Práctica 4“Algoritmos de Vuelta Atrás (Backtracking) y de Ramificación y Poda (Branch and Bound)”

Parte II: El problema del viajante de comercio

Realizado por:

Baeza Álvarez, Jesús
García Moreno, Jorge
López Maldonado, David
Rodriguez Calvo, Jose Manuel
Sánchez Molina, Alejandro

ÍNDICE

1. Objetivo del problema

El objetivo de esta práctica era resolver el problema del TSP (Problema del viajante de comercio).

Para ello hemos implementando un algoritmo que utiliza la técnica de ramificación y acotación, cuyo funcionamiento lo explicamos en el próximo apartado.

Para la visualización de los datos se utilizó gnuplot. Los ficheros de datos utilizados han sido los mismos empleados en la práctica 3.2 (proporcionados por el profesor en la plataforma decsai), pero con la particularidad de que hemos empleado los ficheros con el menor número de ciudades debido al tipo de algoritmo implementado.

También se ha usado un programa para la reordenación de los puntos obtenidos.

El programa nos proporciona como salida el mejor camino a seguir, la distancia, nodos expandidos, número de veces que se realiza la poda, tamaño máximo de la cola con prioridad de nodos vivos y tiempo total empleado.

2. Explicación algoritmo

Lo primero que se hace es cargar las ciudad y crear la matriz de distancia.

Se tiene una clase llamada ciudad, que representa cada nodo, esta estructura tiene asociada un vector, que es el recorrido que lleva el nodo, una cota inferior que es la estimación del nodo y la distancia, que es la distancia que lleva asociado el vector del recorrido del nodo.

Primeramente se inicializa la cota global a un valor muy alto y el primer nodo del árbol que empieza en la ciudad 1. Con ese nodo inicial, se llama a nuestro algoritmo BranchBound.
La estructura de BranchBound es la siguiente, primero se saca el nodo más prometedor de una priority queue(al principio solo está el nodo inicial), después se realiza una comprobación inicial para ver si es hoja o no es hoja.

Si no es hoja y su cota inferior es menor que cota global ese nodo se expande y los sucesores se inicializan (la cota inferior se calcula con la distancia más un cálculo optimista de lo que le queda por recorrer, en nuestro caso escoger la mejor arista de cada ciudad no visitada), una vez generados los hijos, estos se introduce en la priority queue solo si su cota inferior es menor que la cota global.

Si el nodo no es hoja y su cota inferior es mayor que la cota global, dejamos de expandir dicho nodo y podamos.

Si es nodo hoja se comprueba si su valor de distancia mejora la cota global, si mejora ese recorrido se almacenan sus datos como el mejor recorrido hasta el momento, modificando la cota global que teníamos a ese nuevo valor.

BranchBound se vuelve a llamar mientras haya nodos en priority queue.

3. Datos obtenidos y conclusiones.

En este apartado vamos a presentar un estudio comparativo entre distintos resultados obtenido.

Para ello hemos probado con 10 ficheros (de distinto tamaño) obteniendo un tiempo y distancia para cada uno de ellos.

Para la obtención de los datos de las ciudades con las que trabaja nuestro algoritmo, hemos partido de los ficheros que se nos entregó en la práctica 3, solo que hemos escogido de entre algunos y le hemos quitado una cantidad de ciudades hasta dejarlas a un máximo de 9 ciudades, debido a que un número mayor nos provocaba fallos de memoria.

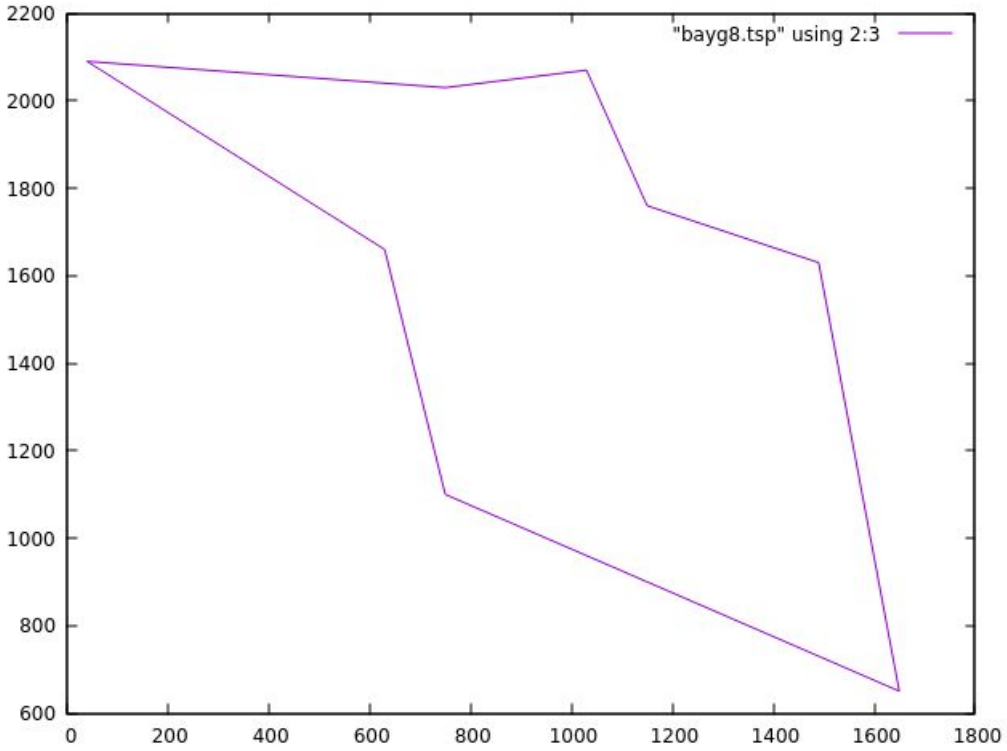
La forma de comparación que utilizaremos será mediante una tabla en la que se plasman los resultados obtenidos.

Fichero	Tiempo	Distancia	Nodos exp.	Nº veces poda	Tamaño max. cola
att6.tsp	0,000325	18061,1	99	55	39
bayg8.tsp	0,001865	4992,78	644	395	353
berlin5.tsp	4.1e-05	2314,55	37	10	19
eil7.tsp	0,000484	130,31	331	152	164
lin5.tsp	7,8e-05	2459,85	31	14	11
pr4.tsp	4,4e-05	8635,54	11	3	4
st9.tsp	0,006869	229,45	3511	4330	864
ulysses9.tsp	0,016139	46,106	6436	8043	2419
ulysses10.tsp	0,036392	46,55	17149	27406	6784

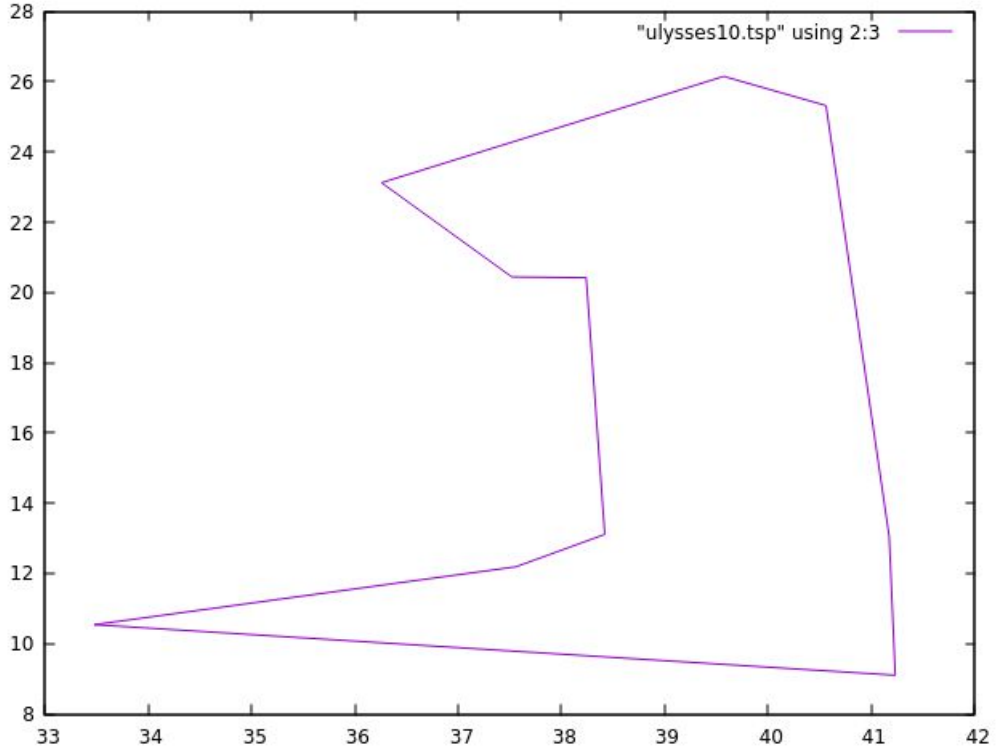
Como se observa en la tabla, a medida que el tamaño del fichero aumenta, el tiempo es mayor. Justamente ocurre lo contrario para los ficheros con menos tamaño. También se observa cómo a medida que el tamaño es mayor el resto de valores también aumenta.

En la página siguiente mostramos algunas de las gráficas obtenidas.

bayg8.tsp

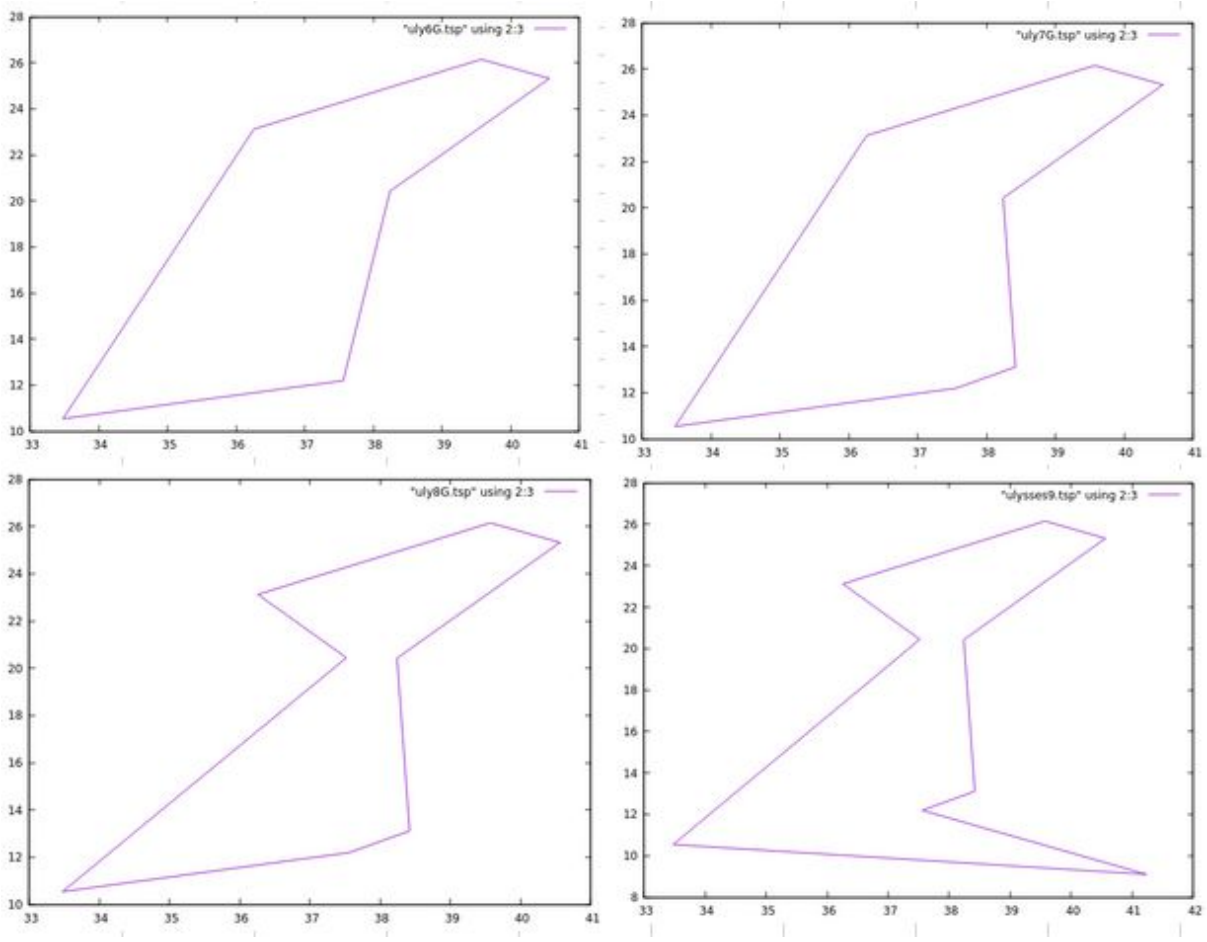


ulysses10.tsp



También hemos comparado el archivo original “ulysses16.tsp” con distintos tamaños, para observar cómo afecta al dibujo de la gráfica, tiempo y distancia el que tenga más o menos ciudades asociadas el archivo. El resultado de esta comparación ha sido el siguiente:

Tamaño	Tiempo	Distancia
6	0.000281	36.74
7	0.001124	37.05
8	0.005081	37.82
9	0.016139	46.10



Como se podía intuir, a medida que aumenta el número de ciudades, tanto el tamaño como el tiempo también aumentan.