Tecnológico de Monterrey, Campus Monterrey

Situación Problema

Calles abarrotadas: las ciudades enfrentan un aumento en las entregas de las compras en línea

Andrea Axel Hernández Galgani - A00835225

Andrés Morones Navarro - A00833287

Annette Pamela Ruiz Abreu - A01423595

Luis Maximiliano López Ramírez - A08833321

Jorge Raúl Rocha López - A01740816

MA2015.101 - Diseño de algoritmos matemáticos bioinspirados (Gpo 101)

Fernando Elizalde Ramírez

06 de septiembre de 2023

Índice

Calles abarrotadas: las ciudades enfrentan un aumento en las entregas de las compras en línea

En el mundo actual, la optimización de rutas y la planificación eficiente de viajes se han convertido en desafíos críticos en una variedad de aplicaciones prácticas, desde la logística empresarial hasta la planificación de circuitos. Uno de los problemas clásicos que aborda esta problemática es el problema del agente viajero (TSP). El TSP se refiere a la tarea de encontrar la ruta más corta que visite un conjunto de ciudades y regrese a la ciudad de origen, minimizando la distancia total recorrida.

El problema del agente viajero (TSP) ha sido un desafío fundamental en la investigación de algoritmos y optimización durante décadas, y su relevancia sigue en aumento en un mundo cada vez más interconectado y globalizado. Con el auge del comercio electrónico, la gestión de flotas de vehículos, la planificación de rutas de transporte público y la optimización de redes de comunicación, la resolución eficiente del TSP se ha convertido en un objetivo crítico para mejorar la eficiencia y reducir los costos en una amplia variedad de industrias. Este trabajo se sumerge en la formulación matemática y la implementación de algoritmos genéticos y algoritmos de hormigas para solucionar este problema. (Del Estado De Hidalgo, s. f.)

Problema

El problema del agente viajero (TSP) es un problema combinatorio que involucra un conjunto de ciudades y las distancias entre ellas. El reto consiste en encontrar la ruta más corta que conecte todas las ciudades exactamente una vez y luego regrese al punto de partida, minimizando así la distancia total recorrida. Este problema presenta una complejidad computacional sustancial debido a la explosión combinatoria en el número de posibles soluciones. Conforme aumenta el número de ciudades, la resolución exhaustiva se torna

inviable, lo que exige el desarrollo de estrategias heurísticas y algoritmos de optimización para encontrar soluciones aproximadas en un tiempo razonable.

Justificación

La capacidad de encontrar rutas óptimas o cercanas a la óptima tiene un impacto significativo en la eficiencia de las operaciones logísticas, la reducción de costos de transporte y el ahorro de tiempo. Además, el TSP sirve como un punto de referencia y un banco de pruebas para el desarrollo y la evaluación de algoritmos de optimización y heurísticas.

Objetivo

El objetivo principal de este trabajo es abordar el problema del agente viajero (TSP) mediante la formulación de una representación matemática precisa, así como el desarrollo y la implementación de dos algoritmos avanzados de optimización: un algoritmo genético y un algoritmo de hormigas. Estos enfoques permitirán encontrar soluciones aproximadas para el TSP y evaluar su desempeño en términos de calidad de la solución y eficiencia computacional. A través de la comparación de estos algoritmos, se buscará identificar cuál de ellos se adapta mejor al problema y puede proporcionar resultados óptimos o cercanos a la óptima en diferentes contextos y tamaños de instancias del TSP.

Antecedentes

El problema del agente viajero (TSP por sus siglas en inglés) es un problema combinatorial catalogado como NP-Completo en el área de computación e investigación de operaciones; es decir, el número de posibles soluciones crece factorialmente con el número de ciudades, lo que hace que sea impracticable resolverlo exhaustivamente para grandes instancias. El reto radica en encontrar el camino más corto para que un vendedor recorra, dado un punto de inicio, una cantidad de ciudades (que son los nodos) y, opcionalmente, un punto final. (Daniells, s.f.)

Con base en lo anterior es posible adaptarlo a otros problemas que surgen en la vida profesional, tales como aumentar la rentabilidad de una empresa al disminuir el tiempo de traslado, así como las emisiones de carbono. (Lee, s.f.)

Actualmente, las tres opciones más comunes para solucionar el problema son:

1. **El método de fuerza bruta:** calcula y compara todas las posibles permutaciones de rutas o caminos para así determinar la solución más corta. (Tobin, 2023)
2. **El algoritmo de ramificación y acotación:** comienza creando una ruta inicial, generalmente desde el punto de inicio hasta el primer nodo en un conjunto de ciudades. Luego, explora sistemáticamente diferentes permutaciones para extender la ruta un nodo a la vez. Cada vez que se agrega un nuevo nodo, el algoritmo calcula la longitud de la ruta actual y la compara con la ruta óptima encontrada hasta ese momento. Si la ruta actual ya es más larga que la ruta óptima, el algoritmo "acota" o elimina esa rama de exploración, ya que no conduciría a una solución óptima. (*Traveling Salesperson problem using branch and bound*, s.f.)

3. **Vecino más Cercano:** se empieza en un punto de inicio seleccionado al azar. Desde ahí, se encuentra el nodo no visitado más cercano y se añade a la secuencia. Luego, se avanza al siguiente nodo y se repite el proceso de encontrar el nodo no visitado más cercano hasta que todos los nodos estén incluidos en el recorrido. Finalmente, se regresa a la ciudad de inicio para completar el ciclo. Es importante mencionar que este algoritmo no suele encontrar la mejor solución posible. (Kirkpatrick, 2020)

Datos

Para la modelación y resolución se usaron direcciones de la ciudad de Guadalajara. El archivo con las ubicaciones de entrega contiene 149 puntos de entrega diferentes. Cada fila de la tabla representa un nodo y la tabla tiene 4 columnas: colonia, código postal, calle, número de casa. Esta tabla fue valiosa para identificar y rastrear la geolocalización de cada colonia y proporciona información esencial para la gestión logística. Al procesar los datos y utlizar la librería geopy.geocoders de Python, se obtuvieron las latitudes y longitudes de cada dirección.

	colonia	Calle	# de casa	Código postal	direccion	latitud	longitud
0	U H FOVISSSTE ESTADI	LISBOA	28	44307	28, LISBOA, Guadalajara, Jalisco, 44307, México	20.7108	-103.331
1	INDEPENDENCIA	ROGELIO BACON	2280	44290	2280, ROGELIO BACON, Guadalajara, Jalisco, 44290, México	20.69718	-103.335
…	…	…	…	…	…	…	…
148	LA ESPERANZA	IGUALDAD	186	44300	186, IGUALDAD, Guadalajara, Jalisco, 44300, México	20.70727	-103.311

Con la latitud y longitud calculamos la distancia geográfica entre cada ubicación utilizando la fórmula de Haversine, una técnica matemática ampliamente reconocida para estimar distancias en la superficie de una esfera, como la Tierra. Esta fórmula es particularmente útil cuando se trata de medir distancias en coordenadas geográficas, ya que tiene en cuenta la curvatura de la Tierra. (Miguel, 2011) Teniendo la matriz de distancias generamos 10 matrices de distancia con 40 nodos aleatorios y una con 100 nodos.

- Nodos matriz 1: `[146, 8, 109, 123, 147, 3, 52, 118, 125, 71, 41, 133, 83, 19, 63, 92, 11, 107, 35, 90, 97, 72, 67, 116, 44, 77, 34, 61, 112, 96, 1, 60, 49, 137, 93, 80, 140, 115, 111, 120]`
- Nodos matriz 2: `[16, 83, 128, 40, 57, 105, 61, 9, 8, 127, 77, 18, 136, 20, 38, 98, 145, 95, 28, 24, 113, 42, 48, 89, 110, 106, 114, 62, 70, 36, 133, 45, 30, 68, 116, 44, 122, 88, 111, 1]`

- Nodos matriz 3: [139, 11, 84, 81, 62, 20, 67, 114, 103, 40, 99, 126, 61, 134, 69, 133, 123, 127, 16, 42, 125, 118, 102, 34, 107, 138, 89, 137, 143, 112, 25, 106, 9, 0, 12, 23, 38, 27, 116, 124]
- Nodos matriz 4: [35, 117, 111, 128, 114, 87, 66, 108, 84, 133, 39, 60, 82, 8, 50, 147, 122, 6, 29, 75, 63, 74, 22, 11, 140, 27, 73, 41, 36, 92, 102, 49, 89, 46, 20, 109, 53, 24, 78, 118]
- Nodos matriz 5: [133, 108, 114, 142, 53, 68, 93, 29, 1, 104, 97, 51, 17, 22, 14, 89, 65, 92, 116, 49, 72, 119, 75, 129, 127, 67, 103, 113, 87, 70, 123, 122, 21, 138, 43, 106, 144, 16, 78, 4]
- Nodos matriz 6: [83, 82, 18, 101, 95, 72, 47, 61, 43, 78, 24, 142, 13, 56, 145, 107, 49, 146, 112, 4, 100, 51, 86, 147, 48, 133, 144, 12, 148, 9, 93, 40, 30, 88, 42, 7, 137, 130, 89, 22]
- Nodos matriz 7: [101, 112, 147, 89, 15, 37, 113, 120, 115, 44, 90, 108, 87, 143, 131, 117, 99, 49, 24, 110, 39, 10, 47, 48, 145, 109, 92, 96, 12, 13, 42, 79, 57, 148, 45, 25, 144, 77, 19, 88]
- Nodos matriz 8: [49, 125, 41, 43, 103, 119, 84, 9, 44, 45, 148, 107, 39, 131, 100, 24, 101, 30, 60, 27, 108, 79, 74, 136, 25, 88, 133, 121, 62, 3, 40, 98, 58, 130, 147, 102, 55, 73, 90, 17]
- Nodos matriz 9: [48, 142, 135, 6, 19, 102, 8, 4, 110, 86, 45, 12, 91, 83, 121, 43, 52, 32, 74, 51, 63, 98, 139, 107, 141, 7, 1, 31, 14, 17, 3, 111, 46, 13, 96, 148, 117, 116, 85, 47]
- Nodos matriz 10: [8, 102, 117, 125, 93, 18, 35, 67, 64, 45, 23, 3, 39, 71, 20, 134, 78, 36, 26, 137, 30, 73, 85, 0, 147, 66, 132, 109, 130, 37, 15, 14, 54, 107, 38, 6, 95, 29, 122, 76]
- Nodos matriz 11: [118, 44, 36, 7, 103, 39, 115, 45, 86, 129, 120, 11, 125, 99, 42, 70, 54, 126, 133, 90, 144, 142, 97, 38, 81, 33, 14, 117, 13, 85, 135, 12, 20, 9, 32, 8, 2, 93, 111, 56, 122, 23, 26, 17, 116, 109, 79, 102, 139, 138, 49, 61, 55, 131, 98, 48, 137, 18, 101, 22, 53, 3, 43, 63, 76, 66, 15, 124, 106, 91, 87, 1, 83, 75, 128, 104, 72, 29, 107, 30, 41, 96, 136, 140, 146, 123, 21, 100, 5, 113, 145, 143, 58, 127, 119, 77, 141, 114, 94, 47]

Modelación

Formulación Matemática

Para resolver el problema sabemos que:

- El objetivo es minimizar la suma de los costos de viajar entre todas las ciudades
- La variable de decisión será una variable binaria x_{ij}. Si $x_{ij} = 1$ significa que se viajó de la ciudad i a la j.
- Las distancias entre las ciudades son los costos.
- Desde cada ciudad i, se debe viajar exactamente a una ciudad diferente j.
- Hacia cada ciudad j, se debe viajar exactamente desde una ciudad diferente i.
- Debemos tener restricciones de suma que aseguren que desde cada lugar de entrega i, se debe viajar a exactamente una ubicación diferente.
- Debemos tener restricciones de eliminación de subciclos eviten que se formen ciclos pequeños no deseados en la solución.

Minimizar: $\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} * x_{ij}$

Sujeto a:

$$\sum_{j \neq i}^{n} x_{ij} = 1, \ \forall i \neq j$$

$$\sum_{i \neq j}^{n} x_{ij} = 1, \ \forall j \neq i$$

$$u_i - u_j + n x_{ij} \leq n - 1, \ \ 2 \leq i \neq j \leq n$$

$$x_{ij} \in \{0, 1\}, \ \ i, j = 1, \ ..., \ n$$

$$u_i \geq 0, \ \ i = 1, \ ..., \ n$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} t_{ij} * x_{ij} \ \leq t_{limite}$$

Donde:

- n es el número total de ubicaciones o lugares de entrega (nodos).
- c_{ij} representa el costo de viajar desde la ciudad i a la ciudad j.
- x_{ij} es una variable binaria que toma el valor 1 si se viaja desde el nodo i al nodo j, y 0 en caso contrario.
- u_i es una variable auxiliar utilizada para eliminar subciclos en la solución, siendo una variable continua con un valor asociado a cada nodo i.

Algoritmo Genético

Los algoritmos genéticos son una técnica de optimización que se basa en el proceso de selección natural para obtener soluciones a problemas y búsqueda. Este intenta replicar el proceso de cruzamiento, mutación y selección en una población. De modo que los individuos de la población son posibles soluciones al problema en cuestión. Para determinar cuáles son los individuos con mayor probabilidad que tiene de cruzarse, se tiene una función fitness, la cual evalúa de acuerdo a un criterio la calidad de la solución. (Enzyme, 2022).

Existen diferentes criterios que pueden usarse para escoger cuáles son las dos soluciones a cruzarse; sin embargo, el método que se usó, fue que se calcularon las probabilidades que tenía cada individuo con base en la división de 1 entre su puntaje de fitness y se obtuvo un intervalo de probabilidad para cada uno de los individuos y se fue seleccionando un número aleatorio que correspondía a esos rangos.

Variables:

- $nodo\ inicio$: Es el nodo desde que empieza el viaje del TSP y en el cual se va acabar al final de la ruta, ya que se hayan visitado todos los nodos restantes.

- $n_{pob\ inicial}$: Es el número que va a tener la población; es decir dentro de la población van a existir $n_{pob\ inicial}$ soluciones
- $num\ interseccion$: Es en donde se van a cruzar los dos padres, los hijos van a tomar hasta ese número del padre 1 y va a retomar la segunda parte a partir de ese número del padre 2.
- $prob\ mutation$: Es la probabilidad que tienen los hijos a mutar.
- $epochs$: Son las iteraciones que el algoritmo debe cumplir hasta que se acabe.
- $generacion\ num$: Es una lista en donde se guardan cada una de las soluciones de la población.
- $costo\ dist$: Es una lista, la cual contiene la suma de las distancias de las rutas de cada una de las soluciones de la población.
- $fitness$: Es una lista, la cual contiene la puntuación de la función fitness de cada una de las soluciones de la generación.
- $prob$: Es una lista la cual contiene la probabilidad de cada una de las soluciones y se calculo a partir de dividir 1 entre la puntuación fitness del estado.
- $prob_{li}$: Es una lista la cual contiene la probabilidad del límite inferior de cada uno de los estados de la generación.
- $prob_{ls}$: Es una lista, la cual contiene la suma acumulada de prob para cada uno de los estado de la generación.
- $indices_{padres}$: Es una lista, la cual contiene los índices de la generación, los cuales fueron seleccionados para cruzarse.
- $hijos$: Es una lista, la cual contiene el resultado después de cruzarse de las soluciones que fueron seleccionados padres.

- dic_{ord}: Es un diccionario con los índices de la generación y con los costos de distancia, ordenados de mayor a menor.
- $dic_{ord\ hijos}$: Es un diccionario con los índices de los hijos y con los costos de su distancia, ordenados de mayor a menor.

Procedimiento:

1. Se carga en un pandas dataframe la matriz de distancias en una variable. Los nombres de los nodos están guardados en las columnas del dataframe.
2. Se crean dos diccionarios, en los cuales se guardan el índice como la llave del primer diccionario y el nombre del nodo y viceversa.
3. Se crea una lista, la cual tiene adentros listas hasta que satisfagan el número de $n_{pob\ inicial}$ y se crean soluciones random, que empiezan y terminan en el $nodo\ inicio$ sin repetir los nodos.
4. Se crea una lista con ceros y de una longitud de $n_{pob\ inicial}$y se va sumando cada una de las distancias de cada una de las rutas de la generación.
5. Se selecciona dos índices para que sean los padres, esto en base a la función fitness, se calculan los límites de las probabilidades y se generan dos números aleatorios del [0,1] y se selecciona los índices que estén entre esos rangos.
6. Se crean a los dos hijos, para crear al hijo_1 se crea poniendo hasta el $num\ interseccion$ del padre 1 y se le agrega hasta a partir del $num\ interseccion$ del padre 2, para crear el hijo_2 se hace exactamente lo mismo solo que en orden contrario. Como esto crea soluciones infactibles, lo que se hace es que se va iterando a partir de $num\ interseccion$ entre cada uno de los hijos y si es que se repite un número se va iterando entre el padre que le dio la segunda parte del hijo y se cambia por uno que no esté.

7. Se crean números aleatorios para cada uno de los hijos y si es que es menor o igual a la $prob_{mutation}$ es que se muta el hijo. Esto se hace creando índices del 1 al 39 y se cambian dos índices en la solución, del hijo que se decidió mutar.
8. Se reemplazan las poblacion, a partir de que se ordena de mayor a menor los costos o distancias de la generación e igual para los hijos. Si los dos valores que son los más altos de la generación son mayores que el de los hijos, se reemplazan.
9. Por cada iteración se imprime la mejor ruta de la generación, la época, la mejor distancia y el mejor valor de la función fitness.

Algoritmo de Hormigas

El algoritmo de colonia de hormigas es una técnica de optimización inspirada en el comportamiento de las hormigas reales. Funciona mediante la construcción de soluciones probabilísticas, donde las hormigas eligen caminos y depositan feromonas en aristas. Estas feromonas influyen en las decisiones futuras de las hormigas, fomentando la exploración y explotación de soluciones. Con el tiempo, el algoritmo tiende a converger hacia una solución óptima o cercana a óptima debido a la acumulación de feromonas en las rutas de alta calidad. Se utiliza para resolver problemas de optimización combinatoria como el problema del viajante o TSP.

Variables:

- $num\_hormigas$: El número de hormigas que se utilizarán en el algoritmo. Se usaron 1 hormiga por cada nodo de la matriz de distancias.
- $max\_iteraciones$: La cantidad máxima de iteraciones que realizará el algoritmo. Se determinó la cantidad de 1000 iteraciones por tiempos computacionales.
- $alfa$ y $beta$: Parámetros que controlan la importancia relativa de la feromona y la distancia en el cálculo de la probabilidad de selección del siguiente nodo. Se eligieron

los valores de 1 y 3 respectivamente a que fueron con los que mejores resultados se obtuvieron.

- $evaporacion$: La tasa de evaporación de las feromonas en cada iteración. Controla cuánto disminuirán las feromonas con el tiempo. Se definió una evaporación de 0.1.
- $feromona\_inicial$: La cantidad inicial de feromona en todas las aristas. Se usó una feromona inicial de 0.1.
- df: Un DataFrame de pandas que contiene la matriz de distancias entre nodos. Los nombres de los nodos se encuentran en el índice y las columnas.

Procedimiento:

1. Carga una matriz de distancias desde un archivo Excel. Esta matriz representa las distancias entre nodos en un problema TSP. Los nombres de los nodos están en el índice y las columnas del DataFrame.
2. Inicializa una matriz de feromonas con valores iniciales iguales.
3. Comienza un bucle principal que se ejecuta un número máximo de iteraciones ($max\_iteraciones$).
4. En cada iteración del bucle principal, se ejecuta un bucle para cada hormiga ($num\_hormigas$) y se realizan los pasos 5 al 7 por cada hormiga:
5. Se elige un nodo inicial al azar para la hormiga.
6. La hormiga construye un camino visitando nodos uno por uno. En cada paso, la hormiga elige el siguiente nodo basándose en la probabilidad de selección, que depende de la cantidad de feromona y la distancia.

   La probabilidad de que una hormiga elija visitar el nodo j desde el nodo i se calcula utilizando la siguiente fórmula:

$$P_{ij} = \frac{\Gamma_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{k \epsilon N_i} \Gamma_{ik}^{\alpha} \cdot \eta_{ik}^{\beta}}$$

Donde:

- P_{ij} es la probabilidad de ir del nodo i al nodo j.
- τ_{ij} es la cantidad de feromona en la arista que conecta los nodos i y j.
- α *o* $alfa$ es un parámetro que controla la importancia relativa de la feromona.
- η_{ij} es la inversa de la distancia entre los nodos i y j. Puede ser $\frac{1}{d_{ij}}$.
- β *o* $beta$ es un parámetro que controla la importancia relativa de la distancia.
- N_i es el conjunto de nodos no visitados desde el nodo i.

7. Después de que una hormiga completa su camino, se calcula la longitud del camino y se actualizan las feromonas en todas las aristas del camino utilizando la siguiente fórmula:

$$\Gamma_{ij} = (1 - \rho) \cdot \Gamma_{ij} + \sum_{k=1}^{K} \Delta\Gamma_{ij}^{k}$$

Donde:

- τ_{ij} es la cantidad de feromona en la arista que conecta los nodos i y j.
- ρ *o* $evaporacion$ es la tasa de evaporación de las feromonas.
- K es el número de hormigas en la colonia.
- $\Delta\tau_{ij}^{k}$ es la cantidad de feromona depositada por la hormiga k en la arista (i , j).

  Puede calcularse como: $\frac{1}{Longitud\ del\ camino\ de\ la\ hormiga\ k}$.

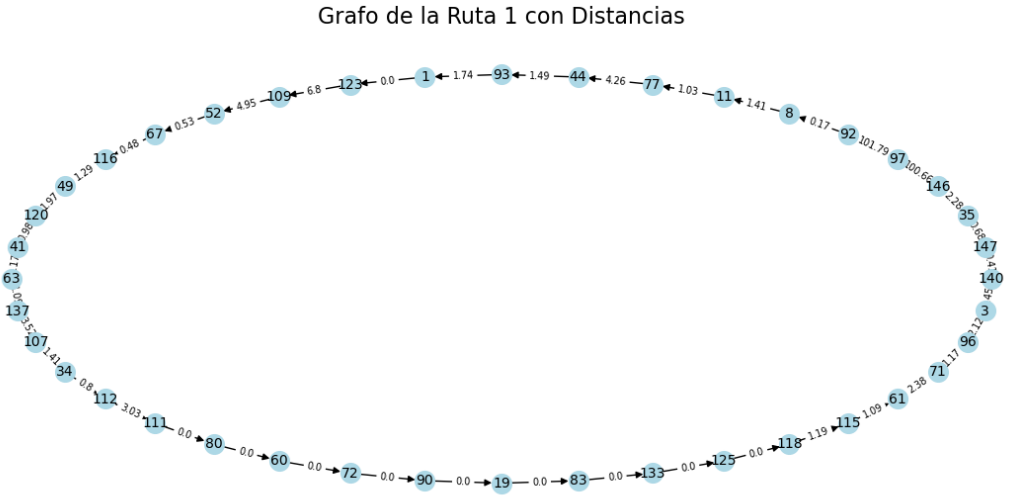
8. Durante el proceso, se realiza un seguimiento del mejor camino encontrado y su longitud mínima (mejor_camino_global y mejor_longitud_global).

9. Al final de las iteraciones, el algoritmo imprime el mejor camino encontrado y su longitud mínima.

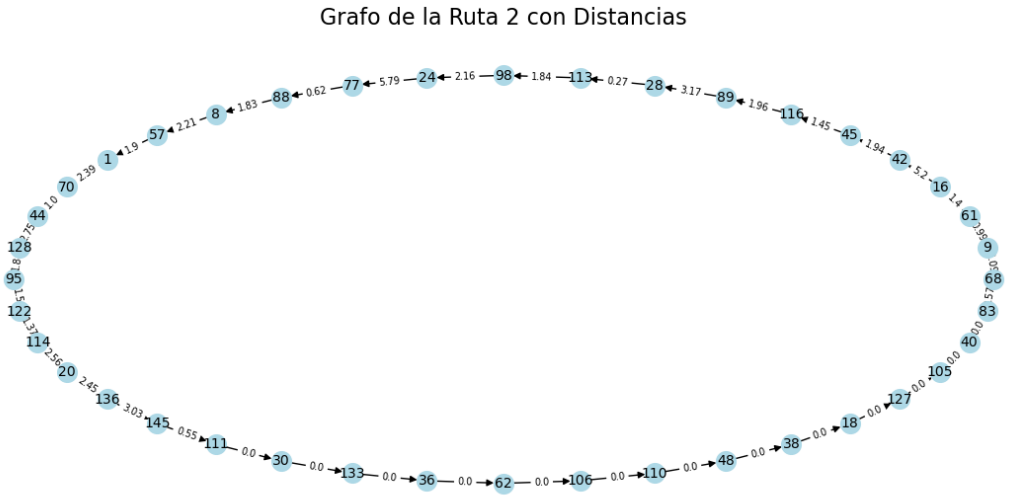
Resultados

Formulación Matemática

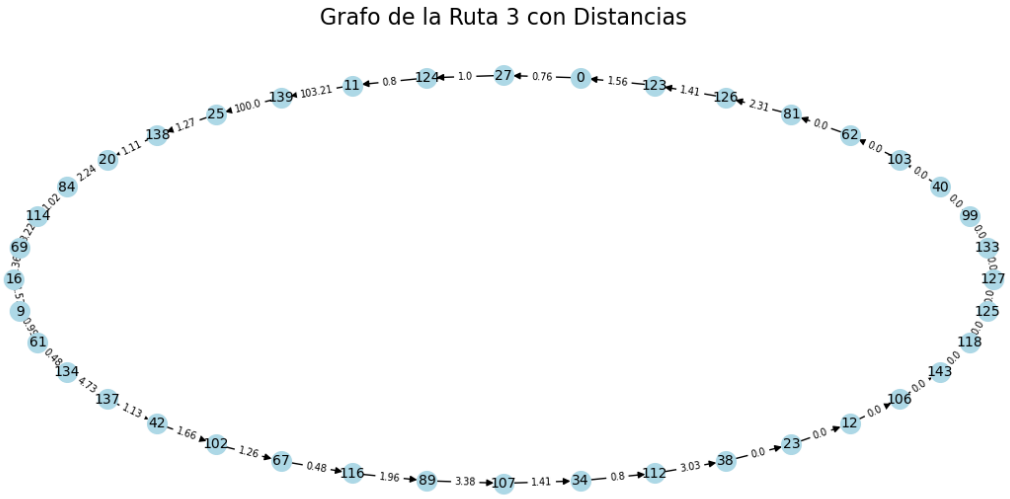
- Resultado matriz 1
    - Ruta: 140 - 147 - 35 - 146 - 97 - 92 - 8 - 11 - 77 - 44 - 93 - 1 - 123 - 109 - 52 - 67 - 116 - 49 - 120 - 41 - 63 - 137 - 107 - 34 - 112 - 111 - 80 - 60 - 72 - 90 - 19 - 83 - 133 - 125 - 118 - 115 - 61 - 71 - 96 - 3 - 140
    - Costo: 253.3715 km
    - Tiempo computacional: 132.237 segundos
    - Grafo:



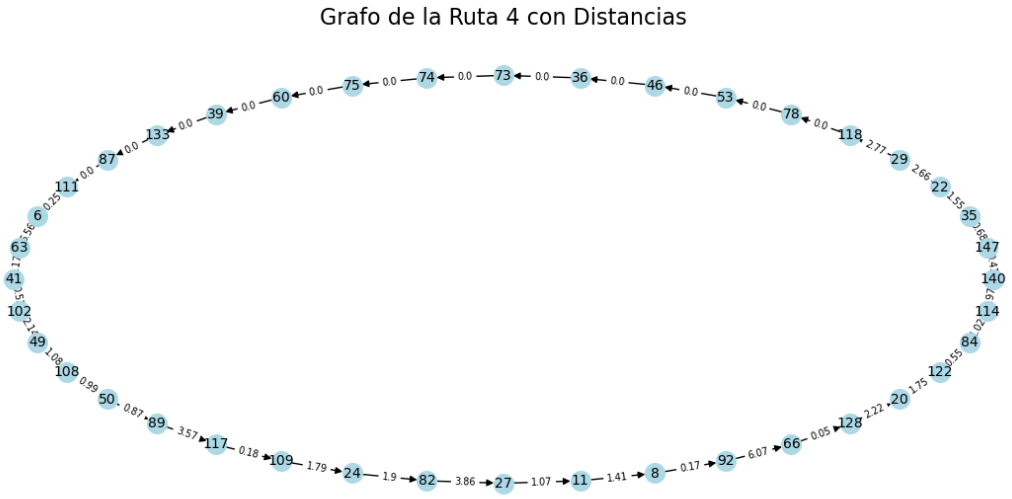
- Resultado matriz 2
    - Ruta: 68 - 9 - 61 - 16 - 42 - 45 - 116 - 89 - 28 - 113 - 98 - 24 - 77 - 88 - 8 - 57 - 1 - 70 - 44 - 128 - 95 - 122 - 114 - 20 - 136 - 145 - 111 - 30 - 133 - 36 - 62 - 106 - 110 - 48 - 38 - 18 - 127 - 105 - 40 - 83 - 68
    - Costo: 53.778 km
    - Tiempo computacional: 45.720 segundos
    - Grafo:



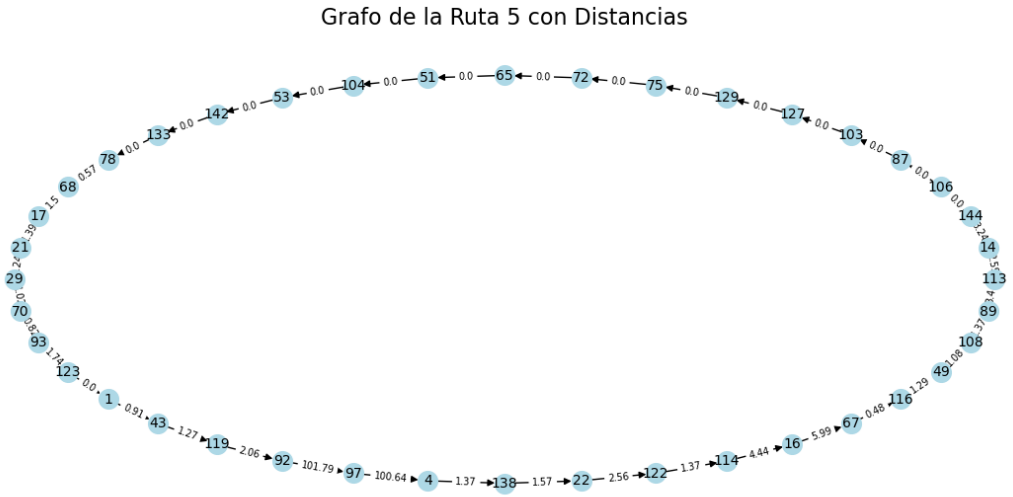
- Resultado matriz 3
  - Ruta: 127 - 133 - 99 - 40 - 103 - 62 - 81 - 126 - 123 - 0 - 27 - 124 - 11 - 139 - 25 - 138 - 20 - 84 - 114 - 69 - 16 - 9 - 61 - 134 - 137 - 42 - 102 - 67 - 116 - 89 - 107 - 34 - 112 - 38 - 23 - 12 - 106 - 143 - 118 - 125 - 127
  - Costo: 244.139 km
  - Tiempo computacional: 11.924 segundos
  - Grafo:



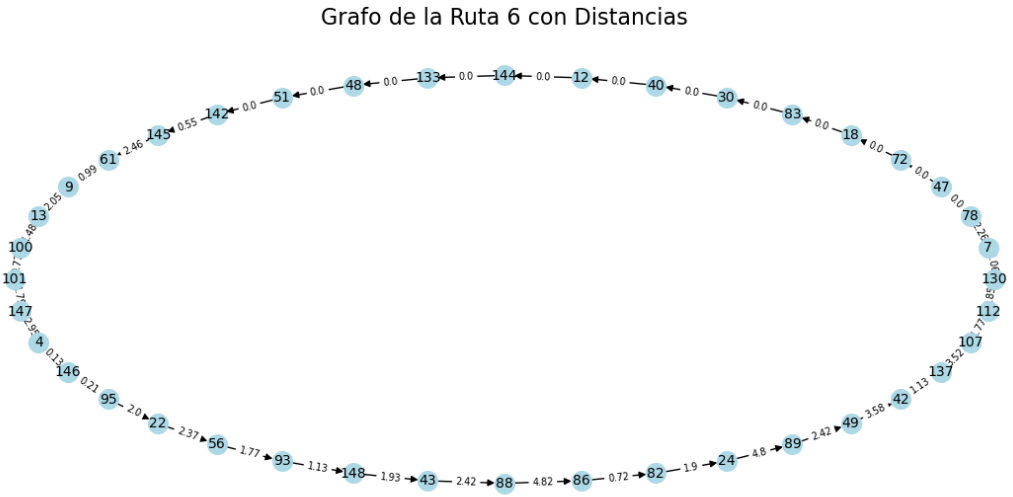
- Resultado matriz 4
    - Ruta: 140 - 147 - 35 - 22 - 29 - 118 - 78 - 53 - 46 - 36 - 73 - 74 - 75 - 60 - 39 - 133 - 87 - 111 - 6 - 63 - 41 - 102 - 49 - 108 - 50 - 89 - 117 - 109 - 24 - 82 - 27 - 11 - 8 - 92 - 66 - 128 - 20 - 122 - 84 - 114 - 140
    - Costo: 49.223 km
    - Tiempo computacional: 28.337 segundos
    - Grafo:



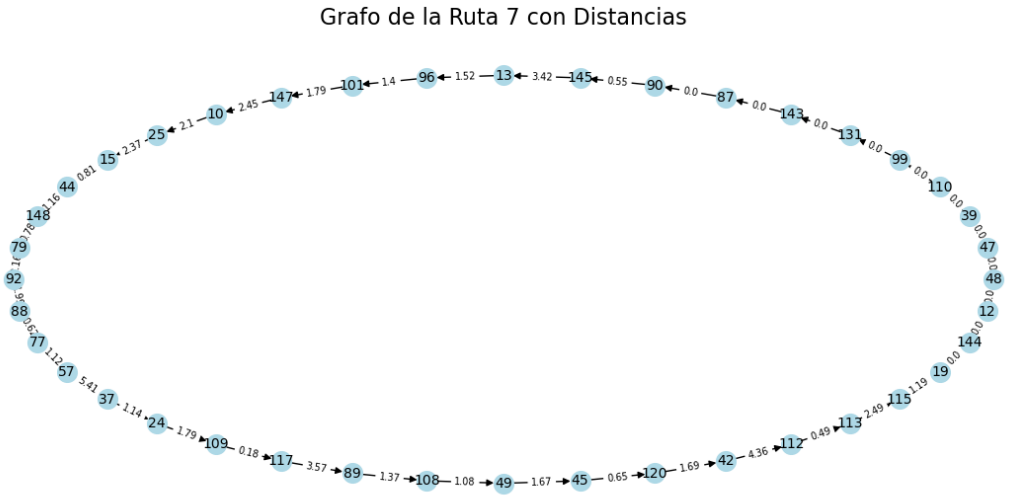
- Resultado matriz 5
    - Ruta: 113 - 14 - 144 - 106 - 87 - 103 - 127 - 129 - 75 - 72 - 65 - 51 - 104 - 53 - 142 - 133 - 78 - 68 - 17 - 21 - 29 - 70 - 93 - 123 - 1 - 43 - 119 - 92 - 97 - 4 - 138 - 22 - 122 - 114 - 16 - 67 - 116 - 49 - 108 - 89 - 113
    - Costo: 245.693 km
    - Tiempo computacional: 3.946 segundos
    - Grafo:



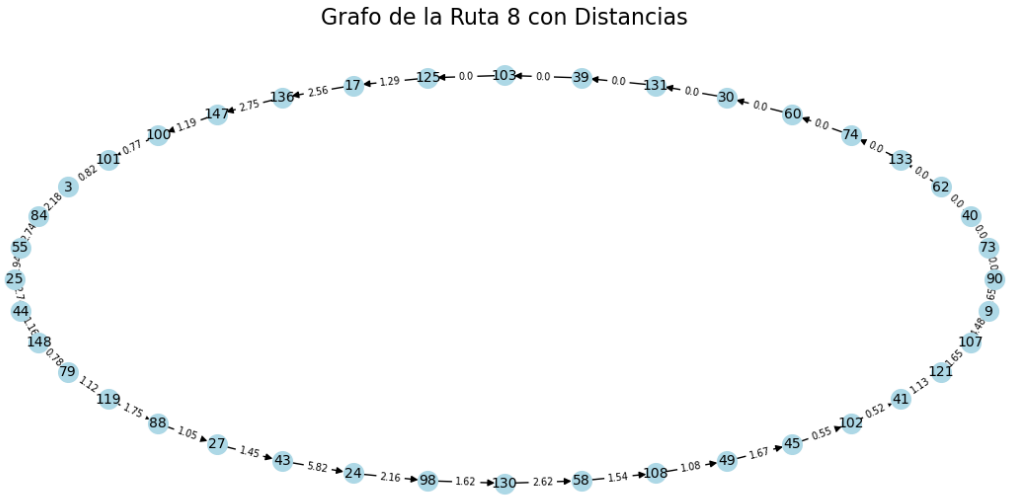
- Resultado matriz 6
    - Ruta: 130 - 7 - 78 - 47 - 72 - 18 - 83 - 30 - 40 - 12 - 144 - 133 - 48 - 51 - 142 - 145 - 61 - 9 - 13 - 100 - 101 - 147 - 4 - 146 - 95 - 22 - 56 - 93 - 148 - 43 - 88 - 86 - 82 - 24 - 89 - 49 - 42 - 137 - 107 - 112 - 130
    - Costo: 53.819 km
    - Tiempo computacional: 42.910 segundos
    - Grafo:



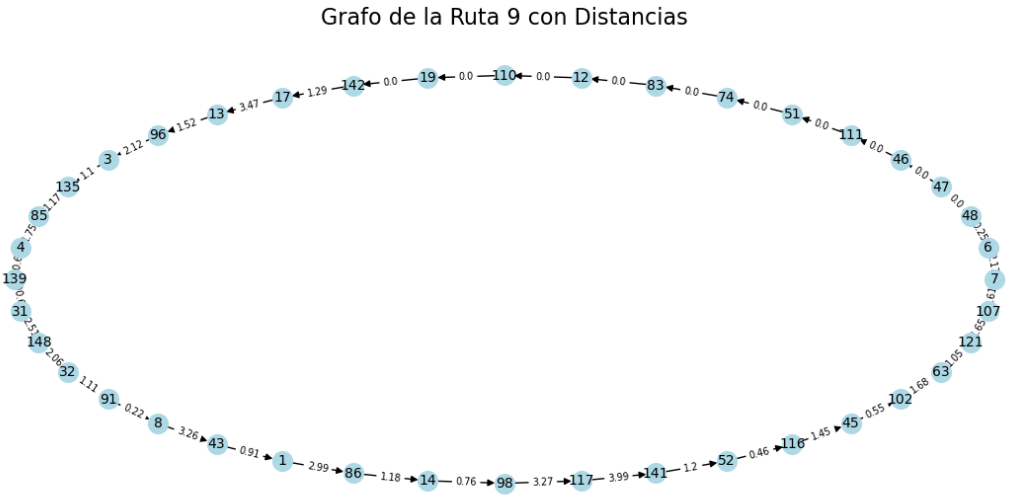
- Resultado matriz 7
    - Ruta: 48 - 47 - 39 - 110 - 99 - 131 - 143 - 87 - 90 - 145 - 13 - 96 - 101 - 147 - 10 - 25 - 15 - 44 - 148 - 79 - 92 - 88 - 77 - 57 - 37 - 24 - 109 - 117 - 89 - 108 - 49 - 45 - 120 - 42 - 112 - 113 - 115 - 19 - 144 - 12 - 48
    - Costo: 51.335 km
    - Tiempo computacional: 76.673 segundos
    - Grafo:



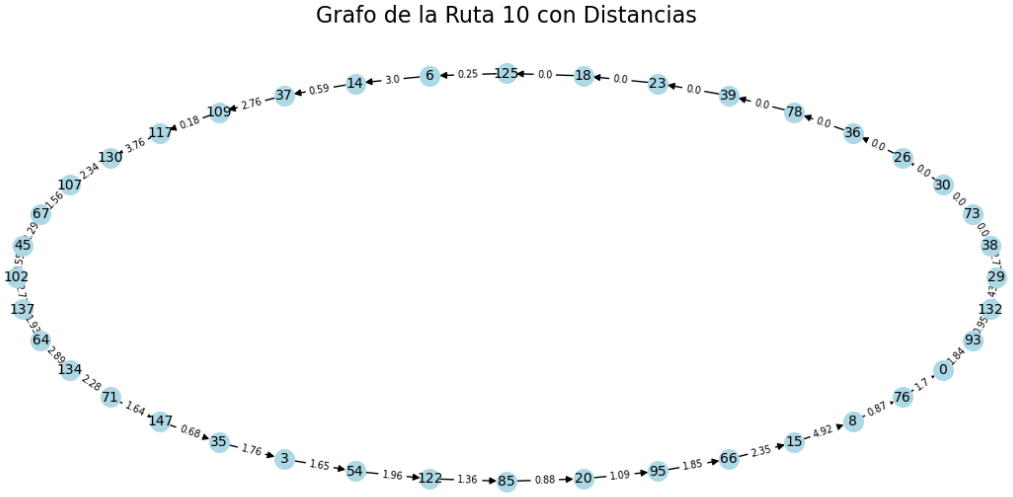
- Resultado matriz 8
    - Ruta: 90 - 73 - 40 - 62 - 133 - 74 - 60 - 30 - 131 - 39 - 103 - 125 - 17 - 136 - 147 - 100 - 101 - 3 - 84 - 55 - 25 - 44 - 148 - 79 - 119 - 88 - 27 - 43 - 24 - 98 - 130 - 58 - 108 - 49 - 45 - 102 - 41 - 121 - 107 - 9 - 90
    - Costo: 52.809 km
    - Tiempo computacional: 77.193 segundos
    - Grafo:



- Resultado matriz 9
    - Ruta: 7 - 6 - 48 - 47 - 46 - 111 - 51 - 74 - 83 - 12 - 110 - 19 - 142 - 17 - 13 - 96 - 3 - 135 - 85 - 4 - 139 - 31 - 148 - 32 - 91 - 8 - 43 - 1 - 86 - 14 - 98 - 117 - 141 - 52 - 116 - 45 - 102 - 63 - 121 - 107 - 7
    - Costo: 249.039 km
    - Tiempo computacional: 16.160 segundos
    - Grafo:



- Resultado matriz 10
    - Ruta: 29 - 38 - 73 - 30 - 26 - 36 - 78 - 39 - 23 - 18 - 125 - 6 - 14 - 37 - 109 - 117 - 130 - 107 - 67 - 45 - 102 - 137 - 64 - 134 - 71 - 147 - 35 - 3 - 54 - 122 - 85 - 20 - 95 - 66 - 15 - 8 - 76 - 0 - 93 - 132 - 29
    - Costo: 54.846 km
    - Tiempo computacional: 256.859 segundos
    - Grafo:



- Resultado matriz 11

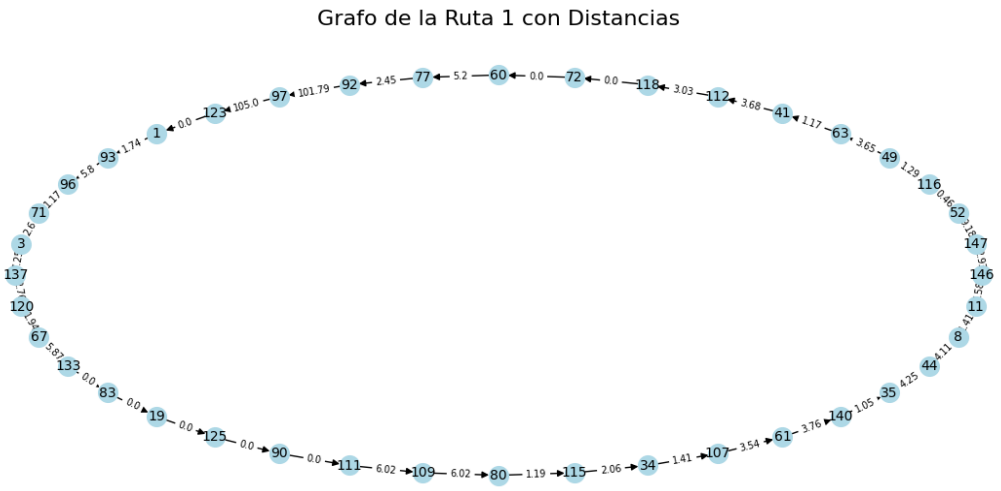
  Lamentablemente, el programa GAMS no puede resolver el problema con 100 nodos; ya que excede los límites de la membresía gratuita y arroja el siguiente mensaje:

```
The model exceeds the demo license limits for linear models of more than
2000 rows or columns. As an academic user, you can submit large models to
the NEOS service for free (see https://www.gams.com/neos) Alternatively,
feel free to contact sales@gams.com to discuss your options
*** Status: Terminated due to a licensing error
```

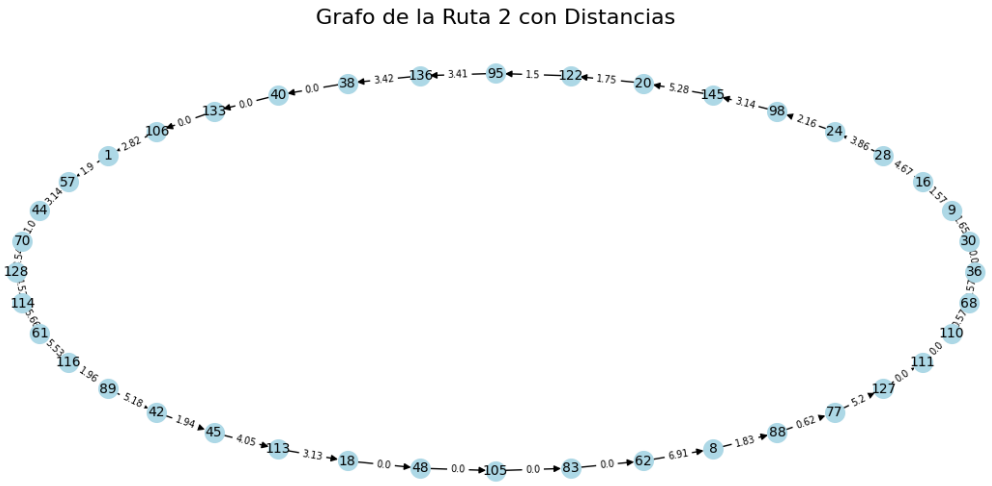
Algoritmo Genético

- Las matrices de 1 a 10, se hicieron con 5,000 iteraciones.
- La matriz de 11, se hizo con 10,000 iteraciones.

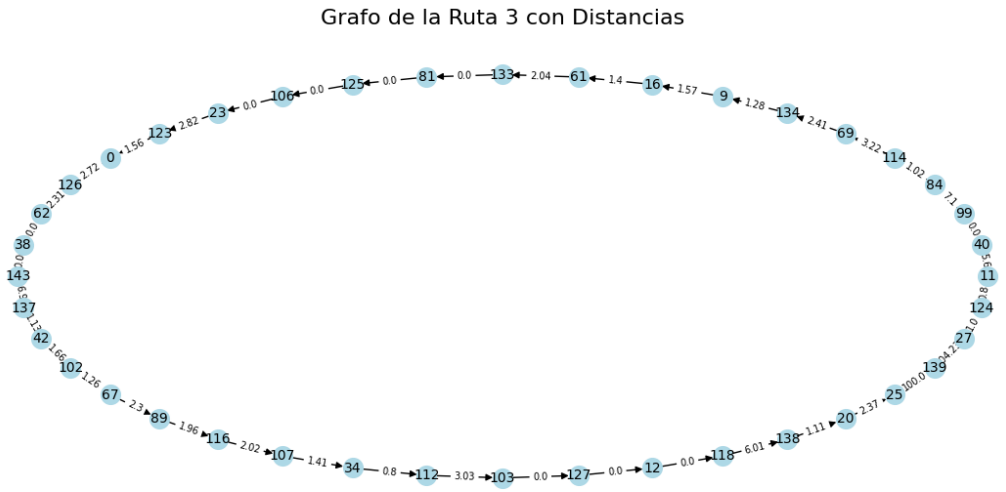
- Resultado matriz 1
    - Ruta: 146 - 147 - 52 - 116 - 49 - 63 - 41 - 112 - 118 - 72 - 60 - 77 - 92 - 97 - 123 - 1 - 93 - 96 - 71 - 3 - 137 - 120 - 67 - 133 - 83 - 19 - 125 - 90 - 111 - 109 - 80 - 115 - 34 - 107 - 61 - 140 - 35 - 44 - 8 - 11 - 146
    - Costo: 311.385506 km
    - Tiempo computacional: 20.33 segundos
    - Grafo:



- Resultado matriz 2
    - Ruta: 36 - 30 - 9 - 16 - 28 - 24 - 98 - 145 - 20 - 122 - 95 - 136 - 38 - 40 - 133 - 106 - 1 - 57 - 44 - 70 - 128 - 114 - 61 - 116 - 89 - 42 - 45 - 113 - 18 - 48 - 105 - 83 - 62 - 8 - 88 - 77 - 127 - 111 - 110 - 68 - 36
    - Costo: 92.45738 km
    - Tiempo computacional: 20.577 segundos
    - Grafo:

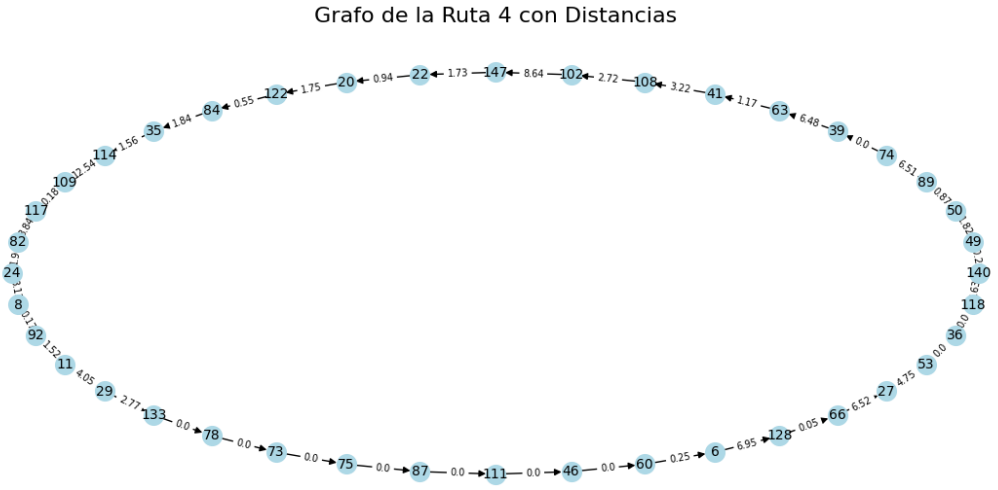


- Resultado matriz 3
    - Ruta: 11 - 40 - 99 - 84 - 114 - 69 - 134 - 9 - 16 - 61 - 133 - 81 - 125 - 106 - 23 - 123 - 0 - 126 - 62 - 38 - 143 - 137 - 42 - 102 - 67 - 89 - 116 - 107 - 34 - 112 - 103 - 127 - 12 - 118 - 138 - 20 - 25 - 139 - 27 - 124 - 11
    - Costo: 273.022288852557
    - Tiempo computacional: 20.69 segundos
    - Grafo:

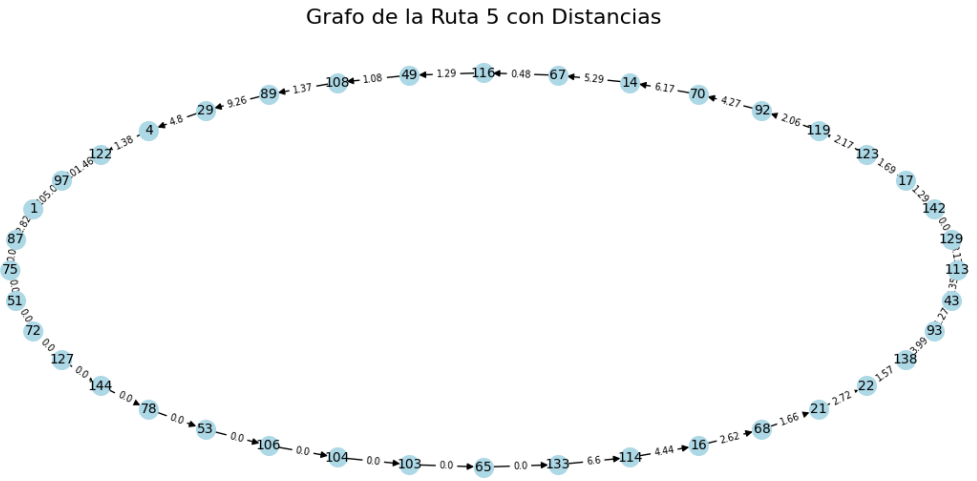


- Resultado matriz 4

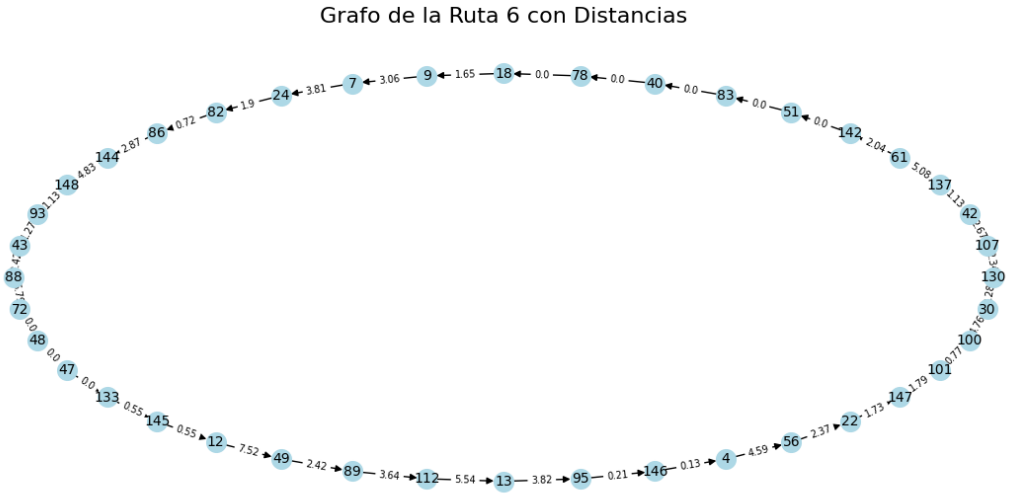
- Ruta: 140 - 49 - 50 - 89 - 74 - 39 - 63 - 41 - 108 - 102 - 147 - 22 - 20 - 122 - 84 - 35 - 114 - 109 - 117 - 82 - 24 - 8 - 92 - 11 - 29 - 133 - 78 - 73 - 75 - 87 - 111 - 46 - 60 - 6 - 128 - 66 - 27 - 53 - 36 - 118 - 140
- Costo: 108.2816 km
- Tiempo computacional: 20.7125 segundos
- Grafo:



- Resultado matriz 5
  - Ruta: 113 - 129 - 142 - 17 - 123 - 119 - 92 - 70 - 14 - 67 - 116 - 49 - 108 - 89 - 29 - 4 - 122 - 97 - 1 - 87 - 75 - 51 - 72 - 127 - 144 - 78 - 53 - 106 - 104 - 103 - 65 - 133 - 114 - 16 - 68 - 21 - 22 - 138 - 93 - 43 - 113
  - Costo: 286.2192 km
  - Tiempo computacional: 21.1852 segundos
  - Grafo:

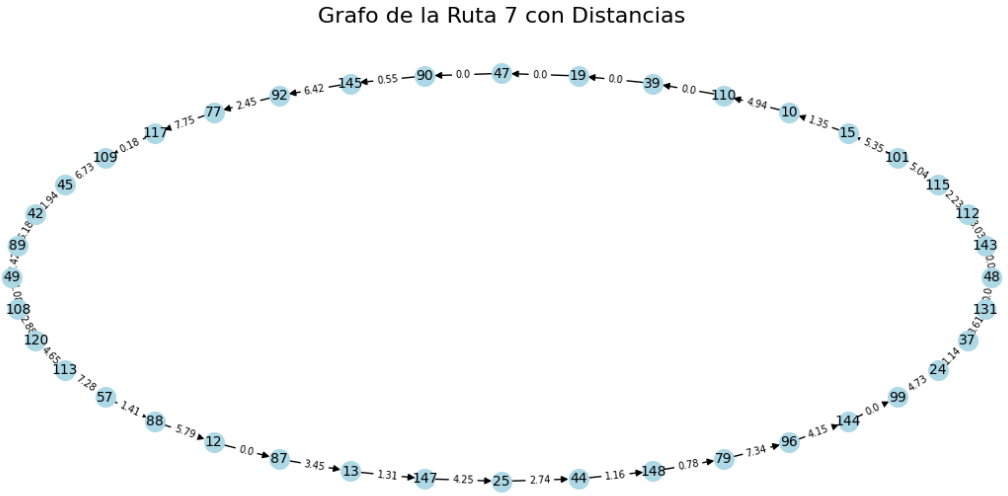


- Resultado matriz 6:
    - Ruta: 130 - 107 - 42 - 137 - 61 - 142 - 51 - 83 - 40 - 78 - 18 - 9 - 7 - 24 - 82 - 86 - 144 - 148 - 93 - 43 - 88 - 72 - 48 - 47 - 133 - 145 - 12 - 49 - 89 - 112 - 13 - 95 - 146 - 4 - 56 - 22 - 147 - 101 - 100 - 30 - 130
    - Costo: 86.3856 km
    - Tiempo computacional: 21.5 segundos
    - Grafo:

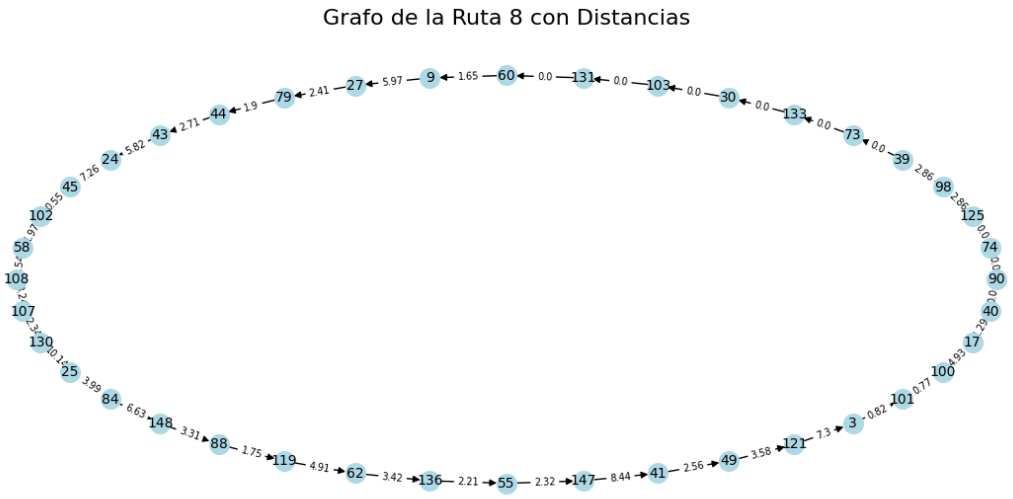


- Resultado matriz 7:

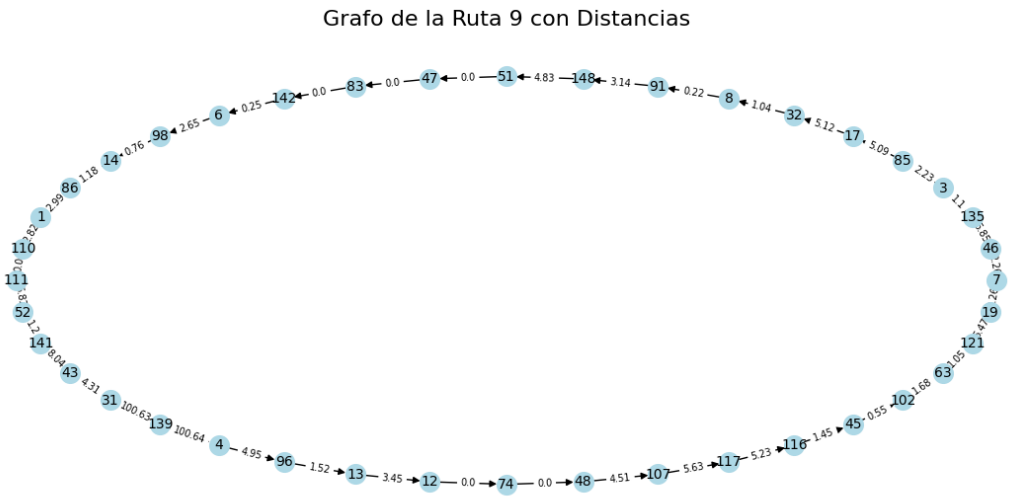
- Ruta: : 48 - 143 - 112 - 115 - 101 - 15 - 10 - 110 - 39 - 19 - 47 - 90 - 145 - 92 - 77 - 117 - 109 - 45 - 42 - 89 - 49 - 108 - 120 - 113 - 57 - 88 - 12 - 87 - 13 - 147 - 25 - 44 - 148 - 79 - 96 - 144 - 99 - 24 - 37 - 131 - 48
- Costo: 113.32562 km
- Tiempo computacional: 20.76138 segundos
- Grafo:



- Resultado matriz 8:
  - Ruta: 90 - 74 - 125 - 98 - 39 - 73 - 133 - 30 - 103 - 131 - 60 - 9 - 27 - 79 - 44 - 43 - 24 - 45 - 102 - 58 - 108 - 107 - 130 - 25 - 84 - 148 - 88 - 119 - 62 - 136 - 55 - 147 - 41 - 49 - 121 - 3 - 101 - 100 - 17 - 40 - 90
  - Costo: 111.4758 km
  - Tiempo computacional: 22.23093 segundos
  - Grafo:

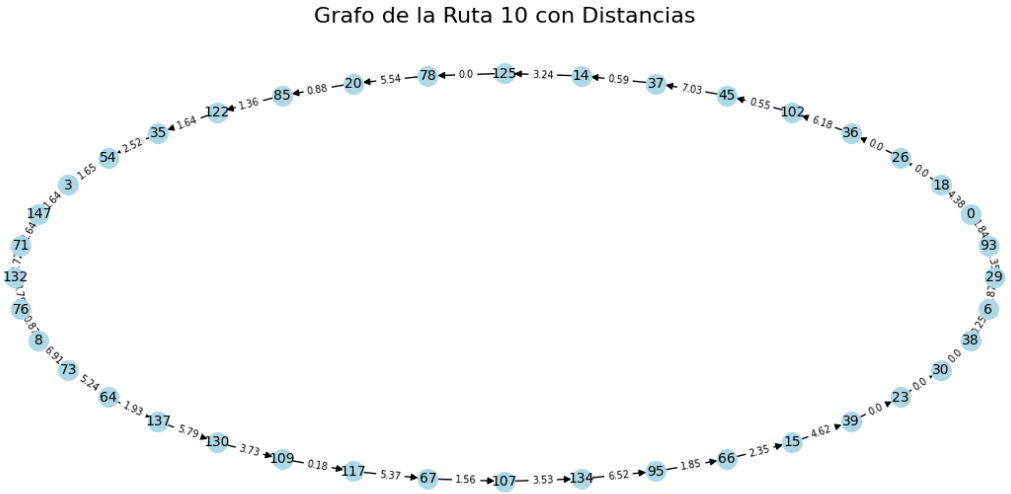


- Resultado matriz 9:
    - Ruta: 7 - 46 - 135 - 3 - 85 - 17 - 32 - 8 - 91 - 148 - 51 - 47 - 83 - 142 - 6 - 98 - 14 - 86 - 1 - 110 - 111 - 52 - 141 - 43 - 31 - 139 - 4 - 96 - 13 - 12 - 74 - 48 - 107 - 117 - 116 - 45 - 102 - 63 - 121 - 19 - 7
    - Costo: 299.9334430683181 km
    - Tiempo computacional: 21.497066 segundos
    - Grafo:

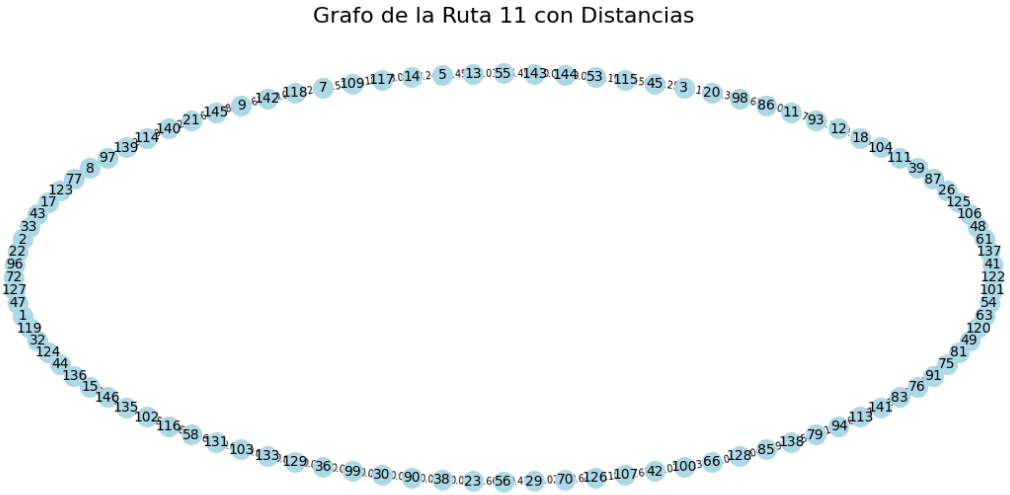


- Resultado matriz 10:

- Ruta: 29 - 93 - 0 - 18 - 26 - 36 - 102 - 45 - 37 - 14 - 125 - 78 - 20 - 85 - 122 - 35 - 54 - 3 - 147 - 71 - 132 - 76 - 8 - 73 - 64 - 137 - 130 - 109 - 117 - 67 - 107 - 134 - 95 - 66 - 15 - 39 - 23 - 30 - 38 - 6 - 29
- Costo: 103.1156822 km
- Tiempo computacional: 22.40292 segundos
- Grafo:

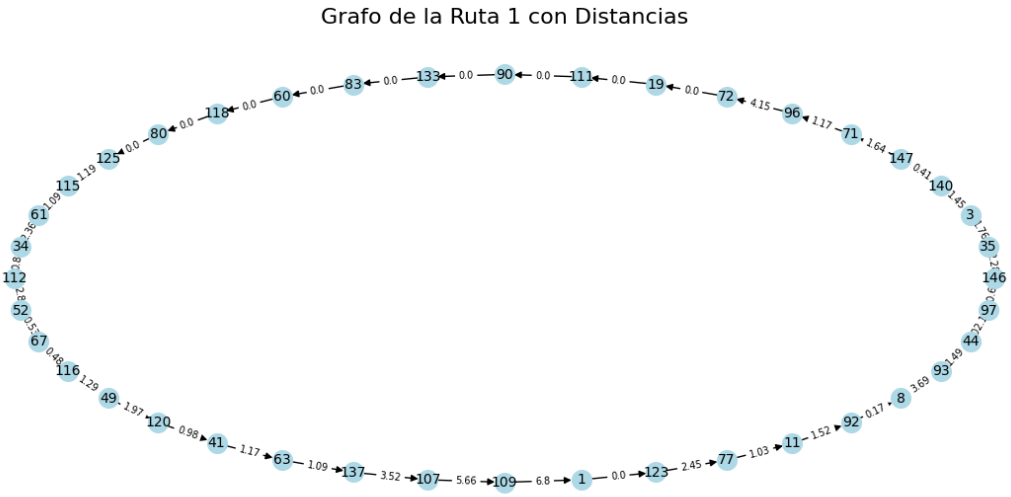


- Resultado matriz 11:
  - Ruta: 122 - 41 - 137 - 61 - 48 - 106 - 125 - 26 - 87 - 39 - 111 - 104 - 18 - 12 - 93 - 11 - 86 - 98 - 20 - 3 - 45 - 115 - 53 - 144 - 143 - 55 - 13 - 5 - 14 - 117 - 109 - 7 - 118 - 142 - 9 - 145 - 21 - 140 - 114 - 139 - 97 - 8 - 77 - 123 - 17 - 43 - 33 - 2 - 22 - 96 - 72 - 127 - 47 - 1 - 119 - 32 - 124 - 44 - 136 - 15 - 146 - 135 - 102 - 116 - 58 - 131 - 103 - 133 - 129 - 36 - 99 - 30 - 90 - 38 - 23 - 56 - 29 - 70 - 126 - 107 - 42 - 100 - 66 - 128 - 85 - 138 - 79 - 94 - 113 - 141 - 83 - 76 - 91 - 75 - 81 - 49 - 120 - 63 - 54 - 101 - 122
  - Costo: 660.04304 km
  - Tiempo computacional: 138.27192 segundos
  - Grafo:

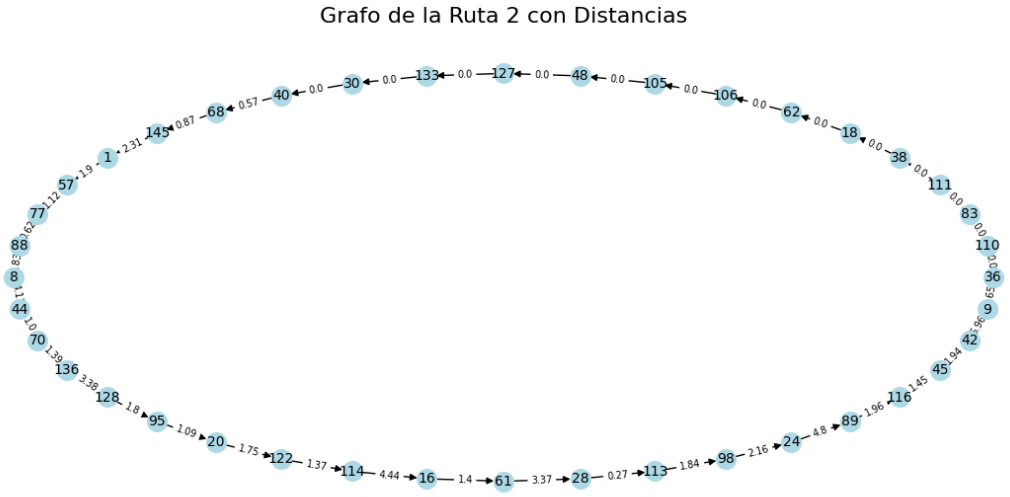


Algoritmo de Hormigas

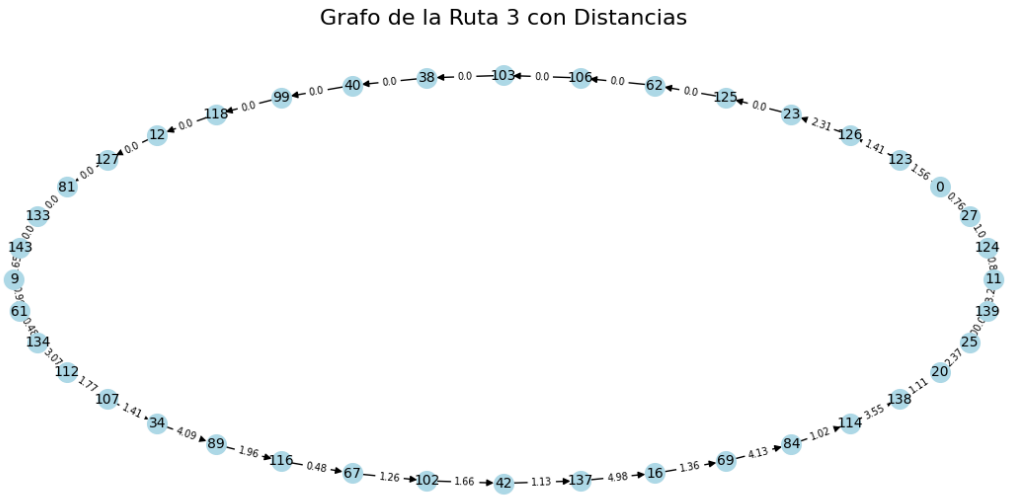
- Resultado matriz 1
  - Ruta: 146 - 35 - 3 - 140 - 147 - 71 - 96 - 72 - 19 - 111 - 90 - 133 - 83 - 60 - 118 - 80 - 125 - 115 - 61 - 34 - 112 - 52 - 67 - 116 - 49 - 120 - 41 - 63 - 137 - 107 - 109 - 1 - 123 - 77 - 11 - 92 - 8 - 93 - 44 - 97 - 146
  - Costo: 257.725 km
  - Tiempo computacional: 32 segundos
  - Grafo:



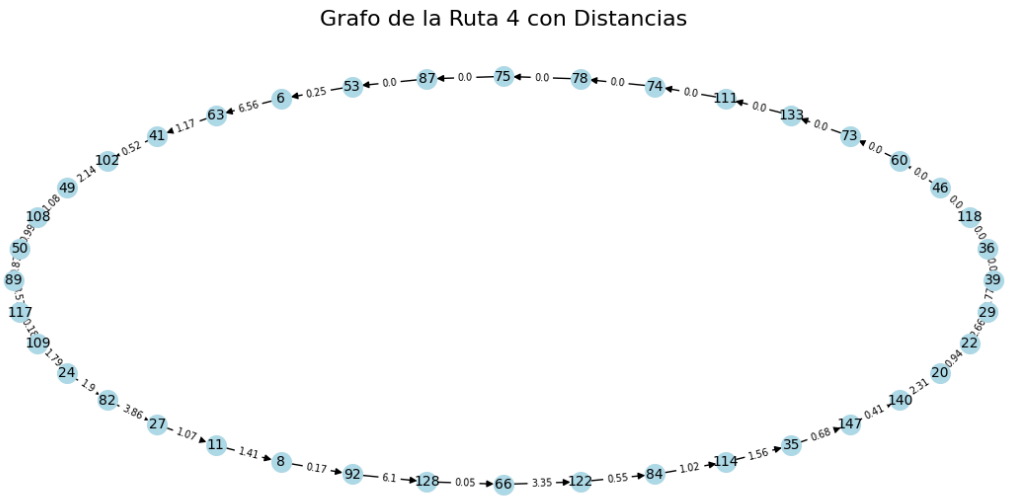
- Resultado matriz 2
    - Ruta: 36 - 110 - 83 - 111 - 38 - 18 - 62 - 106 - 105 - 48 - 127 - 133 - 30 - 40 - 68 - 145 - 1 - 57 - 77 - 88 - 8 - 44 - 70 - 136 - 128 - 95 - 20 - 122 - 114 - 16 - 61 - 28 - 113 - 98 - 24 - 89 - 116 - 45 - 42 - 9 - 36
    - Costo: 56.348 km
    - Tiempo computacional: 32 segundos
    - Grafo:



- Resultado matriz 3
    - Ruta: 11 - 124 - 27 - 0 - 123 - 126 - 23 - 125 - 62 - 106 - 103 - 38 - 40 - 99 - 118 - 12 - 127 - 81 - 133 - 143 - 9 - 61 - 134 - 112 - 107 - 34 - 89 - 116 - 67 - 102 - 42 - 137 - 16 - 69 - 84 - 114 - 138 - 20 - 25 - 139 - 11
    - Costo: 249.511 km
    - Tiempo computacional: 31 segundos
    - Grafo:

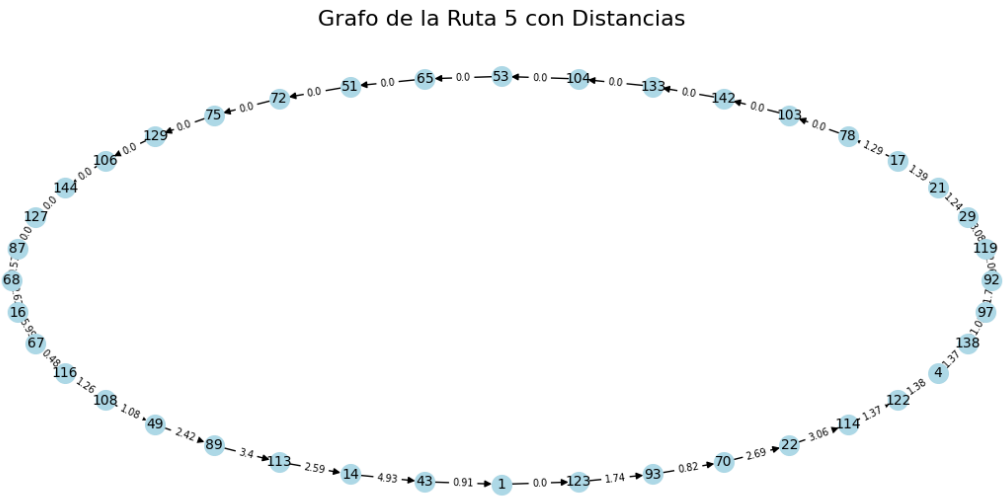


- Resultado matriz 4
    - Ruta: 39 - 36 - 118 - 46 - 60 - 73 - 133 - 111 - 74 - 78 - 75 - 87 - 53 - 6 - 63 - 41 - 102 - 49 - 108 - 50 - 89 - 117 - 109 - 24 - 82 - 27 - 11 - 8 - 92 - 128 - 66 - 122 - 84 - 114 - 35 - 147 - 140 - 20 - 22 - 29 - 39
    - Costo: 49.927 km
    - Tiempo computacional: 30 segundos
    - Grafo:

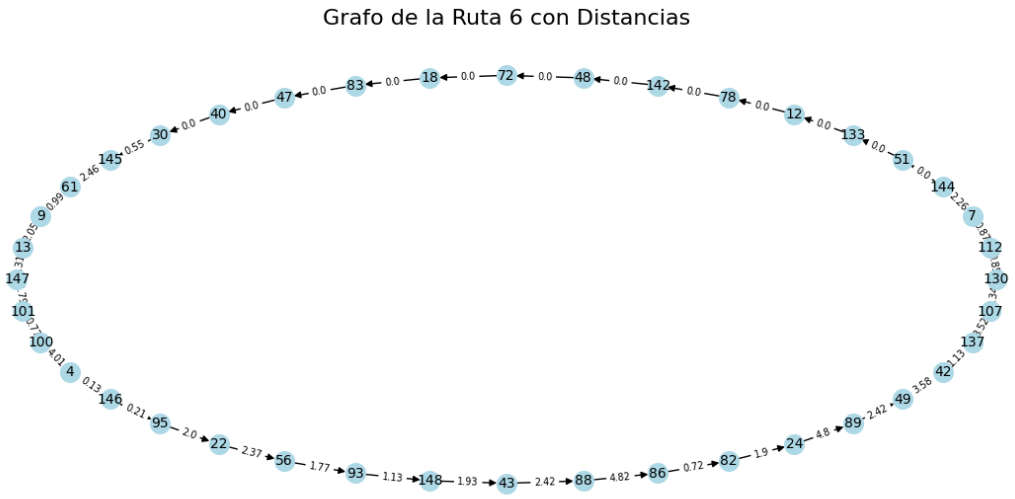


- Resultado matriz 5

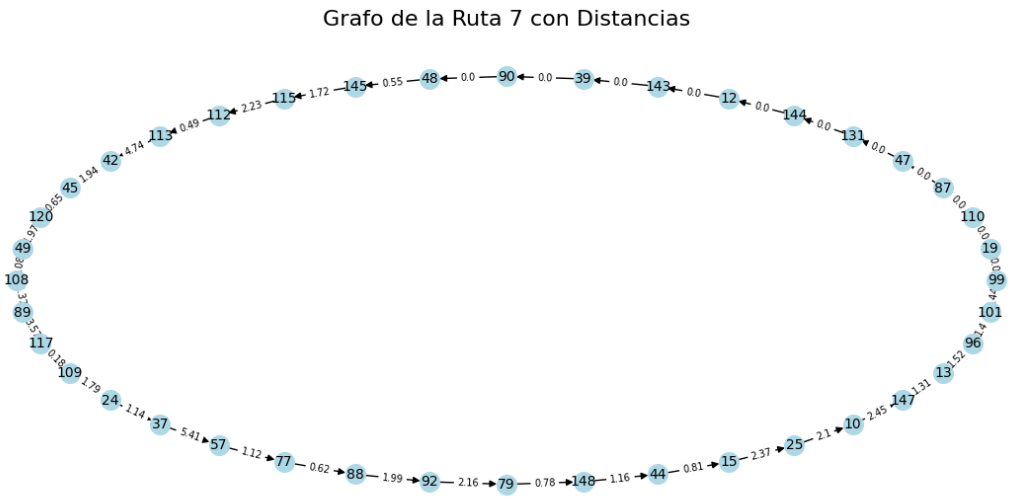
- Ruta: 92 - 119 - 29 - 21 - 17 - 78 - 103 - 142 - 133 - 104 - 53 - 65 - 51 - 72 - 75 - 129 - 106 - 144 - 127 - 87 - 68 - 16 - 67 - 116 - 108 - 49 - 89 - 113 - 14 - 43 - 1 - 123 - 93 - 70 - 22 - 114 - 122 - 4 - 138 - 97 - 92
- Costo: 250.593 km
- Tiempo computacional: 31 segundos
- Grafo:



- Resultado matriz 6
  - Ruta: 130 - 112 - 7 - 144 - 51 - 133 - 12 - 78 - 142 - 48 - 72 - 18 - 83 - 47 - 40 - 30 - 145 - 61 - 9 - 13 - 147 - 101 - 100 - 4 - 146 - 95 - 22 - 56 - 93 - 148 - 43 - 88 - 86 - 82 - 24 - 89 - 49 - 42 - 137 - 107 - 130
  - Costo: 55.087 km
  - Tiempo computacional: 32 segundos
  - Grafo:

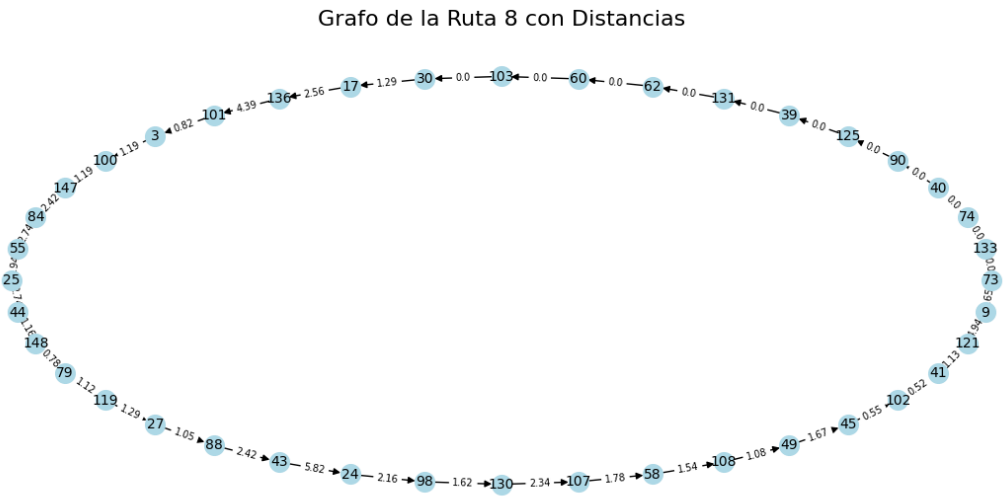


- Resultado matriz 7
    - Ruta: 99 - 19 - 110 - 87 - 47 - 131 - 144 - 12 - 143 - 39 - 90 - 48 - 145 - 115 - 112 - 113 - 42 - 45 - 120 - 49 - 108 - 89 - 117 - 109 - 24 - 37 - 57 - 77 - 88 - 92 - 79 - 148 - 44 - 15 - 25 - 10 - 147 - 13 - 96 - 101 - 99
    - Costo: 54.070 km
    - Tiempo computacional: 33 segundos
    - Grafo:

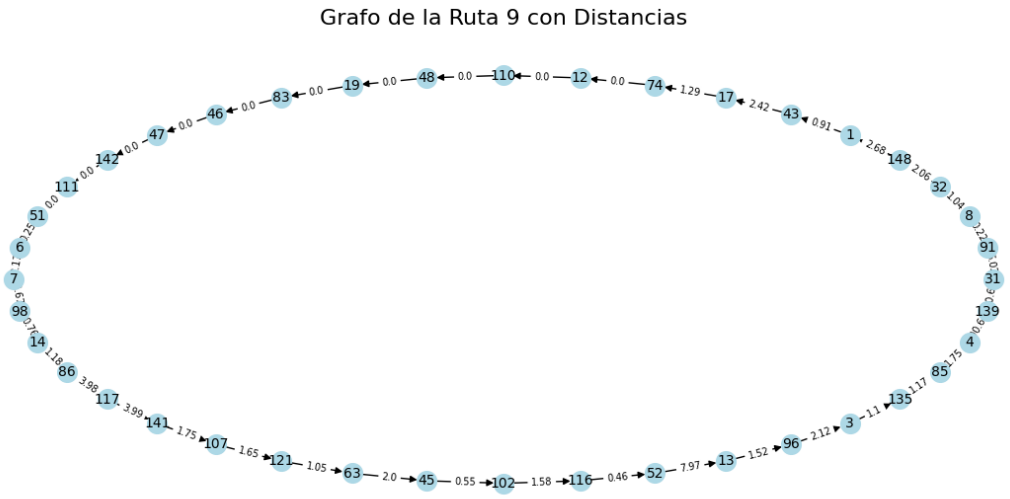


- Resultado matriz 8

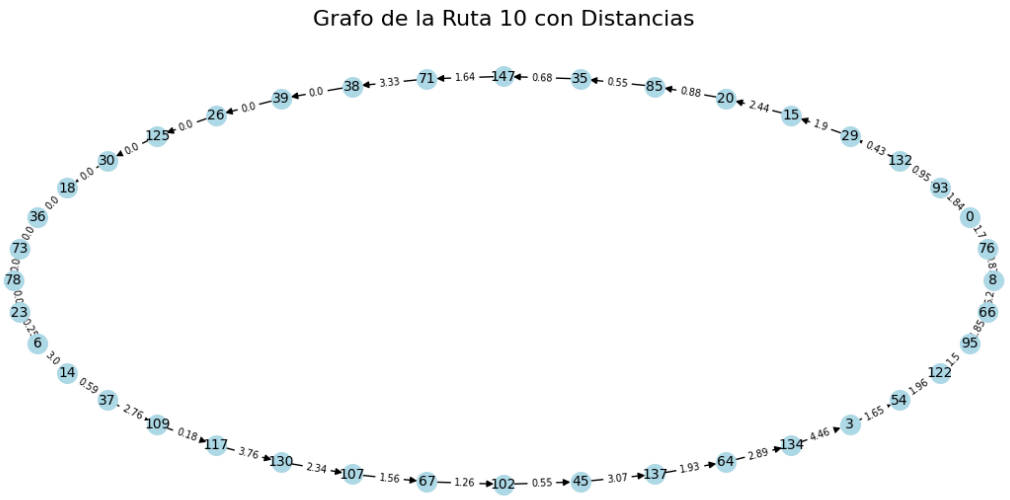
- Ruta: 73 - 133 - 74 - 40 - 90 - 125 - 39 - 131 - 62 - 60 - 103 - 30 - 17 - 136 - 101 - 3 - 100 - 147 - 84 - 55 - 25 - 44 - 148 - 79 - 119 - 27 - 88 - 43 - 24 - 98 - 130 - 107 - 58 - 108 - 49 - 45 - 102 - 41 - 121 - 9 - 73
- Costo: 55.930 km
- Tiempo computacional: 33 segundos
- Grafo:



- Resultado matriz 9
  - Ruta: 31 - 91 - 8 - 32 - 148 - 1 - 43 - 17 - 74 - 12 - 110 - 48 - 19 - 83 - 46 - 47 - 142 - 111 - 51 - 6 - 7 - 98 - 14 - 86 - 117 - 141 - 107 - 121 - 63 - 45 - 102 - 116 - 52 - 13 - 96 - 3 - 135 - 85 - 4 - 139 - 31
  - Costo: 255.583 km
  - Tiempo computacional: 32 segundos
  - Grafo:

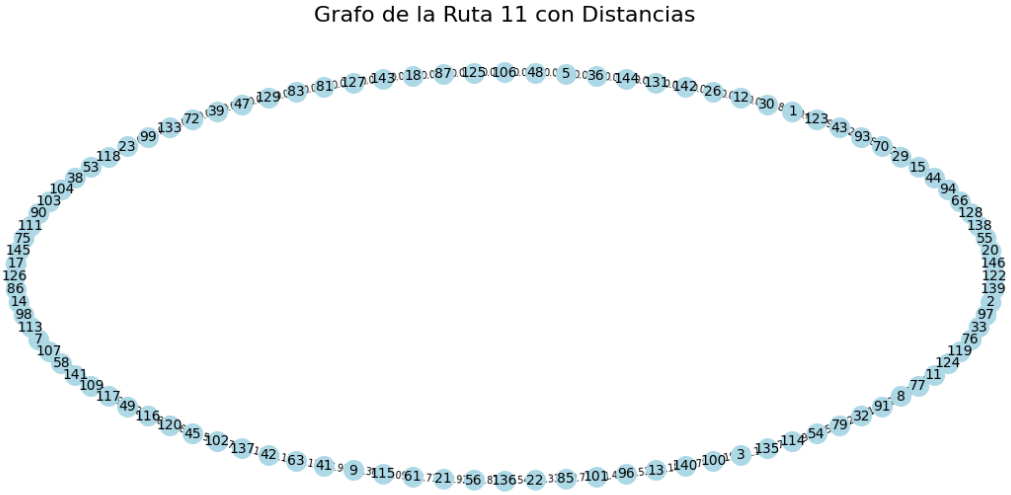


- Resultado matriz 10
    - Ruta: 8 - 76 - 0 - 93 - 132 - 29 - 15 - 20 - 85 - 35 - 147 - 71 - 38 - 39 - 26 - 125 - 30 - 18 - 36 - 73 - 78 - 23 - 6 - 14 - 37 - 109 - 117 - 130 - 107 - 67 - 102 - 45 - 137 - 64 - 134 - 3 - 54 - 122 - 95 - 66 - 8
    - Costo: 58.960 km
    - Tiempo computacional: 32 segundos
    - Grafo:



    - Grafo:
    - Resultado matriz 11

- Ruta: 122 - 146 - 20 - 55 - 138 - 128 - 66 - 94 - 44 - 15 - 29 - 70 - 93 - 43 - 123 - 1 - 30 - 12 - 26 - 142 - 131 - 144 - 36 - 5 - 48 - 106 - 125 - 87 - 18 - 143 - 127 - 81 - 83 - 129 - 47 - 39 - 72 - 133 - 99 - 23 - 118 - 53 - 38 - 104 - 103 - 90 - 111 - 75 - 145 - 17 - 126 - 86 - 14 - 98 - 113 - 7 - 107 - 58 - 141 - 109 - 117 - 49 - 116 - 120 - 45 - 102 - 137 - 42 - 63 - 41 - 9 - 115 - 61 - 21 - 56 - 136 - 22 - 85 - 101 - 96 - 13 - 140 - 100 - 3 - 135 - 114 - 54 - 79 - 32 - 91 - 8 - 77 - 11 - 124 - 119 - 76 - 33 - 97 - 2 - 139 - 122
- Costo: 302.396 km
- Tiempo computacional: 7 minutos



Discusión

El propósito central de este estudio fue abordar el desafiante problema del agente viajero (TSP) empleando una variedad de enfoques de optimización avanzados, que incluyen un modelo matemático preciso, un algoritmo genético y un algoritmo de colonia de hormigas. Los resultados obtenidos brindan una visión interesante de la eficacia y eficiencia de estos métodos en la resolución de TSP en diversos contextos.

Inicialmente, el modelo matemático exacto demostró su capacidad para encontrar soluciones óptimas en instancias más pequeñas del problema, especialmente en el caso de 40 nodos. No obstante, se encontró con limitaciones al enfrentar instancias más grandes, como las de 100 nodos, donde la complejidad computacional se volvió un desafío evidente. Este resultado pone de manifiesto la naturaleza NP-duro del TSP y la dificultad de hallar soluciones precisas para problemas de gran envergadura en un tiempo razonable.

Por otro lado, los algoritmos heurísticos, como el algoritmo genético y el algoritmo de colonia de hormigas, demostraron ser más adaptables al TSP en su variante de 100 nodos. Aunque sus soluciones no alcanzaron la óptima, su capacidad para encontrar soluciones subóptimas resulta prometedora. Entre estos dos, el algoritmo de colonia de hormigas se destacó al generar soluciones más próximas a las óptimas, lo que sugiere su utilidad en aplicaciones del mundo real donde la precisión es crucial.

En términos de tiempo computacional, se observó que los algoritmos heurísticos superaron al modelo matemático exacto en eficiencia cuando se trataba de problemas más grandes. Tanto el algoritmo genético como el algoritmo de colonia de hormigas lograron resolver el TSP para 100 nodos en un tiempo relativamente breve en comparación con el modelo matemático exacto. A pesar de las diferencias en los tiempos, ambos mantuvieron un rendimiento aceptable, lo que sugiere su utilidad en aplicaciones prácticas donde la velocidad desempeña un papel crucial.

Este estudio resalta la importancia de seleccionar la estrategia de optimización apropiada según el tamaño del problema y las restricciones de tiempo. Si se busca una solución precisa pero se enfrenta un problema de gran envergadura, los algoritmos heurísticos, como el algoritmo de colonia de hormigas, pueden ofrecer resultados satisfactorios en un plazo razonable. Por otro lado, el modelo matemático exacto sigue siendo valioso para problemas más pequeños que requieren precisión. En última instancia, esta investigación brinda una perspectiva valiosa sobre cómo abordar el desafiante problema del agente viajero en diversos contextos y destaca la importancia de la eficiencia en la toma de decisiones de optimización en el mundo real.

Conclusiones

En resumen, este estudio se centró en la resolución del desafiante problema del agente viajero (TSP) mediante diversas estrategias de optimización. Se evaluaron tres enfoques principales: un modelo matemático exacto, un algoritmo genético y un algoritmo de colonia de hormigas, con respecto a su capacidad para hallar soluciones óptimas, su adaptabilidad a problemas más grandes y su eficiencia en términos de tiempo de cómputo.

El modelo matemático exacto demostró ser eficaz al encontrar soluciones óptimas en instancias más pequeñas del TSP. Sin embargo, su complejidad computacional limitó su aplicabilidad en problemas más grandes.

Por otro lado, los algoritmos heurísticos, como el algoritmo genético y el algoritmo de colonia de hormigas, destacaron por su capacidad para manejar problemas de mayor envergadura con eficiencia. Aunque no siempre alcanzaron soluciones óptimas, proporcionaron soluciones subóptimas adecuadas para aplicaciones del mundo real. En cuanto al tiempo de cómputo, los algoritmos heurísticos demostraron ser más eficientes que el modelo matemático exacto para problemas de mayor tamaño.

En última instancia, este estudio enfatiza la importancia de elegir la estrategia de optimización adecuada según el tamaño y la complejidad del problema. Mientras que el modelo matemático exacto sigue siendo valioso para problemas pequeños que requieren precisión, los algoritmos heurísticos representan una elección eficiente y efectiva para abordar problemas más grandes en aplicaciones del mundo real. Estos resultados proporcionan una valiosa orientación para tomar decisiones informadas en la resolución de problemas de optimización en diversos contextos.

Bibliografía

Daniells, L., (s.f.). *The travelling salesman problem.* https://www.lancaster.ac.uk/stor-i-student-sites/libby-daniells/2020/04/21/the-travelling-salesman-problem/.

Del Estado De Hidalgo, U. A. (s. f.). *Problema del agente viajero*. https://www.uaeh.edu.mx/scige/boletin/tlahuelilpan/n3/e5.html

Enzyme. (2022). Algoritmos genéticos y su gran cantidad de aplicaciones. *Algoritmos genéticos y su gran cantidad de aplicaciones*. https://enzyme.biz/blog/algoritmos-geneticos-y-sus-aplicaciones-para-soluciones

Kirkpatrick, I. S. (2020). *A nearest neighbor solution in go to the traveling salesman problem.* https://levelup.gitconnected.com/a-nearest-neighbor-solution-in-go-to-the-traveling-salesman-problem-d4d56125b571.

Miguel, J. (2011). *¿Cómo calcular la distancia entre dos puntos geográficos en C? (Fórmula de Haversine)*. En: Genbeta. https://www.genbeta.com/desarrollo/como-calcular-la-distancia-entre-dos-puntosgeograficos-en-c-formula-de-haversine.

Tobin, B. (2023). *The travelling salesman problem (TSP)*. https://smartroutes.io/blogs/the-travelling-salesman-problem/.

Traveling Salesperson problem using branch and bound. (s.f.). https://www.javatpoint.com/traveling-salesperson-problem-using-branch-and-bound.