Práctica de la asignatura

El problema del viajante de comercio

Javier Herrer Torres (NIP: 776609)
Javier Fuster Trallero (NIP: 626901)

Algoritmia básica
Grado en Ingeniería Informática

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza
Curso 2020/2021

1. Metodología

Fuerza bruta

El algoritmo de *fuerza bruta* para resolver el problema consiste en intentar todas las posibilidades, es decir, calcular las longitudes de todos los recorridos posibles, y seleccionar la de longitud mínima. Para calcular todas las permutaciones posibles, se ha empleado el Algoritmo de Heap.

Algoritmo voraz

Se ha empleado el algoritmo de Kruscal, basado en la propiedad de los árboles de recubrimiento mínimo. Partiendo del árbol vacío, se selecciona en cada paso la arista de menor coste que no provoque ciclo sin requerir ninguna otra condición sobre sus extremos. Para comprobar esta restricción, se emplea una matriz de adyacencia de booleanos.

Programación dinámica

Se trata de calcular $g(1, V\backslash\{1\})$ con la función parametrizada $g(i, S) = min_{j \in S}\{L_{ij} + g(j, S \setminus \{j\})\}$.

Se emplea la matriz `gtab` para almacenar los valores de g calculados para cada S. Esta estructura se ha implementado con un `Map<Vertice, Map<Set<Vertice>, Tupla>>` , donde `Tupla` almacena tanto el valor de g como el de J (para poder construir el circuito óptimo).

Ramificación y poda

Se ha definido la función de estimación $\hat{c}(x)$ siguiendo los pasos de las tranparencias del tema y el esquema algorítmico genérico. Además, se ha empleado un montículo con prioridades (`PriorityQueue<Nodo>`) para almacenar la frontera.

2. Resultados

Los siguientes resultados (cuadro 1 y la figura 1) se han obtenido con matrices generadas aleatoriamente con un cierto número de vértices. Estas matrices representan grafos dirigidos.

Además, en el cuadro 2 se han incluido los resultados obtenidos para los ficheros proporcionados en los que ha sido posible ejecutar los algoritmos.

3. Conclusiones

El **algoritmo de fuerza** para resolver el problema consiste en intentar todas las posibilidades. Obviamente, el coste crece exponencialmente con el número de puntos a visitar.

Vértices	FB	AV	PD	RyP
7	321	15	104	17
8	1292	14	180	25
9	6346	13	368	42
10	96438	15	516	98
11		14	1081	62
12		17	1800	61
13		19	3641	82
14		17	10194	53
15		18	35922	158
16		20	120832	164
17		18	447403	72
18		19	1722253	145
19		19	6558672	82
20		20		131
21		19		340
22		23		241
23		22		699
24		21		6211
25		22		8100
26		23		11213
27		22		16770
28		28		28384
29		27		18160
30		25		
50		71		
100		147		
500		1439		
1000		3445		

Cuadro 1: Tiempos de ejecución en ms.

El **algoritmo voraz** tiene el mejor coste temporal, pero debe destacarse que *tiene la posibilidad, pero no la certeza, de encontrar la solución óptima*. De hecho, en la mayoría de casos no la encuentra aunque el coste pueda aproximarse al óptimo.

Programación dinámica mejora a fuerza bruta pasando de un tiempo $\Omega(n!)$ a $\Theta(n^2 2^n)$, es decir, *pasando de un coste temporal factorial a exponencial*. Pero, a cambio, el coste en espacio es de $\Omega(n2^n)$ resultante de almacenar la matriz `gtab`. A partir de 19 vértices, el tamaño de esta matriz crece generando un `java.lang.OutOfMemoryError: Java heap space`.

En el algoritmo de **Ramificación y poda**, el coste del caso peor de la solución; pese a ser teóricamente el mismo que en programación dinámica, $\Theta(n^2 2^n)$; se observa una clara diferencia en la práctica. Pudiendo resolver problemas de hasta 10 vértices más. A partir de 29 vértices, el tamaño de la frontera crece

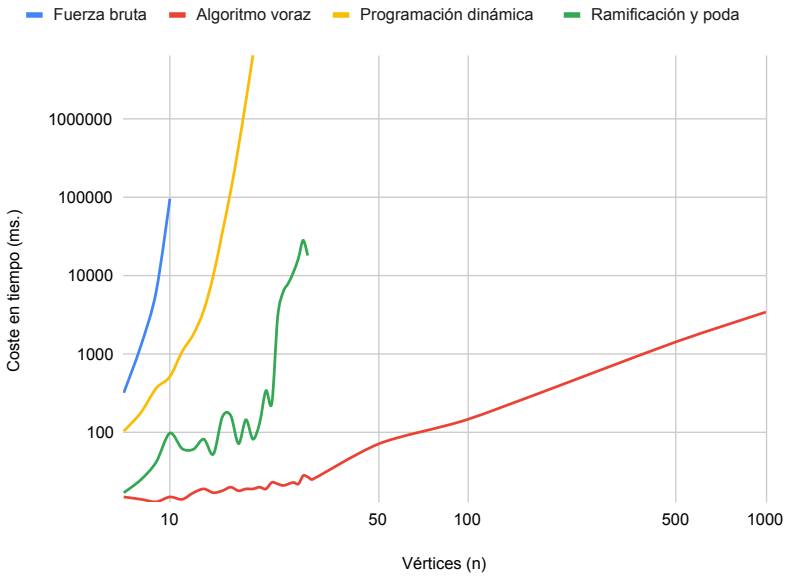

Figura 1: Gráfico de costes temporales

Fichero	AV	PD	RyD
a4	13	12	13
a11	14	1162	72
a15	19	40531	179
a16	21	136522	151

Cuadro 2: Resultados para los ficheros proporcionados

generando un `java.lang.OutOfMemoryError: Java heap space`. Además, se ha apreciado que en este algoritmo existe una gran variabilidad en el coste temporal para distintas matrices con mismo número de vértices, esto se debe a la función de poda.