Resolución del Problema del Viajero Frecuente (TSP) mediante Algoritmos Genéticos

Carolina Rodríguez López

28 de marzo, 2025

1 Objetivo

El presente documento tiene como objetivo describir la implementación de un Algoritmo Genético (AG) para resolver el Problema del Viajero Frecuente (TSP, por sus siglas en inglés). Se detallan los conceptos teóricos, las técnicas empleadas y los resultados obtenidos.

2 Introducción

El Problema del Viajero Frecuente (TSP) es un problema de optimización combinatoria en el que se busca encontrar la ruta más corta que permita visitar un conjunto de ciudades y regresar al punto de origen. Dado que la cantidad de permutaciones crece factorialmente con el número de ciudades, los métodos exactos se vuelven inviables para grandes instancias del problema. En este contexto, los Algoritmos Genéticos se presentan como una alternativa eficiente.

3 Marco Teórico

3.1 Algoritmo Genético

Un Algoritmo Genético es una técnica de optimización inspirada en la evolución biológica. Su funcionamiento se basa en los siguientes operadores: selección, cruza y mutación.

3.2 Selección

Dos métodos de selección fueron empleados en este trabajo:

- **Tournament**: Se forman parejas entre el individuo más apto con el menos apto, el segundo más apto con el segundo menos apto, y así sucesivamente.
- **Ranking**: Se ordenan los individuos por aptitud y se forman parejas entre los primeros dos, luego el tercero con el cuarto, etc.

3.3 Cruza

Se implementaron los siguientes métodos de cruce:

- **PBx (Position-Based Crossover)**: Se seleccionan posiciones aleatorias fijas de un padre y se rellenan con elementos del otro padre. PBX es un operador que trata de garantizar la diversidad en la recombinación cuidando al mismo tiempo de preservar la posición.

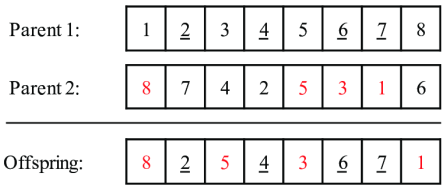

Figure 1: Ejemplo de un operador de cruce de posiciones (PBX)

- **PMx (Partially Mapped Crossover)**: Se intercambian segmentos de ambos padres y se corrigen duplicados mediante un mapeo parcial.

  En el caso del PMX, se empieza seleccionando al azar dos puntos de cruce. Estos puntos dividen los cromosomas de los padres en tres segmentos: Segmentación izquierda, media y derecha.

  En primer lugar, la copia se realiza en la parte intermedia entre los primeros progenitores de la descendencia. De este modo, se evita que determinados genes cambien con la descendencia.

  Para cada descendiente, tendremos que identificar los genes del lado izquierdo y derecho que necesitan ser reemplazados. Además, estableceremos una relación de mapeo para determinar su colocación, que debería evitar la redundancia de genes y sustituir el gen del primer progenitor por su correspondiente en el segundo progenitor.

  Basándonos en la relación, encontraremos y reemplazaremos los genes que se repiten en cada cromosoma.

  Para dar una idea clara de cómo funciona PMX, supongamos que tenemos las siguientes dos soluciones parentales P1 [1, 2, 3, 4, 5, 6, 7, 8, 9] y P2 [5, 4, 6, 9, 2, 1, 7, 8, 3].

  Seleccionaremos puntos de cruce en las posiciones 3 a 6 como sigue:

  Intercambiando los genes de esos dos progenitores, se crearán dos nuevos descendientes de la siguiente manera:

Determinemos la relación cartográfica de la siguiente manera:
Esto conduce a una descendencia que contiene algunos genes de los padres en su orden original, y esto mantiene un segmento medio con su identidad relativa sin duplicaciones como sigue:

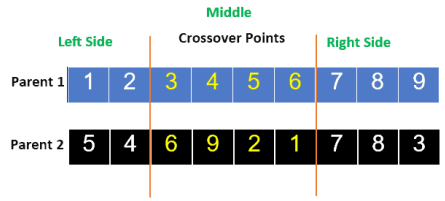

Figure 2: Ejemplo PMx (1)

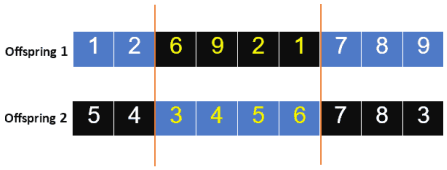

Figure 3: Ejemplo PMx (2)

3.4 Mutación

Se consideraron dos métodos de mutación:

- **Scramble**: Se eligen dos cortes aleatorios, se permutan los valores entre estos y se reintegran en una posición aleatoria.
- **Mutación Inverse**: En la mutación de inversión, se selecciona un subconjunto de genes como en la mutación scramble, pero en lugar de barajar el subconjunto, simplemente se invierte la cadena completa en el subconjunto.

3.5 Elitismo por Árbol Genealógico

El elitismo garantiza que los mejores individuos se conserven en cada generación. En este caso, se utilizó el elitismo por árbol genealógico, el cual mantiene un seguimiento de los individuos más aptos y sus descendientes.

4 Materiales y Métodos

Se desarrolló el algoritmo en Python, utilizando la biblioteca NumPy para las operaciones matriciales y Pandas para la manipulación de datos.

El dataset contiene una matriz de distancias entre 18 ciudades en kilómetros. La primera columna y la primera fila enumeran los nombres de las ciudades, y los valores numéricos representan las distancias geodésicas entre ellas.

Para la construcción del conjunto de datos, se realizó un diccionario con las coordenadas de cada ciudad (Tabla 1) y se implementó la función 'geodesic' de la librería geopy. Finalmente, la información fue trasladada a un archivo Excel.

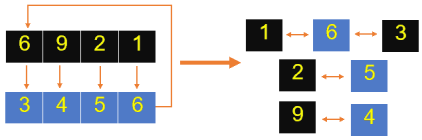

Figure 4: Ejemplo PMx (3)

Figure 5: Ejemplo PMx (4)

4.1 Pseudocódigo del Algoritmo

Algorithm 1 Algoritmo Genético para el TSP

```
Inicializar población aleatoria
for cada generación do
    Evaluar aptitud de cada individuo
    Aplicar selección (Tournament o Ranking)
    Aplicar cruce (PBx o PMx) para generar descendencia
    Aplicar mutación (Scramble o Heurística)
    Aplicar elitismo por árbol genealógico
    Actualizar población
end for
Retornar mejor solución
```

5 Resultados

Los resultados obtenidos muestran que el Algoritmo Genético es capaz de encontrar rutas cercanas al óptimo en tiempos razonables. La implementación permitió evaluar diferentes combinaciones de operadores y analizar su impacto en la convergencia del algoritmo.

A todas las versiones del código genético se les aplican dos criterios de paro:

1. Un máximo de 30 generaciones estagnadas, es decir, iteraciones en las que el promedio de aptitud de la población tiene un cambio menor a 0.001

2. Un máximo de 200 iteraciones totales

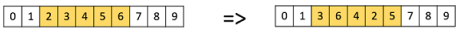

Figure 6: Mutación Scramble

Ciudad	**Latitud**	**Longitud**
Ciudad de México	19.4326	-99.1332
Quito	-0.1807	-78.4678
Miami	25.7617	-80.1918
San Salvador	13.6929	-89.2182
Mendoza	-32.8895	-68.8458
Guadalajara	20.6597	-103.3496
Mérida	20.9674	-89.5926
Washington D.C.	38.9072	-77.0369
Monterrey	25.6866	-100.3161
Managua	12.1150	-86.2362
Caracas	10.4806	-66.9036
Boston	42.3601	-71.0589
Buenos Aires	-34.6037	-58.3816
Nueva York	40.7128	-74.0060
Ciudad de Panamá	8.9824	-79.5199
Brasilia	-15.8267	-47.9218
Montevideo	-34.9011	-56.1645
Bogotá	4.7110	-74.0721

Table 1: Coordenadas geográficas de diversas ciudades

Además, se trabaja con un tamaño de población de 200 individuos y la misma población inicial para tener un punto de comparación similar.

En el primer algoritmo, se implementa **selección rank, cruza PMx y mutación scramble al 10% de los peores individuos**. Es aplicada la mutación a partir de un estancamiento de 5 iteraciones o más. El algoritmo tuvo convergencia en la iteración número 68, encontrando una ruta de **26,132.2499km** totales.

Esta ruta fue [5, 0, 7, 13, 11, 2, 6, 3, 1, 4, 12, 16, 15, 10, 17, 14, 9, 8]: **'Guadalajara', 'Ciudad de México', 'Washington D.C.', 'Nueva York', 'Boston', 'Miami', 'Mérida', 'San Salvador', 'Quito', 'Mendoza', 'Buenos Aires', 'Montevideo', 'Brasilia', 'Caracas', 'Bogotá', 'Ciudad de Panamá', 'Managua', 'Monterrey' y de regreso a 'Guadalajara'.**

En el segundo algoritmo, se implementa **selección por torneo, cruza PBx y mutación scramble al 10% de los peores individuos**. Se tuvo convergencia en 131 iteraciones para una ruta de **23,799.53 km**.

Esta ruta fue [12, 16, 15, 10, 11, 13, 7, 2, 6, 8, 5, 0, 3, 9, 14, 17, 1, 4]: **'Buenos Aires', 'Montevideo', 'Brasilia', 'Caracas', 'Boston', 'Nueva York',**

	Ciudad de Méx	Quito	Miami	an Salvado	Mendoza	Guadalajara	Mérida	shington D	Monterrey	Managua	Caracas	Boston	uenos Aire	Nueva York	dad de Pan	Brasilia	Montevide	Bogotá
dad de Mé	0	3129.83	2067.24	1233.65	6626.35	461.53	1011.33	3029.79	703.15	1600.53	3598.58	3664.18	7372.63	3357.7	2408.55	6828.88	7535.67	3169.42
Quito	3129.83	0	2876.39	1938.94	3758.5	3556.97	2634.08	4330.65	3702.22	1608.07	1740.42	4769.88	4346.62	4551.13	1019.98	3775.49	4488.76	729.04
Miami	2067.24	2876.39	0	1635.72	6601.43	2433.25	1097.56	1487.43	2017.52	1638.51	2195.97	2022.02	7066.44	1754.35	1858.49	5778.71	7176.18	2419.27
an Salvado	1233.65	1938.94	1635.72	0	5591.9	1688.21	806.08	3040.13	1763.41	367.69	2454.16	3626.74	6261.04	3340.41	1179.62	5595.88	6411.72	1936.74
Mendoza	6626.35	3758.5	6601.43	5591.9	0	6969.9	6355.34	7993.68	7290.44	5312.39	4803.94	8335.2	987.57	8166.42	4770.19	2831.46	1193.15	4197.89
Guadalajara	461.53	3556.97	2433.25	1688.21	6969.9	0	1432.06	3229.14	637.42	2055.85	4057.37	3858.37	7747.76	3554.49	2866.17	7278.31	7916.24	3624.94
Mérida	1011.33	2634.08	1097.56	806.08	6355.34	1432.06	0	2323.57	1214.24	1042.96	2689.64	2937.95	6980.7	2639.4	1710.9	6106.37	7121.6	2459.78
shington D	3029.79	4330.65	1487.43	3040.13	7993.68	3229.14	2323.57	0	2625.01	3104.8	3306.49	634.38	8362.93	327.91	3324.17	6775.08	8450	3799.11
Monterrey	703.15	3702.22	2017.52	1763.41	7290.44	637.42	1214.24	2625.01	0	2107.67	3899.95	3247.93	8005.98	2946.9	2873.14	7300.91	8162.25	3636.1
Managua	1600.53	1608.07	1638.51	367.69	5312.39	2055.85	1042.96	3104.8	2107.67	0	2117.93	3660.08	5949.37	3387.47	812.57	5230.52	6094.87	1569.19
Caracas	3598.58	1740.42	2195.97	2454.16	4803.94	4057.37	2689.64	3306.49	3899.95	2117.93	0	3555.76	5070.14	3421.84	1394.12	3583.52	5148.77	1016.03
Boston	3664.18	4769.88	2022.02	3626.74	8335.2	3858.37	2937.95	634.38	3247.93	3660.08	3555.76	0	8619.76	306.49	3789.97	6865.45	8688.91	4181.39
uenos Aire	7372.63	4346.62	7066.44	6261.04	987.57	7747.76	6980.7	8362.93	8005.98	5949.37	5070.14	8619.76	0	8491.79	5313.8	2328.29	205.66	4652.6
Nueva York	3357.7	4551.13	1754.35	3340.41	8166.42	3554.49	2639.4	327.91	2946.9	3387.47	3421.84	306.49	8491.79	0	3557.42	6815.07	8569.61	3987.76
dad de Pan	2408.55	1019.98	1858.49	1179.62	4770.19	2866.17	1710.9	3324.17	2873.14	812.57	1394.12	3789.97	5313.8	3557.42	0	4434.06	5444.05	765.21
Brasilia	6828.88	3775.49	5778.71	5595.88	2831.46	7278.31	6106.37	6775.08	7300.91	5230.52	3583.52	6865.45	2328.29	6815.07	4434.06	0	2267.78	3668.87
Montevide	7535.67	4488.76	7176.18	6411.72	1193.15	7916.24	7121.6	8450	8162.25	6094.87	5148.77	8688.91	205.66	8569.61	5444.05	2267.78	0	4769.3
Bogotá	3169.42	729.04	2419.27	1936.74	4197.89	3624.94	2459.78	3799.11	3636.1	1569.19	1016.03	4181.39	4652.6	3987.76	765.21	3668.87	4769.3	0

Figure 7: Distancias entre ciudades

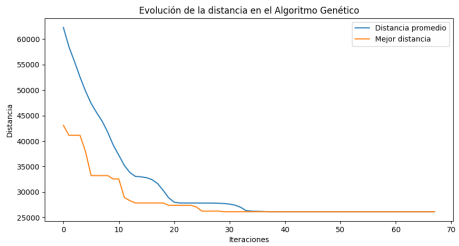

Figure 8: Evolución Distancia en AG 1: selección rank, cruza PMx y mutación scramble al 10% de los peores individuos

'Washington D.C.', 'Miami', 'Mérida', 'Monterrey', 'Guadalajara', 'Ciudad de México', 'San Salvador', 'Managua', 'Ciudad de Panamá', 'Bogotá', 'Quito', 'Mendoza' y de regreso a 'Buenos Aires'.

Aproximadamente, el tiempo de ejecución de los programas con mutación scramble es de 4 minutos.

En el tercer algoritmo, se usa **selección por rank, cruza PMx y se aplica mutación inversa al 10% de los peores individuos**. Se obtuvo una convergencia en un mayor número de iteraciones que en la mutación scramble, un total de 139 para tener una ruta mínima de **25,572.619 km.**

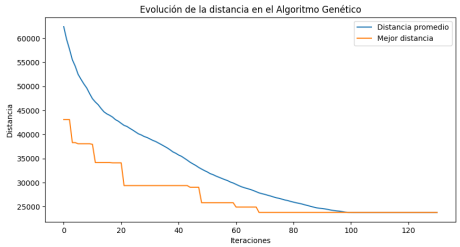

Figure 9: Evolución Distancia en AG 2: selección por torneo, cruza PBx y mutación scramble al 10% de los peores individuos

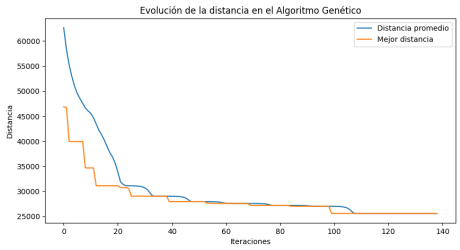

Figure 10: Evolución Distancia en AG 3: selección por rank, cruza PMx y se aplica mutación inversa al 10% de los peores individuos

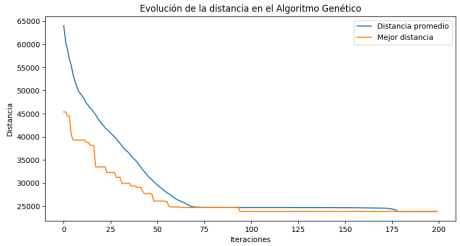

Figure 11: Evolución Distancia en AG 4: selección por torneo, cruza PBx y se aplica mutación inversa al 10% de los peores individuos

Esta ruta fue [11, 13, 7, 2, 6, 5, 8, 0, 3, 9, 17, 14, 1, 4, 16, 12, 15, 10]: **'Boston', 'Nueva York', 'Washington D.C.', 'Miami', 'Mérida', 'Guadalajara', 'Monterrey', 'Ciudad de México', 'San Salvador', 'Managua', 'Bogotá', 'Ciudad de Panamá', 'Quito', 'Mendoza', 'Montevideo', 'Buenos Aires', 'Brasilia', 'Caracas' y de regreso a 'Boston'.**

En el tercer algoritmo, se usa **selección por torneo, cruza PBx y se aplica mutación inversa al 10% de los peores individuos** para tener una ruta mínima de **23,871.299 km.** en 200 iteraciones.

La ruta mínima encontrada fue: [7, 13, 11, 2, 10, 15, 16, 12, 4, 1, 17, 14, 9, 3, 6, 0, 5, 8]: **'Washington D.C.', 'Nueva York', 'Boston', 'Miami', 'Caracas', 'Brasilia', 'Montevideo', 'Buenos Aires', 'Mendoza', 'Quito', 'Bogotá', 'Ciudad de Panamá', 'Managua', 'San Salvador', 'Mérida', 'Ciudad de México', 'Guadalajara', 'Monterrey' y de regreso a 'Washington D.C.'.**

A continuación se reportan algunas ventajas y desventajas de los métodos y técnicas implementadas en este trabajo:

Selección Rank

- **Ventajas:** Reduce la probabilidad de convergencia prematura. Mejora la diversidad genética.

- **Desventajas:** Puede ser más lenta en problemas grandes. Menos efectiva si hay individuos extremadamente superiores.

Selección por Torneo

- **Ventajas:** Fácil de implementar y eficiente. Adecuada para entornos paralelos.
- **Desventajas:** Puede introducir sesgo si el tamaño del torneo es grande. Menor diversidad genética en torneos pequeños.

Cruza PBx (Position-Based Crossover)

- **Ventajas:** Mantiene características específicas de los padres. Útil en problemas de ordenamiento.
- **Desventajas:** Puede generar soluciones poco diversas. Dependencia de la posición inicial.

Cruza PMx (Partially Mapped Crossover)

- **Ventajas:** Preserva información de los padres. Útil en problemas de asignación y enrutamiento.
- **Desventajas:** Puede generar soluciones inválidas sin manejo adicional. Incremento en la complejidad computacional.

Mutación Inversa

- **Ventajas:** Puede mantener características buenas de una solución. Es más rápida y sencilla de implementar que otras mutaciones más complejas.
- **Desventajas:** Si los puntos seleccionados para la inversión están muy próximos, el efecto será mínimo, lo que podría reducir la efectividad de la mutación. Si se aplica con demasiada frecuencia, puede contribuir al estancamiento del algoritmo al no introducir cambios suficientemente disruptivos.

Mutación Scramble

- **Ventajas:** Incrementa la diversidad genética. Útil para evitar convergencia prematura.
- **Desventajas:** Puede destruir buenas soluciones parcialmente. No siempre mejora la solución.

6 Conclusiones

El uso de Algoritmos Genéticos para el TSP resulta efectivo al reducir el tiempo de cómputo respecto a métodos exactos. La selección, cruce y mutación juegan un papel crucial en la calidad de la solución, destacando el elitismo por árbol genealógico como una técnica que preserva la diversidad y calidad de los individuos a lo largo de las generaciones.

Aunque se obtuvieron buenos resultados con todos los algoritmos, la selección rank junto con la cruza PMx resultaron ser una buena combinación para disminuir el rango de iteraciones y con ello, la distancia promedio bajó a una rapidez considerable generación tras generación. Con la selección rank, se preservaron en mayor medida los genes de los individuos más aptos o las rutas más cortas; Mientras que con la selección por torneo, se perdía información de buenas rutas al cruzar los mejores individuos con los peores.

La mutación fue de gran ayuda para escapar de óptimos locales al aumentar la exploración y la diversidad de la población. Además, se mejoró el tiempo de ejecución del algoritmo al implementar la mutación una vez que se tiene una estagnación a través de 5 generaciones, en lugar de implementarla desde la primera generación.