Facultad de Ingeniería
Departamento de Ingeniería Industrial

Tarea Computacional 2
El problema del Vendedor Viajero y su solución mediante Python con PuLP

Integrantes
Luis Galvez Paes - 2019457594
Oscar Castillo Vega - 2020453209

1 Contexto

El barco pirata del Capitán Oscarius Lulang, "La Furia Carmesí", se hizo a la mar en una audaz aventura. ¿Su misión? Trazar un curso a través de una serie de puertos, cada uno marcado por enormes desafíos marítimos. A medida que el Capitán Lulang navegaba por las aguas abiertas, se encontraba con los vientos y las corrientes siempre cambiantes que hacían que su viaje estuviera muy complicado.
En este enigma náutico, llegar a un puerto desde otro no garantizaba un regreso rápido. Tome, por ejemplo, el viaje desde un punto de partida no revelado hacia un destino enigmático. El viaje a estos destinos misteriosos era más rápido gracias a los vientos favorables, pero el viaje de regreso, lleno de corrientes contrarias, llevaba más tiempo.
La tripulación de "La Furia Carmesí" se maravillaba de su astuto capitán, Oscarius Lulang, que tenía una habilidad asombrosa para navegar por las aguas impredecibles con precisión. Su misión era simple pero compleja: asegurarse de visitar cada puerto enigmático solo una vez y minimizar el tiempo total de viaje, todo mientras lidiaba con los caprichos de las fuerzas de la naturaleza. La reputación del Capitán Lulang como un pirata intrépido y astuto crecía con cada puerto que conquistaba, lo que hacía que su leyenda como maestro de los mares fuera aún más celebrada en alta mar.
Pero el secreto del Capitán Oscarius era que usaba un modelo de minimización para calcular sus rutas, el Rey de España ha solicitado a alumnos de la Universidad de Concepción que identifiquen que formulación usa el capitán, para así poder determinar sus rutas y capturarlo de una vez por todas.

2 Problema

El objetivo es encontrar una ruta que minimice el tiempo total de viaje, cumpliendo con las restricciones de visitar cada puerto exactamente una vez. El grafo dirigido que representa los distintos puertos y los pesos de los caminos se representa como una matriz, donde la posición $[i, j]$ representa el costo de viajar de i a j, con las posiciones $i = j$ siendo igual a 0 al representar un camino de un puerto a si mismo. La posición [0,0] del arreglo representa el puerto hogar de La Furia Carmesí a donde deben volver después de la travesía. Este problema está contextualizado dentro de los problemas del tipo ATSP, ya que como fue definido anteriormente las corrientes y vientos son un factor importante en los tiempos de viaje, por lo tanto ir de un puerto i a j no toma necesariamente el mismo tiempo que ir de un puerto j a i.

3 Variables de decisión

$$x_{ij} = \begin{cases} 1 & , \text{ si la arista i,j es parte de la ruta de La Furia Carmesí} \\ 0 & , \text{ e.o.c} \end{cases}$$

4 Parámetros del modelo

- C = Matriz de costos (Tiempos de viaje)
- V = Vértices (Puertos)
- A = Aristas (Caminos entre puertos)

5 Modelos Matemáticos

A continuación se define el modelo de minimización con las respectivas implementaciones. Para la función objetivo se considera c_{ij} como el costo de ir del puerto i al puerto j y x_{ij} la variable binaria anteriormente definida.

5.1 Formulación de Dantzig–Fulkerson–Johnson

En esta implementación las restricciones (2) y (3) representan que el grado de entrada y salida de un nodo tiene que ser siempre 1, en el contexto del problema esto indica que cada puerto es solo una vez origen y una vez destino dentro de la ruta. Luego la implementación incluye la restricción (4), que representa la "eliminación de subtours", con S un subconjunto de V, esta retricción define que no deben haber ciclos innecesarios dentro de la ruta.

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{1}$$

$$\text{s.a} \sum_{i=1}^{n} x_{ij} = 1 \quad \forall i \in V \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j \in V \tag{3}$$

$$\sum_{i\in S}\sum_{j\in S} x_{ij} \leq |S| - 1, \quad S \subset V, S \neq \emptyset \tag{4}$$

$$x_{ij} \in \{0,1\} \quad \forall i,j \in V \tag{5}$$

5.2 Formulación de Miller-Tucker-Zemlin

Para esta implementación se mantiene la forma del modelo anterior pero se modifica la restricción (4) con esta nueva restricción (6), la cual introduce una nueva variable u_i que representa el orden del vértice i en la ruta óptima. Esto también cumple la función de eliminar subtours, pero de una manera más eficiente.

$$u_i - u_j + (n-1)x_{ij} \leq n-2, \quad \forall i, j \in \{2, \ldots, n\} \tag{6}$$

5.3 Formulación de Gavish-Graves

En esta última implementación también se mantiene la forma del modelo inicial, pero modificando la restricción (4) por las nuevas restricciones (7) y (8). En este caso se introduce la variable g_{ij}, la cual representa el número de arcos en el camino desde el vértice 1 hasta el arco (i, j) en la ruta óptima. Esta nuevas restricciones también son engargadas de la elminación de subtours.

$$\sum_{j=1}^{n} g_{ij} - \sum_{j=2}^{n} g_{ji} = 1, \quad \forall i \in V \tag{7}$$

$$0 \leq g_{ij} \leq (n-1)x_{ij} \quad \forall j \in V, \forall i \in \{2, \ldots, n\} \tag{8}$$

6 Resolución instancias

6.1 Creación de instancias

Las diez instancias a resolver por los modelos han sido creadas mediante un script en Python el cual mantiene el formato "instancia.atsp", pero a la vez permite crear matrices de los tamaños requeridos. Para este caso se utilizaron matrices cuadradas de dimensiones 8, 9 y 10 las cuales cumplen los requerimientos de una matriz de instancia ATSP.

6.2 Resultados

Se probó cada algoritmo con cada una de las instancias y se grabó la duración de calculo, expresado en la siguiente tabla.

	inst1	inst2	inst3	inst4	inst5	inst6	inst7	inst8	inst9	inst10
DFJ	0	0.06	0.01	0.02	0.02	0.19	1.49	0.05	0.03	0.05
GG	0.01	0.07	0.02	0.03	0.03	0.07	0.07	0.01	0.01	0.01
MTZ	0.02	0.07	0.01	0.04	0.09	0.16	0.12	0.02	0.02	0.08

Table 1: Resultados de experimentación mostrando duración de calculo en segundos de CPU

- Duración promedio de DFJ, $0.192s$
- Duración promedio de GG, $0.033s$
- Duración promedio de MTZ, $0.063s$

Se puede ver que hay casos donde el DFJ es mas lento que el GG y MTZ, calculando los promedios podemos apreciar que en esta serie de instancias, el GG es el más rápido para realizar la minimización.

7 Conclusiones

La eliminación de subtours en el modelo DFJ, aunque es efectiva para garantizar que la solución final sea una ruta completa y sin ciclos más pequeños, introduce una carga computacional significativa. A medida que aumenta el tamaño de la matriz que representa los tiempos de viaje entre los puertos, la complejidad computacional de la eliminación de subtours se vuelve más pronunciada, lo que contribuye a un aumento drástico en el tiempo de resolución. En contraste, el modelo MTZ y GG muestran un rendimiento más eficiente en términos de tiempo de resolución.

Por lo tanto se puede concluir que el Capitán Oscarius Lulang esta utilizado la formulación de Gavish-Graves, con esta información el Rey de España te ha recompensado con una práctica no remunerada en su castillo.