Miguel Méndez Murias
UO287687 – Algoritmia

PRÁCTICA 7

El viajante de comercio

En esta práctica tuvimos que desarrollar un algoritmo de resolución del problema del viajante más eficiente que el hecho en la práctica anterior. Más concretamente, en la práctica anterior se resolvía el programa mediante backtracking, lo cual es bastante ineficiente, mientras que el objetivo de esta práctica era encontrar e implementar un heurístico que nos permitiese resolver el problema por Ramifica y Poda de la forma más óptima posible.

Heurístico

La calidad del heurístico es el que determina la eficiencia del algoritmo en la resolución del problema, por lo que es muy importe elegir uno adecuado. Como el problema del viajante del comercio es muy conocido, existen numerosos heurísticos pensados por estudiosos en la materia que son muy eficientes, por lo que una opción era implementar uno de ellos. Sin embargo, finalmente he decidido realizar mi propio heurístico por dos razones principales: primero porque me obliga a encontrar una posible solución a un problema complejo desde cero, y segundo porque es mucho más desafiante pensar y desarrollar un heurístico propio que no sabes si va a funcionar (por lo que se puede necesitar desarrollar varios diferentes) que implementar uno que ya sabes que es muy eficiente.

El heurístico utilizado se basaba primero en una ordenación de los nodos según el coste de acceso desde el nodo origen (y sucesivamente desde los siguientes según se fuese accediendo), lo que permite que se acceda siempre primero a los nodos menos costosos para poder encontrar la mejor solución posible. Aunque este puede ser un buen comienzo, no garantiza que sea lo mejor para resolver el problema de forma eficiente, aunque si que es sencillo de implementar.

Además de esto, para podar las ramas que no podían llevar a ninguna solución se desarrollo otro heurístico que selecciona el coste mínimo de acceso desde cada nodo que esté disponible (es decir, que no haya sido visitado). Sumando todos estos costes mínimos junto con el coste de la solución parcial actual, podemos saber si la rama tiene posibilidades de ser una solución o no (esto se sabe comparando el coste mínimo de finalización de la rama con el coste de la solución que se tiene hasta el momento para el problema). En caso de que la rama no tenga posibilidades se descarta, lo que reduce en gran medida la complejidad del problema. Sin embargo, esto no siempre conlleva la misma cantidad de mejora, ya que es muy dependiente de los costes de los caminos y el calculo de este coste mínimo de finalización de la rama actual también puede aumentar la complejidad del problema, ya que tiene una complejidad cuadrática. A pesar de esto, en la gran mayoría de los casos la mejora si que es muy notable, ya que es mucho mejor desarrollar una operación $O(n^2)$ adicional que profundizar en una rama que no puede ser solución.

Finalmente, para la solución del problema se mezclaron los dos heurísticos explicados anteriormente (el de ordenación según mínimo coste y la poda de ramas que superan el coste mínimo de finalización), lo que permitió mejorar mucho el tiempo que se tarda en resolver el problema respecto a la solución con backtracking, tal y como se muestra posteriormente.

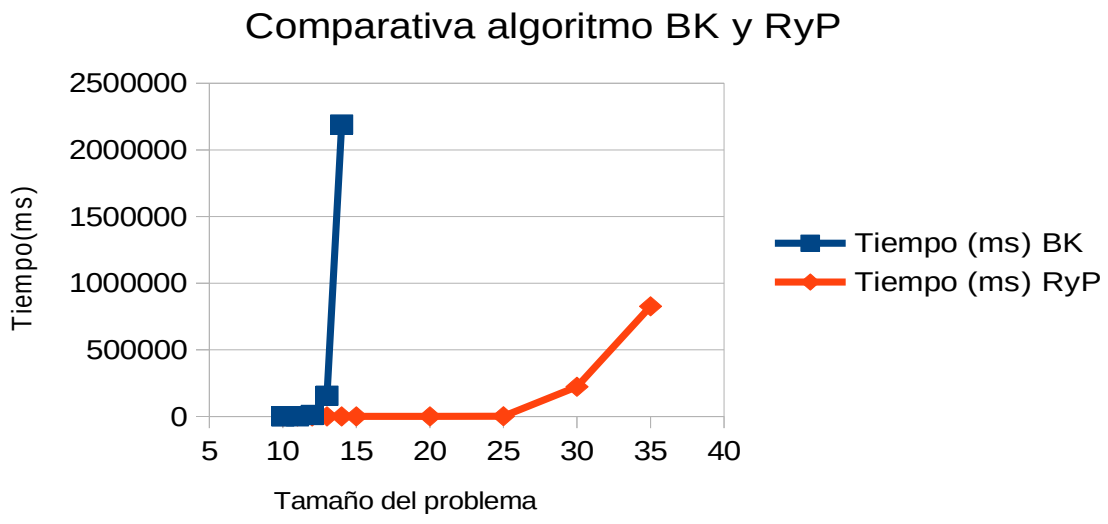
Miguel Méndez Murias
UO287687 – Algoritmia

Tiempos

Para comprobar la mejora en la eficiencia del nuevo algoritmo desarrollado, realizamos mediciones del algoritmo usando backtracking y lo comparamos con este que usa Ramifica y Poda. Para que estas mediciones fueron fiables, para cada tamaño del problema, creamos cinco grafos generados aleatoriamente, de forma que las mediciones se realizasen sobre el mismo problema. Teniendo en cuanta esto, el tiempo que tarda el algoritmo en resolver un problema de ese tamaño sería la media de lo que tarda en resolver los cinco problemas.
Estos fueron los resultados:

Tamaño	Tiempo (ms) BK	Tiempo (ms) RyP
10	105	1
11	1016	2
12	11124	2
13	155335	3
14	2190397	4
15	-	4
20	-	70
25	-	1279
30	-	222960
35	-	826731

Como el algoritmo backtracking necesita mucho más tiempo, realizamos las mediciones para los tamaños del 10 al 15 de uno en uno, para posteriormente ir incrementando el tamaño de cinco en cinco para medir los tiempos del algoritmo que usa el Ramifica y Poda.
El gráfico correspondiente a la tabla de tiempos anterior es el siguiente:



Como se puede observar en la gráfica, el nuevo algoritmo desarrollado en esta práctica es mucho más eficiente que el que usa backtracking (realizado en la práctica anterior). De hecho, el nuevo algoritmo necesita menos tiempo para solucionar un problema de tamaño 35 que el algoritmo que utiliza backtracking en resolver uno de tamaño 14, lo que es una mejora muy significativa.
A pesar de que la mejora es muy notable, esta depende del heurístico utilizado, por lo que en función de este podría haber algún otro algoritmo que utilizase Ramifica y Poda y que fuese mucho más eficiente que el desarrollado en esta práctica.