AMC_MemoriaP2

Problema del viajante

Partimos de uno de los problemas de optimización combinatoria más conocidos: encontrar el circuito de menor coste, dadas una serie de ciudades, que parta de una ciudad y recorra el resto una única vez, volviendo a la ciudad de origen.

Si no le damos importancia a la ciudad de origen, tendríamos $(n-1)!$ rutas posibles. En cambio, si tenemos en cuenta la ciudad de origen, obtenemos $n!$ rutas posibles.

Por la dificultad de encontrar soluciones a problemas NP-completos, empleamos técnicas que den con soluciones aceptables en un tiempo computacionalmente prudente.

> **BÚSQUEDA VORAZ EXHAUSTIVA**
>
> Según Wikipedia y los apuntes de la asignatura[1] podemos entender que el algoritmo voraz es una estrategia de búsqueda que se centra en buscar la solución más óptima en cada paso local/iteración para así intentar alcanzar la solución general óptima, que no es más que la solución que más se aproxima a la función objetivo.
> Podríamos decir que su principal desventaja es que, una vez toma una decisión, no puede retroceder.

Aproximaciones

Unidireccional exhaustiva

En esta aproximación, partiendo de una ciudad aleatoria, nos desplazamos a su ciudad más cercana, y una vez en ella, volvemos a desplazarnos a su ciudad más próxima[2] que no haya sido visitada. Esto quiere decir que sólo comprobamos las distancias en la cola (tail) de la cola/array. Hay que tener en cuenta que con esta estrategia de búsqueda no podemos retroceder, es decir, una vez hemos establecido como ciudad actual la que tenía menor distancia con la anterior, sólo podemos comprobar distancias con esta ciudad y no con las anteriores.

Bidireccional exhaustiva

En esta aproximación, partiendo de una ciudad aleatoria, nos desplazamos a su ciudad más cercana, y una vez en ella, volvemos a desplazarnos a su ciudad más próxima[2-1] que no haya sido visitada. Comprobamos las distancias en la cola (tail) y cabeza de la cola/array. Hay que tener en cuenta que con esta estrategia de búsqueda no podemos retroceder, es decir, una vez hemos establecido como ciudad actual la que tenía menor distancia con la anterior, sólo podemos comprobar distancias con esta ciudad y no con las anteriores.

Unidireccional con Poda

En esta aproximación, partimos ordenando el array por la coordenada x del punto. Partiendo de una ciudad aleatoria, empezamos a recorrer el array buscando la ciudad más cercana, y una vez en ella, volvemos a desplazarnos a su ciudad más próxima[2-2] que no haya sido visitada. Esto quiere decir que sólo comprobamos las distancias en la cola (tail) de la cola/array. Hay que tener en cuenta que con esta estrategia de búsqueda no podemos retroceder, es decir, una vez hemos establecido como ciudad actual la que tenía menor distancia con la anterior, sólo podemos comprobar distancias con esta ciudad y no con las anteriores.
La diferencia con respecto a la exhaustiva es que, como los puntos están ordenados por coordenada X, si la distancia entre las dos coordenadas X de los dos puntos a comparar es mayor (o igual) que la distancia mínima actual, quiere decir que, por lo menos, la distancia mínima de esos dos puntos y los siguientes va a ser igual o mayor a la mínima actual. Por eso mismo, es posible realizar la poda y no hacer comparaciones innecesarias.

Bidireccional con Poda

En esta aproximación, partimos ordenando el array por la coordenada x del punto. Partiendo de una ciudad aleatoria, nos desplazamos a su ciudad más cercana, y una vez en ella, volvemos a desplazarnos a su ciudad más próxima[2-3] que no haya sido visitada. Comprobamos las distancias en la cola (tail) y cabeza de la cola/array. Hay que tener en cuenta que con esta estrategia de búsqueda no podemos retroceder, es decir, una vez hemos establecido como ciudad actual la que tenía menor distancia con la anterior, sólo podemos comprobar distancias con esta ciudad y no con las anteriores.
La diferencia con respecto a la exhaustiva es que, como los puntos están ordenados por coordenada X, si la distancia entre las dos coordenadas X de los dos puntos a comparar es mayor (o igual) que la distancia mínima actual, quiere decir que, por lo menos, la distancia mínima de esos dos puntos y los siguientes va a ser igual o mayor a la mínima actual. Por eso mismo, es posible realizar la poda y no hacer comparaciones innecesarias.

Código

Vamos a hacer uso de la librería externa JMathPlot
Gestionamos su dependencia con Maven por lo que no deberíamos hacer nada al respecto para ejecutar el programa en otro dispositivo

datosAMC_Prac2.xlsx

---

1. Diapositiva [48-52] del Tema 3 ↩
2. De manera exhaustiva ↩ ↩ ↩ ↩