Instituto Tecnológico Autónomo de México

Investigación de Operaciones

Proyecto: Problema del Viajero

Santiago Fernández Del Castillo	189210
Alexander Yanci	192185
Antonio Fernández	-

Sábado 14 de diciembre de 2024

Profesora: Mayra Nuñez Lopez

1. Abstract

El texto aborda el problema del viajero, que debe de recorrer, sin repetición, todos los nodos de un grafo de la forma más corta, terminando en el mismo nodo en el que comienza. ¿Sencillo? Pues es un problema que ha puesto a pensar a grandes matemáticos desde hace más de un siglo y que recientemente, según Jane Rigny cientifica de oeraciones de la NASA, fue tema de discusión para los expertos a cargo del telescopio James Webb.

En este texto abordamos 3 distintos métodos conocidos para aproximar la resolución del problema y discutimos los resultados obtenidos con cada uno. Presentamos las benevolencias de la heurística del vecino más cercano (NNH), una variazión poco prometedora y un modelo que, inspirado en los rastros de feromonas de la hormigas, combina NNH y aprendizaje.

2. Introducción

El Problema del Viajero o Problema del Vendedor Viajero como se le conoce comúnmente (TSP, por sus siglas en inglés) se define formalmente de la siguiente manera: dado un conjunto finito de ciudades (serán nuestros Nodos) y los costos de viaje entre cada par de ellas (en nuestro caso costo = distancia), se busca determinar la ruta de costo mínimo (distancia mínima) que permita visitar cada ciudad exactamente una vez, regresando a la ciudad de origen.

En términos de complejidad computacional, el TSP es un problema NP-duro (NP-hard). Esto significa que es muy probable que no exista un algoritmo conocido que pueda resolver el problema, en este caso encontrar la ruta óptima, en tiempo polinómico. La dificultad radica en que el número de posibles rutas crece factorialmente con el número de ciudades, lo que hace imposible, incluso con la mejor computadora, evaluar todas las combinaciones posibles a medida que el tamaño del problema aumenta. En particular para el TSP se tiene que, eliminando las rutas redundantes y con n el número de nodos a visitar, existen $\frac{(n-1)!}{2}$ configuraciones que se deberían intentar antes de asegurar con total certexa que una configuración (ruta) es óptima. Para 7 nodos, este número se eleva a 360, para 10 nodos estamos hablando de 181,400. Nos podemos dar una idea de a dónde vamos. Es cuando menos muy costoso explorar todas las configuraciones exhaustivamente y cuando más... mejor dejémoslo ahí. De ahí la necesidad de definir una heurística.

Las heurísticas son estrategias que buscan encontrar soluciones satisfactorias en un tiempo razonable, aunque no garanticen la optimalidad de la ruta. En el contexto del TSP, las heurísticas son útiles porque proporcionan soluciones aproximadas en casos donde los métodos exactos serían computacionalmente muy caros o incluso imposibles de consultar por la capacidad actual de los recursos de cómputo.

El TSP puede representarse como un problema de programación lineal entera (PLE) de la siguiente forma:

$$
\begin{aligned}
\text{Minimizar} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\
\text{Sujeto a} \quad & \sum_{j=1, j\neq i}^{n} x_{ij} = 1 \quad \forall i = 1, \ldots, n \\
& \sum_{i=1, i\neq j}^{n} x_{ij} = 1 \quad \forall j = 1, \ldots, n \\
& \sum_{i\in S}\sum_{j\in S} x_{ij} \leq |S| - 1 \quad \forall S \subset \{1, \ldots, n\}, 2 \leq |S| \leq n-1 \\
& x_{ij} \in \{0, 1\} \quad \forall i, j = 1, \ldots, n
\end{aligned}
$$

Donde:

- c_{ij} representa el costo de viajar de la ciudad i a la ciudad j.
- x_{ij} es una variable binaria que toma el valor 1 si la ruta incluye el trayecto de i a j, y 0 en caso contrario.

Las dos primeras restricciones aseguran que cada ciudad sea visitada exactamente una vez. La tercera restricción, conocida como restricción de subtour, previene la formación de ciclos no deseados que no incluyan todas las ciudades. Esta formulación permite abordar el TSP mediante técnicas de programación lineal entera, aunque su resolución exacta sigue siendo compleja para grandes volumenes de nodos, como es el caso del conjunto de datos que nos fueron asignados.

2.1. Objetivo

El objetivo del proyecto es probar distintas heurísticas y compararlas con el costo del arbol de expansión mínima (AEM en adelante) que representa un límite inferior al costo del TSP óptimo. Rápidamente, el árbol de expansión mínima consiste, en cada iteración, de la concatenación del nodo más cercano a cualquier nodo de la configuración actual. Es fácil ver que esto no necesariamente forma un ciclo y por ende no es solución al TSP.

Para esto, reportaremos los resultados como una tasa entre nuestra solución con cada heurística y el costo del árbol de expansión mínima, con el objetivo de acercarnos al 1.

2.2. Datos

Para la implementación del proyecto, se cuenta con los datos de la ubicación, expresada como coordenadas (x,y), de todas las ciudades de tres países: Qatar, Zimbabwe y Uruguay.

3. Heursíticas

Para aproximar la resolución del problema, se implementaron y probaron tres heurísticas que se discuten a continuación:

1. Heurísitca del vecino más cercano (NNH por sus siglas en inglés)

2. Heurística de los k-vecinos cercanos más probales (variación de NHH)

3. Heurística de la Colonia de Hormigas (que utiliza NHH para resolver el problema)

3.1. Vecino más cercano (NNH)

El método del vecino más cercano comienza con el viajero en una ciudad inicial y selecciona iterativamente la ciudad no visitada más cercana como el siguiente destino. Este proceso se repite hasta visitar todas las ciudades, regresando finalmente a la ciudad inicial.

Algorithm 1 Vecino Más Cercano

Input: $D = \{d_{ij}\}$: matriz de distancias, s: ciudad inicial
Output: **ruta**: lista de ciudades visitadas, C_{total}: costo total de la ruta

Inicializar $n = |D|$, **visitadas** = [Falso] $\times\, n$, **ruta** = $[s]$, $C_{\text{total}} = 0$
Marcar s como visitada **while** *no todas las ciudades estén visitadas* **do**
    Sea i la última ciudad en **ruta**
    Seleccionar próxima = $\arg\min_{j \notin \textbf{visitadas}} d_{ij}$
    Actualizar **visitadas**[próxima] = Verdadero, **ruta**.append(próxima), C_{total} += $d_{i,\text{próxima}}$
Cerrar el ciclo: sumar $d_{\text{último},s}$ a C_{total} y añadir s a **ruta**
return ruta, C_{total}

3.2. Vecino cercano más probable (NNHP)

Variación del método anterior, este método selecciona iterativamente la próxima ciudad a visitar basándose en probabilidades calculadas a partir de las distancias hacia las ciudades no visitadas. En cada iteración, se consideran únicamente los k vecinos más cercanos en un intento de, estocásticamente, aumentar la diversidad en las rutas exploradas. La probabilidad de elegir un vecino es inversamente proporcional a su distancia, favoreciendo así las opciones más cercanas pero con un componente de aleatoriedad.

Algorithm 2 Vecino Más Cercano Probabilístico

Input: $D = \{d_{ij}\}$: matriz de distancias, s: ciudad inicial, k: número de vecinos a considerar
Output: **ruta**: lista de ciudades visitadas, C_{total}: costo total de la ruta

Inicializar $n = |D|$, **visitadas** = [Falso] $\times\, n$, **ruta** = $[s]$, $C_{\text{total}} = 0$
Marcar s como visitada **while** *no todas las ciudades estén visitadas* **do**
    Sea i la última ciudad en **ruta**
    Calcular $V = \{j : \text{no visitado}\}$ y sus distancias $\{d_{ij} \mid j \in V\}$
    Ordenar V según d_{ij} y seleccionar los k más cercanos
    Calcular probabilidades $p_j = \frac{1/d_{ij}}{\sum_{l \in V}(1/d_{il})},\ \forall j \in V$
    Seleccionar la próxima ciudad j usando p_j
    Actualizar **visitadas**$[j]$ = Verdadero, **ruta**.append(j), C_{total} += d_{ij}
Cerrar el ciclo: sumar $d_{\text{último},s}$ a C_{total} y añadir s a **ruta**
return ruta, C_{total}

3.3. Colonia de hormigas (HC)

El método de colonia de hormigas utiliza un enfoque bioinspirado pues Se basa en simular el comportamiento de una colonia de hormigas que exploran rutas posibles mientras depositan feromonas en los caminos más prometedores. Las feromonas se refuerzan proporcionalmente a la calidad (menor costo) de la ruta, mientras que se evaporan con el tiempo para evitar la sobresaturación. Cada iteración involucra múltiples hormigas generando rutas, y el sistema converge hacia una solución cercana a la óptima tras varias iteraciones.

Algorithm 3 Colonia de Hormigas para el TSP

Input: $D = \{d_{ij}\}$: matriz de distancias, m: número de hormigas, t: iteraciones, ρ: tasa de evaporación de feromonas

Output: $\mathbf{ruta}^*$: mejor ruta encontrada, C^*: costo de la mejor ruta

Inicializar $n = |D|$, $\tau_{ij} = 1,\ \forall (i,j)$, $C^* = \infty$ **for** *iter* $= 1$ ***to*** t **do**

  Inicializar rutas $\{\mathbf{ruta}_k\}_{k=1}^m$ y costos $\{C_k\}_{k=1}^m$

  for $k = 1$ ***to*** m *(cada hormiga)* **do**

    Seleccionar aleatoriamente ciudad inicial s_k

    Construir $\mathbf{ruta}_k$ usando probabilidades $p_{ij} = \frac{\tau_{ij} \cdot (1/d_{ij})^\beta}{\sum_{l \notin \mathbf{ruta}_k} \tau_{il} \cdot (1/d_{il})^\beta}$

    Calcular $C_k = \sum_{(i,j) \in \mathbf{ruta}_k} d_{ij}$

  Evaporar feromonas: $\tau_{ij} \leftarrow (1 - \rho) \cdot \tau_{ij},\ \forall (i,j)$

  Actualizar feromonas: $\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^m \frac{1}{C_k},\ \forall (i,j) \in \mathbf{ruta}_k$

  if $\min(C_k) < C^*$ **then**

    Actualizar $C^* = \min(C_k)$, $\mathbf{ruta}^* = \mathbf{ruta}_{\text{argmin}(C_k)}$

return $\mathbf{ruta}^*, C^*$

Una vez definidos los métodos con los que estaremos identificando las rutas, proponemos un método de optimización local utilizado para mejorar nuestras soluciones obtenidas con los 3 algoritmos presentados en la sección anterior.

3.4. K-Opt

Los métodos k-opt son métodoso locales que consisten en iterativamente eliminar cruces en una ruta con el objetivo de reducir su longitud total y conducir a una ruta más eficiente. La K proviene del número de conexiones que se intercambian en cada iteración, en nuestro caso, por la complejidad de cómputo, serán 2 pues el método es exhaustivo al probar todas las posibles combinaciones de k aristas a intercambiar.

Algorithm 4 Método 2-Opt para el TSP

Input: **ruta**: una solución inicial, $D = \{d_{ij}\}$: matriz de distancias
Output: **ruta***: una solución mejorada

Inicializar **ruta**$^* =$ **ruta**, $C^* =$ costo(**ruta**)
repeat
  for $i = 1$ ***to*** $n - 1$ **do**
    for $k = i + 1$ ***to*** n **do**
      Generar nueva ruta **ruta**$'$ intercambiando los segmentos entre i y k
      Calcular $C' =$ costo(**ruta**$'$)
      if $C' < C^*$ **then**
        Actualizar **ruta**$^* =$ **ruta**$'$, $C^* = C'$
until *no se encuentre mejora*;
return **ruta***

4. Resultados

Los resultados de la implementación de los 4 métodos se discuten a continuación y se presentan en la siguiente tabla, resultados obtenidos promediando 10 rutas obtenidas a partir de 10 distintos nodos de inicio:

Algoritmo	Costo Promedio	Tasa vs AEM	Costo Opt-2	Tasa Opt-2
NNH	100,059.4	1.432	85,942	1.229
NNHP	124,078.4	1.775	88,489	1.266
HC	96,812.3	1.385	85,549	1.224

Cuadro 1: Comparación de Algoritmos según Costo Promedio y Tasa, utilizando las ciudades de Uruguay

Podemos ver que, aunque los resultados con la heurística del vecino más cercano son buenos, el resultado de combinar el vecino más cercano con la matriz de feromonas creada en el método del hormiguero son aún mejores.

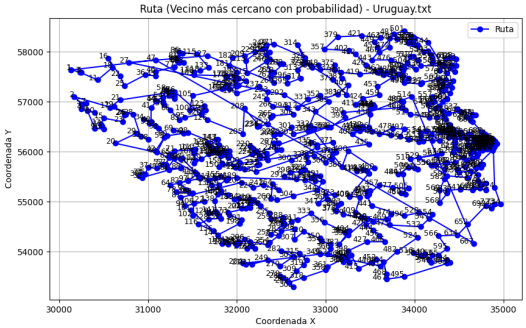

Figura 1: Resultados de aplicar NNHP a los datos de Uruguay

El método de los vecinos cercanos más probables resulto no ser una muy buena heurística,

viendo la gráfica de resultados (Figura 1) podemos ver saltos poco "lógicos.entre ciudades comparándolo con el método no estocástico (Figura 2).

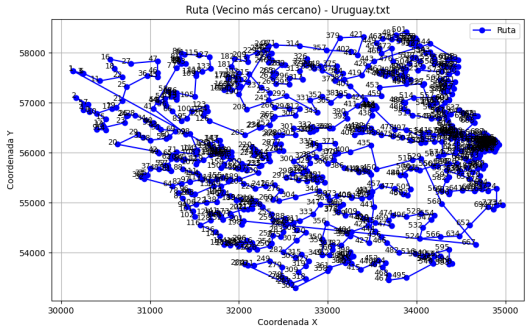

Figura 2: Resultados de aplicar NNH a los datos de Uruguay

Ahora, discutamos la eficiencia en tiempos de cada uno de los métodos. La siguiente tabla se obtuvo para cada método, para cada país y después de promediar los resultados de 60 nodos iniciales aleatorios.

Algoritmo	Tiempo Promedio	Varianza en Tiempo	País
NNH	0.003180	3.027885×10^{-7}	Qatar
PNNH	0.014882	1.623968×10^{-5}	Qatar
HC	0.548215	1.881954×10^{-2}	Qatar
NNH	0.059609	3.284690×10^{-4}	Uruguay
PNNH	0.193515	2.607641×10^{-3}	Uruguay
HC	8.468320	3.477179×10^{-1}	Uruguay
NNH	0.074003	1.870237×10^{-5}	Zimbabwe
PNNH	0.276841	1.928881×10^{-3}	Zimbabwe
HC	12.961262	1.485684×10^{-3}	Zimbabwe

Cuadro 2: Resultados de tiempo promedio y varianza por algoritmo y país

Como podemos ver, el método de NNH es el más rápido de todos, con los menores tiempos en general para cada país. Le sigue el método PNNH. Por su cuenta, el tiempo menos eficiente en cuestión de tiempo, a pesar de ser el más eficiente en términos de resultados, es el método del hormiguero, que consigue los peores resultados en tiempo, con los tiempos de mayor varianza y con una duración de 12 segundos en promedio por iteración para Zimbabwe por ejemplo.

5. Mejoras

La metodología utilizada tiene un par de mejoras a considerar. La primera y más obvia, es el uso de estructuras de datos más eficientes a la hora de programar. Actualmente se utilizan principalmente listas como estructuras para almacenar los nodos y otros resultados claves. Un cambio natural es el paso de las listas a los arreglos de numpy o inclusive, en caso de requerir utilizar métodos de busqueda más avanzados, a los diccionarios.

Otra mejora es el ajuste de los hiperparámetros de la heurística del hormiguero. Esto nos permitirá darle pesos diferentes a la distancia y a las "feromonas"que dejan nuestras hormigas viajeras.

Finalmente, nos quedamos la espina de no haber probado métodos genéticos y métodos de aprendizaje de máquina que aparecieron en muchos momentos de nuestra investigación como métodos que están actualmente siendo discutidos en la academia para aproximar este famoso problema.

6. Conclusiones

A pesar de las posibles mejoras, concluimos que el proyecto fue exitoso pues cumplió con el objetivo al probar 3 distintas heurísticas y además, al probar un método que nos permitió reducir la tasa respecto del AEM, opt-2.

Además, cumplió con los objetivos de la materia pues debimos representar el problema como uno de PLE, identificar las debilidades de métodos analíticos convencionales y aproximar la solución mediante la creación de heuristicas.

En términos técnicos, concluimos que aunque el mejor método es el método del hormiguero mejorado con opt-2, este es el método más costoso computacionalmente por su naturaleza. Por lo tanto, recomendariamos el uso del método NNH pues es apenas "peor"que el método del hormiguero, pero es casi 170 veces más rápido que el método del hormiguero para nuestros datos de Uruguay y Zimbabwe.

Referencias

[1] E Lawler et al. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley-Interscience, 1985.

[2] César Rego et al. «Traveling salesman problem heuristics: Leading methods, implementations and latest advances». En: (jun. de 2011). URL: https://www.sciencedirect.com/science/article/abs/pii/S0377221710006065.

[3] Alireza Resvanian, S Mehdi Vahidipour y Ali Sadollah. «An Overview of Ant Colony Optimization Algorithms for Dynamic Optimization Problems». En: (jun. de 2023). URL: https://www.researchgate.net/publication/371560687_An_Overview_of_Ant_Colony_Optimization_Algorithms_for_Dynamic_Optimization_Problems.