es la distancia que hay entre las ciudades i y j.

4.1. *Parámetros de Entrada*

- Probabilidad de Crossover = 0.90
- Probabilidad de Mutación = 0.05
- Tamaño de la población: 50
- Ciclos del programa: 200
- Método de selección: Ruleta
- Método de Crossover: Cíclico
- Método de Mutación: Intercambio (Swap Mutation)
- Elitismo: Si, con posibilidad de no utilizarlo
- Tamaño de Elite: 10 Cromosomas

4.2. *Población, cromosomas y genes*

La población a utilizar está conformada por un conjunto de 50 cromosomas. La población se inicializa en el comienzo del programa, asignando 50 cromosomas generadas aleatoriamente. Luego, los cromosomas de esta población irán modificándose en el transcurso de la ejecución del programa, tendiendo a mantener aquellos que representen una buena solución.

Un cromosoma representa una posible solución, es decir, representa una ruta que visita todas las ciudades sin pasar dos veces por alguna de ellas y que termine en la ciudad en donde inició. Cada cromosoma contiene 24 genes y cada gen representan las ciudades a visitar. Por lo anterior, decimos que dos genes consecutivos representa un viaje que parte de la primera ciudad a la que hace referencia el primer gen hacia la otra ciudad asociada al segundo gen. Se hace la aclaración de que el

primer y último gen, a pesar de no tener un gen que lo conecte antes o después respectivamente, se conectan entre sí a través de un viaje con partida en la ciudad del último gen y llegada en la ciudad del primer gen, formando así el ciclo solicitado por el enunciado.

Un gen puede tomar un valor entre 1 y 24, siendo este número el ID de la provincia al que hace referencia. Un cromosoma no puede poseer dos genes iguales, ya que esto significaría pasar dos veces por una misma ciudad.

4.3. Población inicial

La población se inicializa en el comienzo de la ejecución del programa, asignando 50 cromosomas al azar. Estos cromosomas son generados a partir de la asignación aleatoria de 24 genes con la condición de no repetir un gen previamente asignado en el mismo cromosoma.

Técnicamente hablando, para generar un cromosoma al azar, se asigna para el primer gen un valor de forma aleatoria ubicado entre 1 y 24. Luego, se asigna para el segundo gen un valor también ubicado entre 1 y 24, pero que debe ser distinto al valor del primer cromosoma. Luego, se asigna el tercer gen con un valor ubicado en el mismo rango pero que no sea igual al valor de alguno de los genes antes asignados. Este proceso se repite hasta asignar los 24 valores en el cromosoma.

4.4. Selección

Una vez formada la población inicial, pasamos al método de selección en donde se van a elegir a los mejores cromosomas para luego reproducirlos entre sí y formar la próxima generación. Para lograr esto, primero debemos asignar a cada cromosoma una puntuación llamada ”Fitness”que representa cuán cerca está una solución de ser la mejor en comparación de los demás cromosomas de la generación a la que pertenece. A partir de este fitness, se determinará los cromosomas que se van a reproducir y aquellos que se van a eliminar.

Para nuestro problema, la funcion Fitness para un cromosoma en particular está definido de la siguiente manera:

$$Fitness(x) = \frac{g(x)}{\sum_{i=1}^{n} g(i)}$$

$$g(x) = \sum_{i=1}^{n} f_{obj}(i) - f_{obj}(x)$$

Donde n el tamaño de la población y $f_{obj}(x)$ es la función objetivo del cromosoma x.

Cuando terminamos de evaluar el fitness de cada cromosoma, se tiene que crear la nueva población tratando de que los buenos rasgos de los mejores cromosomas se transmitan a ella. Para esto, como método de selección de los mejores padres, hemos aplicado el método de la rueda de ruleta.

La rueda de ruleta cosiste en crear un pool genético formado por cromosomas de la generación actual, en donde cada cromosoma ocupa de ese pool una cantidad proporcional a su fitness. Por lo tanto, dentro de este pool se escogerán parejas de cromosomas de manera aleatoria para que se puedan emparejar, logrando que estas sean las mejores de la generación y sin importar incluso que ambas parejas sean iguales.

4.5. Crossover

El crossover consiste en el intercambio de material genético entre dos cromosomas. En caso de que se emparejen dos descendientes del mismo padre, no generaría ningún inconveniente, garantizando la perpetuación de un individuo con buena puntuación. En caso contrario, que se deban cruzar dos cromosomas distintos, ambos cromosomas padres generarán dos cromosomas hijos cuyos genes resulta de la mezcla genética de los padres.

El método de reproducción que hemos usado para realizar este proceso es el crossover cíclico. Este método consiste en seleccionar un conjunto de genes del primer padre y copiarlos en el cromosoma hijo respetando su valor y posición. Una vez hecho esto, se completa el resto de los lugares que quedaron sin rellenar con los genes del segundo padre cumpliendo los valores y posiciones de los genes seleccionados.

La selección de ese conjunto de genes, se realizan los siguientes pasos:

1. Tomar el primer gen del primer padre y copiarlo al hijo.
2. Luego, Tomar el valor del primer gen del segundo padre y observar el gen del primer padre ubicado en la posición del valor tomado.
3. Tomar el valor del gen observado y Copiarlo en el cromosoma del hijo, respetando valor y posición.
4. Los pasos dos y tres se repiten reiteradas veces hasta que el valor encontrado en el segundo padre coincide con el valor de un gen ya copiado en el hijo.
5. Por último, los espacios vacíos se completan con los genes del segundo padre, también respetando valores y posiciones.

Una vez obtenido el primer hijo, se repiten los pasos alternando los padres para así obtener el segundo cromosoma hijo.

Cabe destacar que el crossover, así como la mutación, tienen una probabilidad de ocurrencia. En caso que no ocurra la reproducción, los padres pasarán directamente a la próxima generación como hijos.

4.6. Mutación

En un algoritmo genético tendrán el objetivo de ocurrir con baja frecuencia para contribuir a la diversidad genética.

En nuestro caso, el operador mutación se aplica a cada hijo de manera individual, con probabilidad de ocurrencia del 5 %, y consiste en el intercambio de dos genes del cromosoma seleccionados al azar. Es decir, luego de seleccionar dos genes, se copia el valor del primero para asignárselo en el segundo y lo mismo con el valor del segundo gen, que se lo copia en el primero.

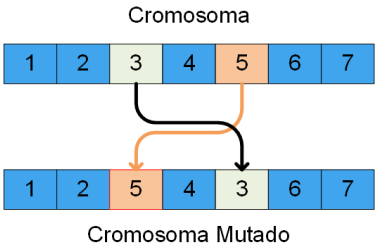

Figura 3: Ejemplo de la mutación utilizada aplicada a un cromosoma con 7 genes.

4.7. Elitismo

El elitismo es un mecanismo que pretende asegurar que los individuos más aptos de la población sobrevivan y continúen participando en el proceso evolutivo pasando a la siguiente generación de manera intacta, sin reproducirse ni mutarse.

El elitismo se aplicará en nuestro programa seleccionando en cada generación los diez cromosomas con mejor fitness y preservándolos para pasarlos sin ninguna modificación a la generación siguiente. Luego, para conseguir los cromosomas restantes se realiza el algoritmo normal, utilizando los mecanismos de selección, crossover y mutación ya mencionados.

En nuestro primer trabajo practico se realizó un elitismo con 2 cromosomas en una población de 10. Decidimos mantener esa misma relación (1/5 de la población total) para este trabajo práctico. Por lo tanto, la cantidad de cromosomas a pasar a la próxima generación sin alterar es de 50/5 = 10.

4.8. Solución obtenida

Una vez realizado el código y ajustado los parámetros de entrada, se realizó una ejecución utilizando elitismo y los resultados fueron los siguientes: Las capitales se recorrieron en este orden:

1. Sgo. Del Estero
2. Santa Rosa
3. Neuquén
4. Viedma
5. Rawson
6. Río Gallegos
7. Ushuaia
8. Posadas
9. Formosa
10. Corrientes
11. Resistencia
12. Paraná
13. Cdad. de Bs. As.
14. La Plata
15. Santa Fe
16. Córdoba
17. San Luis
18. Mendoza
19. San Juan
20. La Rioja
21. S.F.d.V.d. Catamarca
22. S.M. de Tucumán
23. S.S. de Jujuy
24. Salta
25. Sgo. Del Estero

Cantidad de Km recorridos: 10893

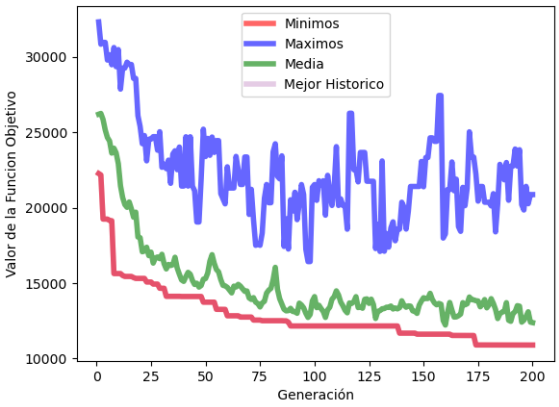

Figura 4: Valores de la función objetivo para cada generación del algoritmo genético.

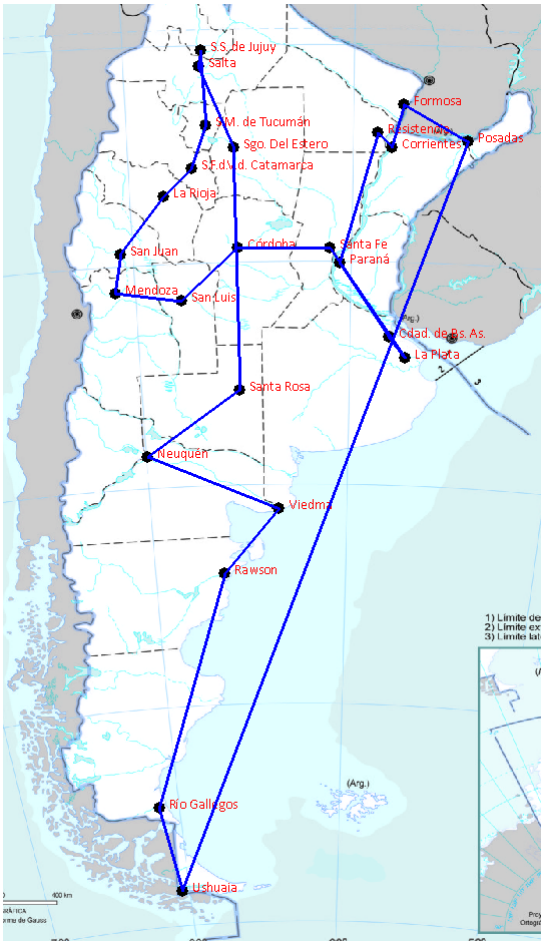

Figura 5: Visualización del recorrido obtenido utilizando el algoritmo genético con elitismo.

Se observa que el mejor recorrido obtenido como salida del programa comienza en la ciudad de Santiago del Estero. Luego continúa con las ciudades de la Patagonia para seguir con las del noreste argentino. Luego, pasa por las ciudades de la Provincia de Buenos Aires, Santa Fe, Córdoba y por último aquellas ciudades de Cuyo y del noroeste.

4.9. Comparación: Heurística y Algoritmo Genético

La solución obtenida en una de las corridas del algoritmo genético, detallada en la sección anterior, muestra una ruta que comienza (y finaliza) en la ciudad de Santiago del Estero y que tiene una longitud total de 10.893 Km.

Ahora, para compararlo con el método heurístico, precisamos el recorrido que se obtiene del mismo ingresando como parámetro de entrada a Santiago del Estero como ciudad de partida para así tener el la longitud de su recorrido y comparar resultados.

La ruta solución a través del método heurístico es la siguiente:

1. Sgo. Del Estero
2. S.M. de Tucumán
3. S.F.d.V.d. Catamarca
4. La Rioja
5. San Juan
6. Mendoza
7. San Luis
8. Córdoba
9. Santa Fe
10. Paraná
11. Cdad. de Bs. As.
12. La Plata
13. Santa Rosa
14. Neuquén
15. Viedma
16. Rawson
17. Río Gallegos
18. Ushuaia
19. Resistencia
20. Corrientes
21. Formosa
22. Posadas
23. Salta
24. S.S. de Jujuy
25. Sgo. Del Estero

Cantidad de Km recorridos: 10419

En consecuencia, se ve que el método heurístico dio como salida una ruta más corta que la obtenida con el algoritmo genético. Resultó ser 474 km o, de manera relativa, un 4,35 % más corta. Aunque se haya una diferencia entre ambas rutas, también podemos decir que son soluciones relativamente eficaces, lo cual nos enseña que el uso de ambos métodos no afecta considerablemente en la calidad de la solución calculada. Pero, si se toma en cuenta la optimización de los recursos, hay que aclarar que el método heurístico es mucho más eficiente con

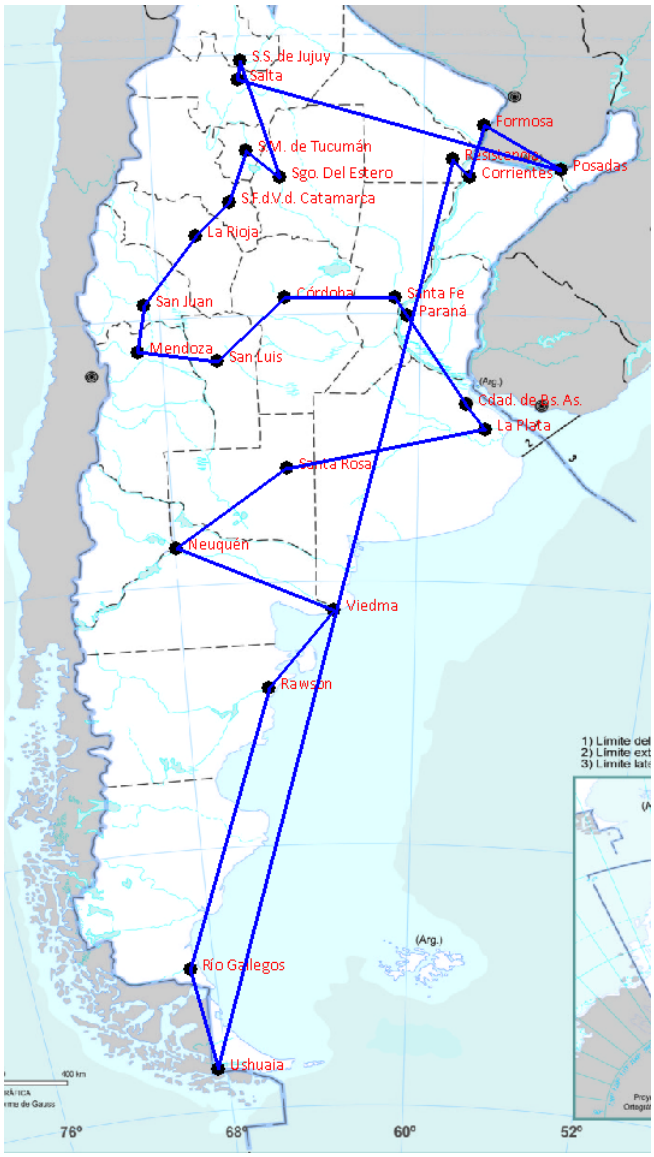

Figura 6: Visualización del recorrido realizado utilizando la heurística y tomando como punto de partida la ciudad de Santiago del Estero

respecto al algoritmo genético ya que consumo menos procesamiento de CPU, menos memoria y, por ende, menos tiempo. Además, la producción de un programa que genere una solución en forma heurística es mucho más fácil y sencillo que la realización de un código con algoritmo genético. Por lo cual, se puede juzgar a la heurística como la herramienta útil para problemas parecidos al tratado en este documento. Pese a eso, las características del problema, como cantidad de nodos o tipo y valor de costo de los arcos, pueden modificar le eficiencia y eficacia de algunos de estos algoritmos, por lo que no hay que pasar por alto la noción de que la breve conclusión realizada anteriormente se aplica para este problema particular.

4.10. Análisis de sensibilidad

Por último, se ha llevado a cabo un acotado análisis de sensibilidad. Aquí, hemos modificado levemente los valores de la probabilidad de crossover y la probabilidad de mutación. Luego, en cada modificación se ha ejecutado el programa y obtenido la solución aportada. Las distancias totales de las soluciones se encuentran descritas en el siguiente cuadro.

Probabilidad de Crossover	Probabilidad de Mutación				
	0.2	0.15	0.1	0.05	0.02
0.9	11274	11620	12687	14056	14373
0.85	12598	11338	11858	13456	15876
0.8	10521	12120	10950	11978	16337
0.75	10973	11632	11513	12918	14970
0.7	11570	11928	11739	11079	14224

Figura 7: Diferentes corridas del algoritmo genético utilizando distintos parámetros.

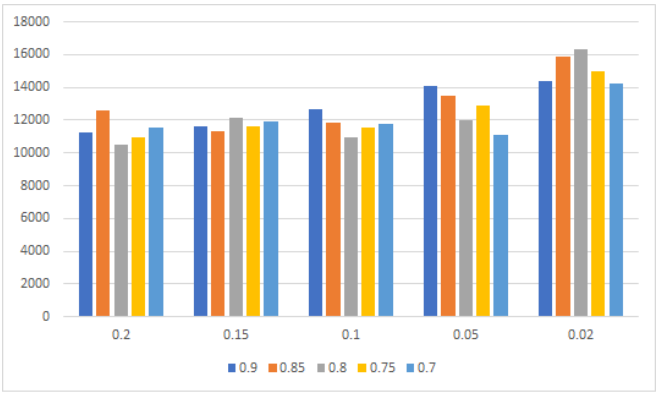

Figura 8: Valores de la Figura 7 agrupados por probabilidad de mutación

Si se observan los resultados en base con las variaciones de la probabilidad de crossover, se puede afirmar que no hay una tendencia clara. Si tomamos el caso de la segunda columna de valores, con una probabilidad de mutación de 0,15, la ejecución con mejor resultado resultó ser aquel que tuvo como probabilidad de crossover 0,85. Pero, si se observa la cuarta columna (con una probabilidad de 0.05) la mejor solución fue generado con una probabilidad de crossover de 0,7. Por lo tanto, concluimos que la variación de la probabilidad de crossover, siempre y cuando esta no disminuya de 0,7, es independiente del resultado obtenido.

Por otro lado, si se estudian los valores tomando en cuenta la probabilidad de mutación se observa una cierta tendencia. Según las corridas realizadas, una probabilidad de mutación entre el 10 y el 20 por ciento da mejores resultados que corridas con valores inferiores. Los cual nos sugiere que en este problema la mutación aporta una diversidad en la población que influye positivamente en la mejora de las siguientes generaciones. Aunque, como se mencionó en trabajos anteriores, no se debe abusar de ella, dado que valores grandes convierten al algoritmo genético en una búsqueda meramente aleatoria.

5. Aportes Prácticos

El problema del viajante de comercio es un problema reconocido por ser la base para la búsqueda y el estudio de diversas metodologías que ayuden a obtener una solución lo bastante cercana a la solución óptima. Además, existen varias situaciones en la realidad que se pueden modelar a este tipo de planteo. Esto permite que, gracias a las incesantes investigaciones sobre las distintas formas de resolver el problema TSP, las dificultades de la vida real puedan ser resueltas con relativa facilidad a pesar de su alta complejidad.

En esta sección se enumerarán varias aplicaciones prácticas de este problema.

5.1. Caso práctico Nº 1: Distribución de productos y maquinarias

En el trabajo realizado por Puchades Cortés, Mula Bru y Rodríguez Villalobos (2008), se presenta a la empresa Semacaf Máquinas de Café S.L. como pyme perteneciente al sector de la distribución automática, cuya actividad empresarial es la compra y distribución de máquinas vending y de agua refrigerada. Esta actividad pone a disposición del consumidor una amplia gama de productos a través de un nuevo modelo de distribución. La empresa objeto de estudio distribuye sus máquinas principalmente en entornos laborables, centros públicos y privados y en empresas dedicadas al sector servicios. Aquí, los autores plantean una nueva forma de asignación de rutas, basado en métodos y algoritmos que resuelven problemas de enrutamiento de vehículos (VRP) para minimizar la distancia total recorrida en cada viaje, la duración de cada una y, por ende, el costo total de distribución.

Un problema de rutas de vehículos (VRP, por sus siglas en inglés) consiste en determinar las rutas de un conjunto de vehículos que deben iniciar un recorrido (y finalizarlo) en uno o varios puntos de partida, como por ejemplo almacenes, para atender la demanda de servicio de un conjunto disperso de clientes sobre una red. Su objetivo es obtener una solución que optimice todas las rutas de todos los vehículos que se vean involucrados.

5.2. Caso práctico Nº 2: Sistema de transporte de autobuses

En este caso, Luna López (2015) presenta el estudio del problema de ruta de vehículos con el fin de aplicarlo al diseño de un sistema de transporte en autobuses, en donde se deberán tomar decisiones como la adecuada localización de las diferentes paradas y las rutas que seguirán cada autobús. La meta final de esta investigación es la de encontrar el diseño de sistema de transporte que logre minimizar la distancia, el tiempo y el costo de los recorridos de cada autobús con la intención adicional de evitar la congestión de rutas y la disconformidades de los usuarios ante un mal servicio.

5.3. Caso práctico Nº 3: reconfiguración de maquinarias

En algunas ocasiones, un taller de manufactura puede contar con una sola máquina en la cual se pueden procesar diferentes tareas, una a la vez. Para esto, la máquina debe ser reconfigurada cada vez que se desea usarlo para una tarea diferente. De

manera que una vez que una tarea ha sido terminada, es necesario preparar la máquina para procesar una nueva tarea, invirtiendo un determinado tiempo, considerado ocioso o perdido, que dependerá de la tarea procesada y la tarea a procesar.

González Velarde y Ríos Mercado han estudiado este caso y lograron plantear una analogía con el problema del viajante.

"Cada tarea puede ser vista como una de las ciudades a visitar, y el tiempo necesario para cambiar la configuración de la máquina corresponde a la distancia que hay entre una ciudad y otra. Encontrar la manera de ordenar las tareas para minimizar el tiempo total de preparación es equivalente a diseñar la ruta, esto es, el orden en el cual se deben de visitar las ciudades para minimizar la distancia total requerida."(González Velarde y Ríos Mercado, 1999, p. 21-22)

5.4. Caso práctico Nº 4: Placa de Circuitos Impresos (PCB)

En esta sección, se expone un caso planteado por Minetti (2000) en donde se desarrolla la siguiente situación:

"Un brazo robotico introduce *n* partes en puntos específicos de un PCB. Cada una de las partes pertenece a un cierto tipo *K* de partes. En un receptáculo se almacenan todas las de un determinado tipo. Los *K* recipientes se ubican a lo largo de uno de los laterales del PCB. Además, se supone que el brazo del robot sólo puede hacer movimientos secuenciales en dirección horizontal o vertical, entonces el tiempo de viaje del brazo del robot de una ubicación a otra es proporcional a la distancia lineal entre los dos puntos. El problema del tour del robot es encontrar una secuencia óptima de inserciones de las partes en el PCB, dadas las posiciones de los *n* puntos de inserción y las ubicaciones de los *K* receptáculos de tal manera de disminuir el tiempo de armado de un PCB."

5.5. Caso práctico Nº 5: Redes y telecomunicaciones

Otra alternativa que se presenta es la toma de decisiones durante el diseño de una red de telecomunicaciones. En el caso de decidir sobre la implementación de una red con una tipología de red denominada anillo en la cual conexiones punto a punto son necesarias. En este caso el objetivo es minimizar la distancia de fibra óptica entre los nodos involucrados y así abaratar los costos. En este tipo de redes se implica un conjunto de enlaces punto a punto por los cuales se transmite un patrón de bits especial, denominados tokens, cuando todas las estaciones están inactivas. En el momento en que algún nodo requiera transmitir toma el token y lo retira del anillo antes de emitir. El mensaje que envía tendrá que pasar por cada uno de los demás nodos solo una vez y regresar al lugar de origen. Una vez que una estación termine de transmitir , esta debe regenerar el token.

5.6. Otros casos prácticos

Así como los casos anteriores demostraron la asociación de problemas cotidianos con el problema del viajante de comercio, existen muchos diversos ejemplos que se adaptan a este tipo de problemáticas con el fin de poder encontrar una solución de manera más fácil, aplicando los conocimientos ya estudiados por la comunidad científica. En el trabajo aportado por Pérez Rave y Jaramillo Álvarez (2012), se encuentra una lista de los artículos publicados en un rango de 10 años con la temática relacionada al problema TSP. Entre estos artículos, se encuentran algunos relacionados a la termodinámica, biofísica del cerebro, fragmentación urbana, entre otros tantos.

6. Conclusiones

En este trabajo práctico, pudimos comparar los resultados de dos técnicas diferentes para la resolución del problema del viajante.

En principio, la utilización de un algoritmo Voraz o Greedy utilizado como técnica heurística, nos permitió obtener resultados casi instantáneos y satisfactoriamente buenos. Por otro lado, las 200 generaciones utilizadas para la corrida del algoritmo genético nos permitieron tener un resultado un poco peor que la técnica heurística, incluso partiendo desde una misma ciudad.

Esto nos permite concluir que si se buscan resultados de manera inmediata, una técnica sencilla de programar como la heurística mencionada, nos podría dar una buena solución. Pero, en caso de que se busquen mejores soluciones aún, y la inmediatez no sea prioridad, el Algoritmo Genético podría brindarnos mejores respuestas si ajustamos correctamente sus parámetros.

Como demostración de la aserción anterior, se dejó correr el algoritmo por 1 hora exacta. (Esto se logró fácilmente modificando la condición de finalización del ciclo While) y se obtuvo un resultado con menor distancia que el mejor resultado posible con la técnica heurística. La ruta obtenida fue la que se detalla a continuación:

1. Ushuaia
2. Rawson
3. Viedma
4. La Plata
5. Cdad. de Bs. As.
6. Paraná
7. Santa Fe
8. Resistencia
9. Corrientes
10. Posadas
11. Formosa
12. S.S. de Jujuy
13. Salta
14. S.M. de Tucumán
15. Sgo. Del Estero .F.d.V.d. Catamarca
16. La Rioja
17. San Juan
18. Mendoza
19. San Luis
20. Córdoba
21. Santa Rosa
22. Neuquén
23. Río Gallegos
24. Ushuaia

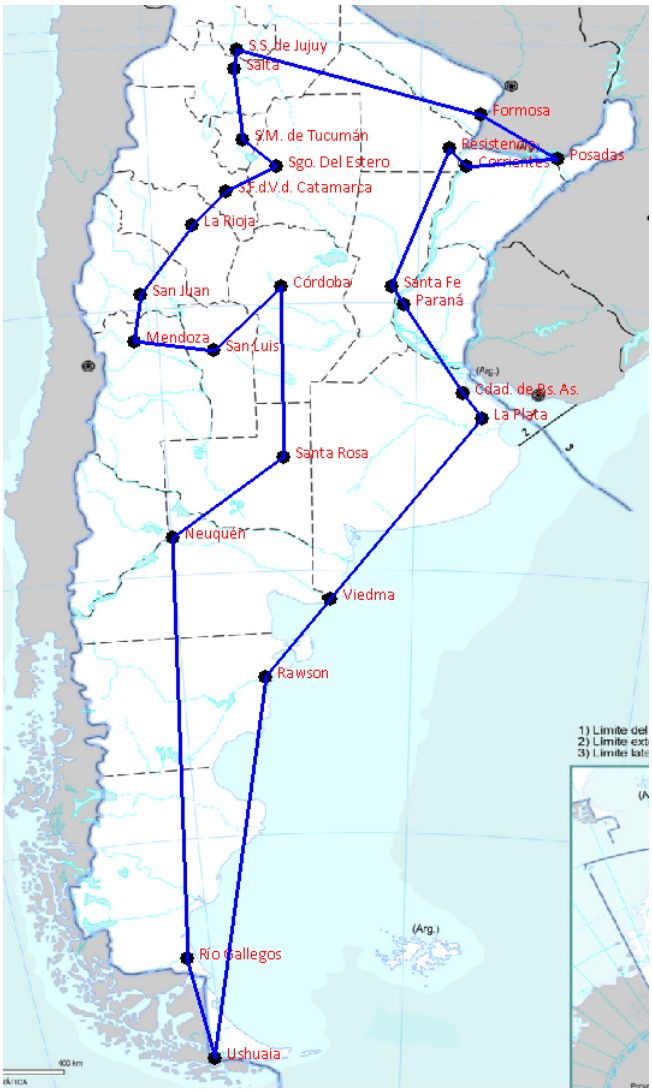

Figura 9: Visualización del recorrido obtenido por el Algoritmo Genético tras una hora de ejecución

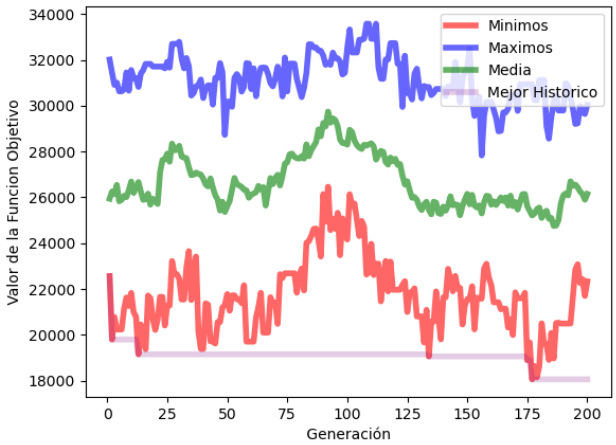

Figura 10: Ejecución del algoritmo genético sin elitismo

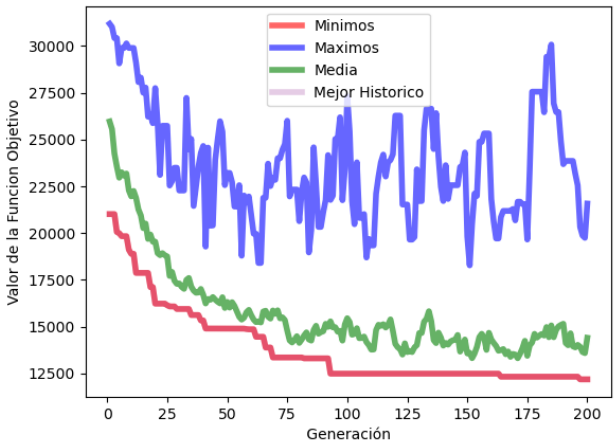

Figura 11: Ejecución del algoritmo genético con elitismo

Cantidad de Km recorridos: 9157

En esta ejecución, el algoritmo genético es capaz de superar al mejor recorrido posible obtenido utilizando la técnica heurística. Se puede observar que a pesar de que el camino parte de provincias diferentes, el recorrido es igual excepto en como se recorren las provincias más al sur del país. En el caso del algoritmo greedy, al estar en Rawson pasa a Rio Gallegos para luego bajar a Ushuaia y volver a subir a Neuquen. El algoritmo genético fue capaz de mejorar esa parte del recorrido, dado que se partió de Ushuaia hacia Rawson y se terminó el recorrido visitando Neuquen, Rio Gallegos y Ushuaia, ahorrando 178 kilómetros.

Además, queremos destacar la importancia de aplicar elitismo en esta solución por Algoritmos Genéticos, dado que en todas las ejecuciones que realizamos sin aplicar elitismo se observaron resultados significativamente peores. Como ejemplo, se dejan dos gráficas que muestran el avance de los valores de la función objetivo en dos ejecuciones distintas utilizando algoritmos genéticos. (ver fig. 10 y 11.)

Puede observarse que en tan solo 200 generaciones, el algoritmo genético con elitismo obtiene una ventaja de más de 5500 km. Si bien estas son dos corridas particulares del algoritmo genético, este fenómeno se repitió en todas las ejecuciones con los diferentes parámetros utilizados en el análisis de sensibilidad. Por lo que concluimos y recomendamos como absoluta necesidad la aplicación de elitismo para obtener mejores resultados en este problema en particular.

7. Bibliografía

- Puchades Cortés, V., Mula Bru, J. y Rodríguez Villalobos, A. (2008), Aplicación de la teoría de grafos para mejorar la planificación de rutas de trabajo de una empresa del sector de la distribución automática, España: Universidad Pablo de Olavide.

  Disponible es: rio.upo.es/xmlui/handle/10433/3597

- Luna López, L. C. (2015), Localización de parads y diseño óptimo de rutas para transporte de personal, México: Universidad Autónoma de Nuevo León. Extraído de: eprints.uanl.mx/9541/1/1080214944.pdf

- González Velarde, J. L. y Ríos Mercado, R. Z. (1999), Investigación de operaciones en acción: aplicación del TSP en problemas de manufactura y logística. Ingenierías, 2(4), p. 18-23.

- Minetti, G. F. (2000), Una solución de computación evolutiva para el TSP, su posible aplicación en las organizaciones, Argentina: UNLP.

  Disponible en: http://sedici.unlp.edu.ar/handle/10915/4059

- Pérez Rave, J. I. y Jaramillo Álvarez, G. P. (2012), Espacio literario relevante sobre el problema del vendedor viajero (TSP): contenido, clasificación, métodos y campos de inspiración, Colombia: Universidad de Antioquía.

  Disponible en:
  https://doi.org/10.1590/S0103-65132013005000003

8. Anexo

Todo el código utilizado puede obtenerse a través del siguiente link: github.com/danilobassi8/algoritmos-geneticos/tree/master/TP3 .
Dentro de la carpeta *Resolucion* podrán encontrarse distintos archivos con extencion .py. Solo será necesario ejecutar el archivo *main.py* dado que los demás scripts son funciones de pantalla y de soporte que se quisieron apartar del script principal.