Prácticas Algorítmica

Práctica 4 : Ramificación y poda (Branch&Bound)

1. Enunciado del problema a tratar: El viajante del comercio.

Lo único que se nos pide en este problema, que ya se resolvió en prácticas anteriores, es buscar un algoritmo de ramificación o poda (Branche&Bound) de tal manera que cumpla las expectativas de este problema, que recordamos que son:

- *Dado un conjunto de ciudades y una matriz con las distancias entre todas ellas:*
  - *Un viajante debe recorrer todas las ciudades exactamente una vez, regresando al punto de partida, de forma tal que la distancia recorrida sea mínima.*
  - *Más formalmente, dado un grafo G, conexo y ponderado, se trata de hallar el ciclo hamiltoniano de mínimo peso de ese grafo.*

Para emplear un algoritmo de ramificación y poda, es necesario utilizar una cota inferior, un valor menor o igual que el verdadero coste de la mejor solución (la de menor coste) que se puede obtener a partir de la solución parcial en la que nos encontramos.

2. Introducción general al algoritmo Ramificación y Poda (Branche&Bound)

El esquema de Ramificación y Poda es el algoritmo de búsqueda más eficiente en espacios de búsqueda que sean árboles.
En este algoritmo se diferencian una serie diferente de nodos en el espacio de búsqueda: nodo vivo, muerto o en expansión.

- *Un nodo vivo es un nodo factible y prometedor (un nodo es prometedor si con la información que tenemos podemos expandirlo y sacar una solución mejor a la mejor solución en curso) del que no se han generado todos sus hijos.*
- *Un nodo muerto es un nodo del que no van a generarse más hijos por alguna de las tres razones siguientes:*
  - *Ya se han generado todos sus hijos.*
  - *No es factible.*
  - *No es prometedor.*

En cualquier instante del algoritmo puede haber muchos nodos vivos, y muertos pero solo puede haber uno en expansión que es aquel que está generando sus hijos en ese instante.
El algoritmo se encarga de detectar en qué ramificación las soluciones dadas ya no están siendo óptimas, para que se <<pode>> esa rama del árbol y así no se malgastan recursos y procesos en casos que se alejan de la solución óptima.
A cada hijo se le hace el siguiente análisis:

- *Sea x el hijo que se está analizando:*
  - *Si x no es factible, entonces x pasa a ser un nodo muerto.*
  - *Si x es factible y tiene un coste(x) peor que el de la mejor solución en curso, x pasa a ser un nodo muerto y nunca más se volverá a considerar.*
  - *Si x es factible y tiene un coste(x) mejor que el de la mejor solución en curso pero x no es solución, entonces x se inserta con prioridad coste(x) en la cola de prioridad de nodos vivos.*
  - *Si x es factible y tiene un coste(x) mejor que el de la mejor solución en curso y x es solución entonces pasa a ser la nueva mejor solución en curso. Además se revisan todos los nodos de la cola de prioridad de nodos vivos y se eliminan de ella todos aquellos que tengan una prioridad peor o igual que coste(x).*
- *Una vez generados y analizados todos los hijos del nodo en expansión, éste se convierte en un nodo muerto y se vuelve a repetir el proceso.*
- *El proceso acaba cuando la cola de prioridad está vacía. En ese momento la mejor solución en curso se convierte en la solución óptima del problema.*

3. Solución al problema del viajante del comercio con Branche&Bound.

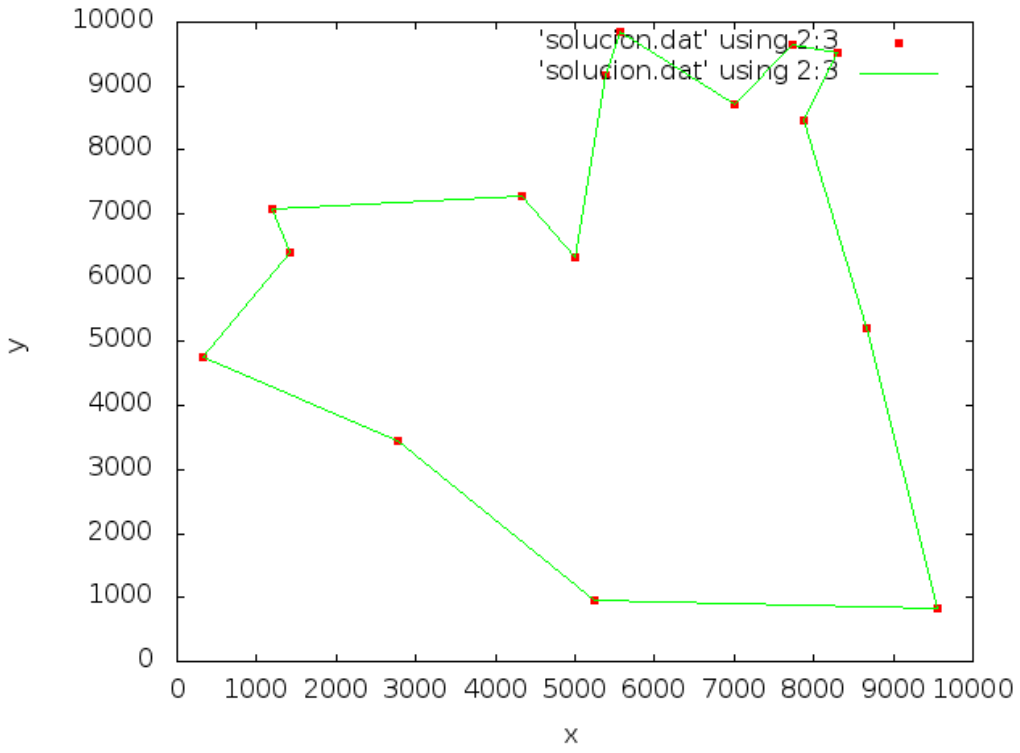
Para la solución de esta práctica creamos una estructura llamada "nodo" la cual representará a cada ciudad, por lo que habrá tantas como número de ciudades les hayamos pasado a través del fichero, y los crea.
Para su representación y fácil localización existe dos ejes el "X" y el "Y" los cuales contienen valores aleatorios entre 0 y 10000.

Para realizar la poda, guardamos en todo momento en un vector de nodos el costo de la mejor solución obtenida hasta ahora (que se utiliza como cota superior global: la solución óptima debe tener un coste menor o igual a esa). Esa variable se inicializa con el valor de la primera rama que explora.
Con esto seguimos hacia delante, y calculamos la distancia absoluta del siguiente nivel, si ésta es menor que la cota anterior, expande este nodo y lo añade a nuestro vector de solución.
En caso contrario, poda esa rama y se continúa con la siguiente.
Podemos controlar los niveles en los que hemos estado mediante un campo "bandera" que nos indica si ese nivel ha sido visitado ya.

3.1 Algunos ejemplos de ejecución con gnuplot.

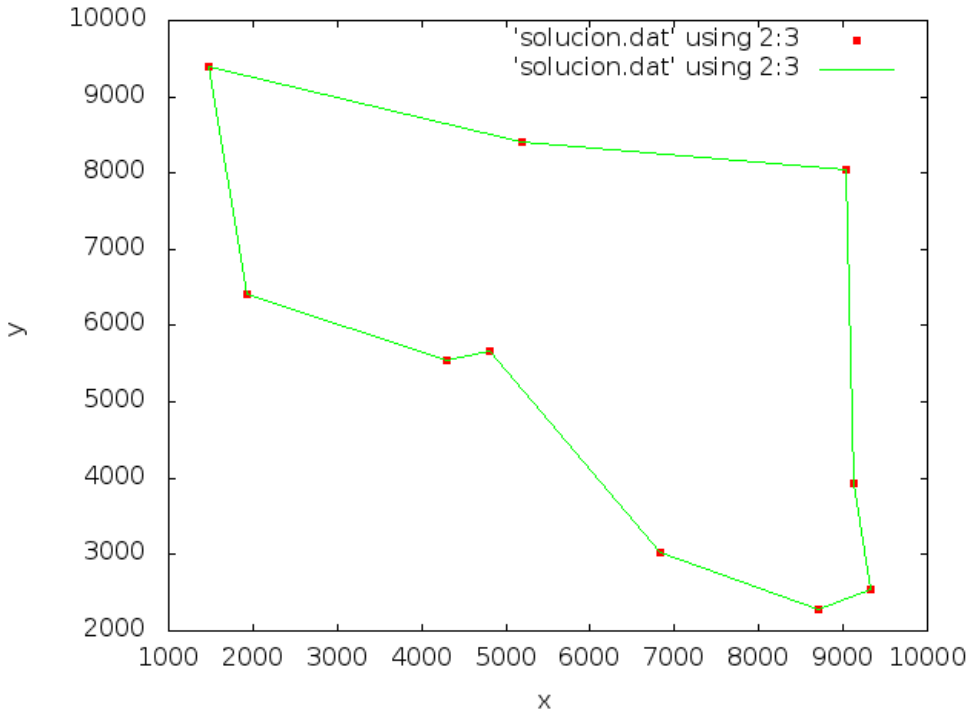
Primer ejemplo de solución con 15 ciudades:

Ciudades	Valor eje X	Valor eje Y
1	8671	5217
8	7871	8460
13	8299	9533
4	7738	9656
5	7007	8718
2	5562	9844
7	5389	9174
12	5007	6325
3	4333	7286
9	1195	7085
6	1408	6409
14	324	4767
0	2782	3443



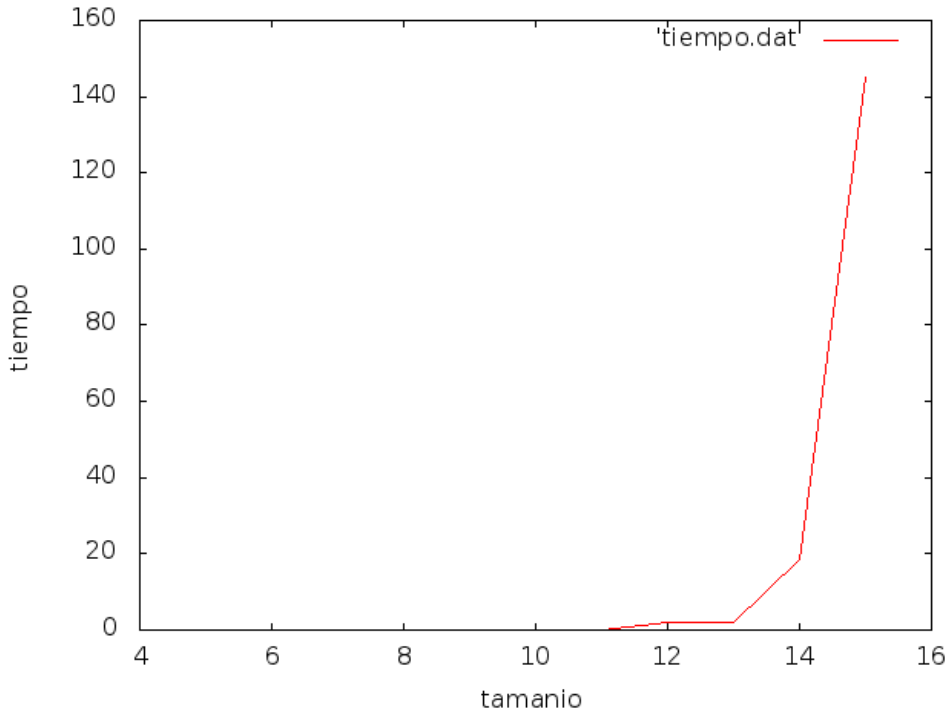
Segundo ejemplo con 10 ciudades:

Ciudades	Valor eje X	Valor eje Y
3	1480	9402
5	1936	6415
4	4302	5554
8	4813	5660
7	6831	3021
9	8702	2278
6	9320	2548
2	9124	3932
0	9039	8043



4. *Cálculo de la eficiencia empírica del algoritmo*

Tamaño	Tiempo
8	0.00281443
9	0.0159347
10	0.0306175
11	0.143857
12	1.93821
13	2.00506
14	18.651
15	144.896



Como podemos comprobar cuando aumenta el tamaño el tiempo va creciendo estrepitosamente, sobre todo al pasar el umbral del 14, el tiempo crece de forma exponencial y rápidamente.