ALGORITMIA

Grado en Ingeniería Informática del Software - Curso 2022-2023

GUIÓN DE LA PRÁCTICA 7

OBJETIVO:

Algoritmos Ramifica y Poda o Branch and Bound (RyP o ByB)

EL VIAJANTE DE COMERCIO

Seguimos con el mismo problema del viajante (cálculo del ciclo mínimo o de Hamilton) visto en la práctica 6 anterior, pero nos planteamos como meta rebajar **lo más posible** el tiempo de resolución o ejecución para un grafo de entrada de tamaño n nodos.

Para conseguir lo anterior, pretendemos calcular el ciclo de Hamilton por Ramifica y Poda (RyP), para ello se ha de diseñar e implementar un heurísticoRyP que permita ir desarrollando el árbol de estados del problema de forma que se mejore (lo más posible) el tiempo logrado por Backtracking en la práctica anterior (Backtracking supone desarrollar el árbol de estados por profundidad).

Como siempre, la calidad del heurístico será la que determinará la obtención de mejores o peores tiempos de ejecución; o en definitiva, alcanzar en mayor o menor grado la meta a la que se aludía en el párrafo anterior. En el directorio HeurísticosRyP hay unos documentos (de diferentes Universidades españolas) buscados en la Red, que inciden en conceptos que se pueden utilizar al respecto, sin perjuicio de que pueda usar cualquier otro heurísticoRyP que encuentre, o incluso realizar el diseño de uno "de cosecha propia".

Se pide:

*Implementar una clase **ViajanteRyP**, que calcule y escriba el ciclo hamiltoniano y su coste mínimo asociado (los datos de entrada estarán en un fichero como los vistos en la práctica anterior).

java paquete.ViajanteRyP nombreFicheroEntrada

Se recuerda lo obtenido en la práctica anterior para:

grafo04.txt:

NODO0 > NODO1 > NODO3 > NODO2 > NODO0

COSTE= 46

grafo07.txt:

NODO0 > NODO3 > NODO2 > NODO1 > NODO5 >NODO6 > NODO4 > NODO0

COSTE=141

grafo10.txt:

NODO0 > NODO5 > NODO1 > NODO2 > NODO4 > NODO8 > NODO6 > NODO3 > NODO7 > NODO9 > NODO0

COSTE= 262

Se pide:

*Implementar una clase **ViajanteTiemposRyP**, que vaya simulando el crecimiento del tamaño del problema (para valores n=5, 10, 15, 20, 25, 30, 35, 40, …) y mida tiempos para cada tamaño n.

Hay que rellenar una tabla así:

n	*tiempoBK*	*tiempoRyP*
5	……	……..
10	……	……..
15	……	……..

y así sucesivamente

Para que los tiempos obtenidos sean más fidedignos, para cada n se han de generar **5** grafos con los pesos o distancias de cada arista aleatorios en el rango [10..99]. Pues bien, para cada uno de esos 5 grafos aleatorios se ha de medir el tiempo tanto de BK como de RyP y al final el tiempo obtenido, que se reflejará en la tabla para cada política, será la media aritmética de los 5 tiempos obtenidos.

Es de esperar que los tiempos RyP mejoren a los tiempos BK (eso es lo que se busca), por lo que cuando los tiempos BK se hagan grandes para un determinado tamaño n (p.e. superiores al minuto), procede seguir midiendo tiempos solamente para RyP, hasta que lleguen tamaños en los que estos tiempos RyP se hagan también superiores al minuto.

Se entregará:

- *Los ficheros fuente de las clases, que se hayan programado, dentro del paquete o del proyecto Eclipse.*
- *Clases de prueba*
- *Un documento PDF con la tabla de tiempos cubierta y la explicación de la correspondencia con la complejidad teórica*

Se habilitará una tarea en el campus virtual para realizar la entrega. Esta práctica se trabajará durante una sesión y el ***plazo límite es un día antes*** *de la próxima sesión de prácticas de tu grupo.*