## TPComplejidad

### Tales Salesman Problem

#### Introducción

El problema del vendedor viajero responde a la siguiente pregunta: dada una lista de ciudades y las distancias entre cada par de ellas.

¿Cuál es la ruta más corta posible que visita cada ciudad exactamente una vez y al finalizar regresa a la ciudad origen? Este es un problema

NP-Hard(conjunto de los problemas de decisión) dentro en la optimización combinatoria.  
El problema fue formulado por primera vez en 1930 y la motivación que encontramos en el problema es que es uno de los problemas de optimización más estudiados. Es usado como prueba para muchos métodos de optimización. Aunque el problema es computacionalmente complejo, una gran cantidad de heurísticas y métodos exactos son conocidos, de manera que, algunas instancias desde cien hasta miles de ciudades pueden ser resueltas. Encontrar una solución con los conocimientos adquiridos hasta el momento en el curso de complejidad algorítmica es un reto para los alumnos, y ayudara a desarrollar nuestro pensamiento crítico al momento de solucionar problemas.

Se buscaran soluciones para conjuntos de datos de distintos tamaños, 8 para se especificos, los cuales varian entre 25 a 145 000 datos. De esta manera se demostrara como escara el tiempo de los algoritmos implementados dependiendo de la cantidad de puntos los cuales deben ser visitados.

#### Objetivos

* Encontrar la mayor cantidad de conexiones con las mínimas distancias de un nodo 'x' a un nodo 'y' en un tiempo optimo para las 8 bases de datos.
* Encontrar distintos tipos de algoritmos usando los temas visto en la segunda parte del curso de Complejidad Algoritmica: Kruscal, Prim, Programación Dinámica, Bellman-Ford Floyd-Warshall, Johnson o Algoritmos Greedy..
* Desarrollar algoritmos que nos permitan encontrar una solucion (no necesariamente la mas optima) de las 8 bases de datos con las que se trabaja.
* Demostrar que las propuestas presentadas son escalables con respecto a la cantidad de datos con los que se trabaje.

#### Marco teorico

Algoritmo 1:

-BellmanFord: «El algoritmo de Bellman-Ford (algoritmo de Bell-End-Ford) produce el camino más corto en un grafo dirigido ponderado (en el que el peso de ciertas aristas puede ser negativo). Con lo que el **Algoritmo**Bellman-Ford en general se emplea cuando hay aristas con peso negativo. Este algoritmo fue desarrollado por Richard Bellman, Samuel End y Lester Ford.» (Paraisodigital.org, 2018)

-Algoritmo Greedy:»El propósito de un algoritmo voraz es encontrar una solución, es decir, una asociación de valores a todas las variables tal que el valor de la función objetivo sea óptimo. Para ello sigue un proceso secuencial en el que a cada paso toma una decisión (decide qué valor del dominio le ha de asignar a la variable actual) aplicando siempre el mismo criterio (función de selección). La decisión es localmente óptima, es decir, ningún otro valor de los disponibles para esa variable lograría que la función objetivo tuviera un valor mejor, y luego comprueba si la puede incorporar a la secuencia de decisiones que ha tomado hasta el momento, es decir, comprueba que la nueva decisión junto con todas las tomadas anteriormente no violan las restricciones y así consigue una nueva secuencia de decisiones factible.» (Universidad Politecnica de Cataluño, 2008)

El algoritmo consiste en tomar 2 de las de los nodos que esten lo mas lejos posible, y estableciendo uno como inicial. Luego utilizando BellmanFord se encuentra la ruta mas efectiva del nodo inicial hasta su nodo mas lejano. Una vez trasada esta ruta se usa un algoritmo greedy partiendo del nodo mas alejado del inicial antes establecido, este va escogiendo su siguiente coneccion que este lo mas cerca al nodo antes especificado hasta que encuentre el nodo inicial. En caso que el algoritmo greddy encuentre una situacion en la que no puede continuar porque sus conecciones han sido todas ya visitadas este usara backtracking para encontrar otra ruta posible.

Algoritmo 2:

Algoritmo 3:

#### Tiempo asintótico

##### Algoritmo 1:

BellmanFord:

T(n)=n x n

Greedy:

-En el mejor de los casos:

T(n)=n

-En el peor de los casos:

T(n)=n x n

Por lo tanto el algoritmo creado tendra un tiempo maximo de:

T(n)=(n x n)+(n x n)

Algoritmo 2:

Algoritmo 3:

#### Conclusiones:

Utilizar Bellmand Ford es una opcion factible para para encontrar una ruta entre x a y, sin embargo como no se tienen rutas negativas entonces no es necesario y un algoritmo como UCS puede ser mucho mas eficiente en cuanto al tiempo asindotico.

Hay algortmos que pueden servir para crear algoritmos que encuentren una buena solucion, como algoritmos que encuentran rutas mas cortas, programacion dinamica para reducir el tiempo, algoritmos greedy dependiendo como sean los datos, sin embargo no se tiene un algoritmo que por si solo pueda dar una solucion, siempre tiene que cambiarse varias partes del mismo o fucionarlos. Ademas los tiempos son muy grandes y no encuentran la mejor solucion solo una buena solucion.

Utilizando las diferentes bases de datos propuestas se puede notar como mientras mas grande es la base de datos (mas grande es el tiempo de ejecucion del problema. Esto se debe a que los algoritmos usados suelen tener un tiempo asindotico exponencial lo cual no es nada conveniente.

**Bibliografia:**

Paraisodigital.org. (2018). ıllı Algoritmo de Bellman-Ford : Que es, definición y significado. [online] Revisado en: <https://www.paraisodigital.org/internet/ll-algoritmo-de-bellman-ford-que-es-definicion-y-significado-descargar-videos-y-fotos.html> [Visto el 18 Nov. 2018].

Cópiala y pégala en tu documento. La ficha bibliográfica es :   
Abad, Maria. (2008). ALGORITMOS VORACES. noviembre 17, 2018, de Universidad Politecnica de Cataluña Sitio web: <http://www.cs.upc.edu/~mabad/ADA/curso0708/GREEDY.pdf>