UNIDAD PROFESIONAL INTERDISCIPLINARIA
DE INGENIERÍA CAMPUS ZACATECAS

UNIDAD DE APRENDIZAJE:

ANÁLISIS DE ALGORITMOS

ALUMNO:

ERNESTO JASSIEL HERNÁNDEZ BASURTO

TRAVEL SALESMAN PROBLEM DINÁMICO

FECHA:

7 DE NOVIEMBRE DE 2019

Introducción

El algoritmo *Traveling Salesman Problem* o TSP, puede ser ejemplificado de la siguiente manera; una persona tiene que viajar a través de un conjunto de ciudades, cada ciudad tiene una distancia especificada entre cada una y no puede visitar más de una vez la misma ciudad, con este algoritmo se busca cumplir lo anterior, pero en la distancia más corta posible. Resolver el problema TSP, es muy similar a encontrar circuito mas corto en un problema Hamiltoniano.

El problema en esta práctica se busca resolver con programación dinámica, este método de programación funciona resolviendo el problema partiendo la solución en un conjunto de pasos o fases, de esta manera podemos ver la solución del problema desde una serie de decisiones interrelacionadas.

En resumen, la programación dinámica busca remover una pequeña parte del problema a cada paso que se resuelva, y usar estos pequeños resultados para obtener la solución final.

Desarrollo

Para aplicar la programación dinámica al problema TSP, podemos realizar los siguientes pasos:

Teniendo la siguiente matriz de distancias:

0	12	11	16
15	0	15	10
8	14	0	18
9	11	17	0

Estando marcada en amarillo la ciudad de partida.

Paso 1:

Usando la siguiente función: $f(i,\varnothing) = c_{i,1}, \quad 2 \le i \le n$

Donde *i* representa la ciudad de partida y O las siguientes ciudades posibles desde ahí.

Podemos calcular y obtener lo siguiente:

$$f(2,\varnothing) = c_{21}, = 15$$

$$f(3,\varnothing) = c_{31}, = 8$$

$$f(4,\varnothing) = c_{41}, = 9$$

Paso 2:

Usando la siguiente función

$$f(i,s) = \min_{j \in S} \left\{ c_{ij} + f(j, S - \{j\}) \right\}$$

Calculamos para |*S*| = *1*, y luego podemos obtener para |*S*| = *2*, esto hasta |S| =n-1

Obtenemos para **|*S*| = *1***

$$f(2,\{3\}) = \{c_{23} + f(3,\varnothing)\} = 15 + 8 = 23$$

$$f(3,\{2\}) = \{c_{32} + f(2,\varnothing)\} = 14 + 15 = 29$$

$$f(4,\{2\}) = \{c_{42} + f(2,\varnothing)\} = 11 + 15 = 26$$

$$f(2,\{4\}) = \{c_{24} + f(4,\varnothing)\} = 10 + 9 = 19$$

$$f(3,\{4\}) = \{c_{34} + f(4,\varnothing)\} = 18 + 9 = 27$$

$$f(4,\{3\}) = \{c_{43} + f(3,\varnothing)\} = 17 + 8 = 25$$

Para |S| = 2:

$$f(2,\{3,4\}) = \min\left\{c_{23}+f(3,\{4\}),c_{24}+f(4,\{3\})\right\}$$
$$= \min\ \{15+27,10+25\}$$
$$= \min\ \{42,35\} = 35$$

$$f(3,\{2,4\}) = \min\left\{c_{32}+f(2,\{4\}),c_{34}+f(4,\{2\})\right\}$$
$$= \min\ \{14+19,27+11\}$$
$$= \min\ \{33,38\} = 33$$

$$f(4,\{2,3\}) = \min\left\{c_{42}+f(2,\{4\}),c_{43}+f(3,\{2\})\right\}$$
$$= \min\ \{11+22,17+29\}$$
$$= \min\ \{33,46\} = 33$$

Paso 3:

Guiandonos por la ecuación

$$f\left(1,v-\{1\}\right) = \min_{2<k<n}\left\{c_{1k}+f(k,V-\{1,k\})\right\}$$

Se calcula lo siguiente:

$$f(1,\{2,3,4\}) = \min\left\{c_{12}+f(2,\{3,4\}),c_{13}+f(3,\{2,4\}),c_{14}+f(4,\{2,3\})\right\}$$
$$= \min\ \{12+35,11+33,16+33\}$$
$$= \min\ \{47,44,49\} = 44$$

Obteniendo aquí que el camino más corto para esta matriz tiene una distancia total de 44.

Paso 4:

Finalmente, para obtener el camino, primeramente, hay que señalar que el valor del camino más corto es de *44,* valor obtenido de 11 + 33 y tomando en cuenta los cálculos hechos anteriormente.

$$c_{13} + f(3,\{2,4\}$$

El camino es:

1 -> 3 -> 2 -> 4 = 44

El código realizado puede ser encontrado aquí:

https://github.com/jass1302/TSP-Mochila-Dinamicos

Conclusión

En comparación a resolverlo por fuerza bruta, los requerimientos y tiempo de ejecución de esta forma dinámica son menores, sin embargo, requiere de más memoria para resolver el problema y está crece dadas las *n-ciudades* que puedan necesitar resolverse.

Referencias:

https://github.com/evandrix/SPOJ/blob/master/DP_Main112/Solving-Traveling-Salesman-Problem-by-Dynamic-Programming-Approach-in-Java.pdf