ALGORITMIA

Grado en Ingeniería Informática del Software - Curso 2018-2019

GUIÓN DE LA PRÁCTICA 7

OBJETIVO:

- **Algoritmos de Ramifica y Poda (RyP)**

Encontrar el emparejamiento estable óptimo

Seguimos con el problema enunciado en la práctica anterior (emparejamiento estable óptimo). Le hemos dado solución por backtracking, ya que al tratarse de un problema NP no tiene soluciones de tipo polinómico. Posteriormente, hemos cubierto la tabla de tiempos y hemos comprobado lo que ya se esperaba, que para tamaños del problema bastante pequeños el tiempo de ejecución se dispara y no acaba de ejecutarse.

Por otra parte, ciertos problemas resueltos por backtracking son susceptibles de ser resueltos por RyP, pudiendo mejorar significativamente los tiempos de ejecución. Muy probablemente este problema sea uno de ellos, siempre y cuando encontremos un heurístico de ramificación eficaz, que suponga ir desarrollando (en el orden que ese heurístico dictamine) los nodos del árbol de estados del problema, y así poder llegar a la solución pedida antes que yendo en profundidad (backtracking).

En definitiva, el reto es encontrar la forma "más inteligente" de ramificar el árbol de estados, que nos permita resolver en tiempos razonables, tamaños mayores del problema del emparejamiento estable óptimo.

SE PIDE: Diseñar un algoritmo (Clase EmparejamientoEO_RyP) capaz de mejorar lo más posible los tiempos de la clase EmparejamientoEO de la práctica anterior.

Caso de prueba:

Si ejecutamos: **java paquete.EmparejamientoEO_RyP 5 rechazo06.txt**

Evidentemente, ha de dar la misma solución, para cualquier caso de entrada, que en la práctica anterior. En este caso:

para umbralRechazo<=3: NO HAY EMPAREJAMIENTO POSIBLE

para umbralRechazo>=4: EMPAREJAMIENTO ÓPTIMO POSIBLE:
hombre[0] --- mujer[1]
hombre[1] --- mujer[3]
hombre[2] --- mujer[2]
hombre[3] --- mujer[5]
hombre[4] --- mujer[0]
hombre[5] --- mujer[4]
PUNTUACIÓN TOTAL MÍNIMA=17

Tiempos de ejecución

Calcular los tiempos de ejecución de la tabla siguiente para un umbralRechazo=5. Las matrices en este caso se generarán aleatoriamente (cada elemento rango 0..10) y para que el tiempo sea más fiable, haga (para cada n) tomas de tiempos sobre 10 matrices diferentes y lleve a la tabla la media aritmética de esos 10 valores temporales.

TABLA TIEMPOS RyP:

Caso de estudio (umbralRechazo=5; media de 10 casos aleatorios para cada n)

n	*tiempo*
20	…….
30	…….
40	…….
…	…….
100	…….
…	…….
200	…….
…	…….

Y así sucesivamente hasta que el tiempo se dispare

Heurístico de Ramificación

Explique claramente en qué consiste su algoritmo propuesto (**heurístico RyP**).

*Si utiliza **Eclipse**, se creará el proyecto **prac07_Ryp<UOpropio>** con todas las clases necesarias.*
*Si utiliza **JDK**, cree los paquetes necesarios para esta práctica.*
*Las clases necesarias se crearán dentro del paquete **algdnipropio.p7***

Se entregará, por separado:

- *Los ficheros fuente de las clases, que se hayan programado, dentro del paquete o del proyecto Eclipse.*
- *Un documento con una pequeña explicación de los algoritmos utilizados, su complejidad y la tabla de tiempos de ejecución.*

Se habilitará una tarea en el campus virtual para realizar la entrega. La fecha tope de entrega finalizará el domingo 28 de abril a las 23:55 (último día antes de empezar la semana del control que finaliza la evaluación continua de prácticas).