Resolviendo el problema del TSP con Recocido Simulado

Víctor Zamora Gutiérrez

El recocido simulado es una heurística de optimización combinatoria que sirve para resolver problemas NP duros. En esta ocasión, utilizamos dicha heurística para resolver un caso modificado del Traveling Salesman Problem (TSP): dado un conjunto de ciudades, encontrar el camino de menor peso que pase por todas las ciudades exactamente una vez.

1. Para correr el programa

Para correr el programa, se necesita una distribución de Linux con java 8, ant y sqlite3. Para instalar dichos programas desde ArchLinux, ejecutar:

```
$sudo pacman -S jre8-openjdk
$sudo pacman -S jdk8-openjdk
$sudo pacman -S apache-ant
$sudo pacman -S sqlite
```

Teniendo esto, desde la carpeta raíz del proyecto (la que tiene el build.xml) ejecutar:

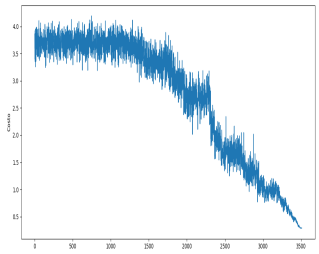
```
$ant tsp.jar
$java -jar tsp.jar <semilla> <ciudades>
```

Donde semilla es la semilla con la que se quiere correr el recocido y ciudades es un archivo que contiene una lista de ids de ciudades, separadas por una coma y un espacio. Por ejemplo, ciudades puede tener la siguiente línea:

1, 5, 9, 12, 16, 22, 23, 29, 30, 31, 39, 48, 52, 56, 58, 62, 65, 66, 70, 75, 80, 84, 86, 90, 92, 94, 95, 101, 107, 117, 119, 122, 133, 135, 143, 144, 146, 147, 150, 158, 159, 160, 166, 167, 176, 178, 179, 185, 186, 188, 190, 191, 194, 198, 200, 203, 207, 209, 213, 215, 216, 220, 221, 224, 227, 232, 233, 235, 238, 241, 244, 248, 250, 254, 264, 266, 274, 276

2. Ejecución en una gráfica

Mostraremos la ejecución del algoritmo de Warshall en la siguiente gráfica:



Paso 1 Primero tomamos a 1 como pivote. Recordemos que lo que queremos es relacionar dos elementos que estén indirectamente relacionados por medio del 1. En este caso, los únicos nodos que cumplen esto son 2 y 4, como se muestra a continuación:
Sin embargo, 2 y 4 ya están relacionado por lo que no hacemos nada.

Paso 2 Tomamos a 2 como pivote. En este caso tenemos que 3 está relacionado con 2, por lo que debemos relacionar a 3 con todos los elementos con los que 2 está relacionado (en este caso son 1 y 4):

Paso 3 Tomamos a 3 como pivote. Como 4 se relaciona con 3, debemos relacionar a 4 con todos los vecinos de 3 (en este caso, con todos los nodos excepto 3).

Paso 4 Por último, tomamos a 4 como pivote. Como 1, 2 y 3 se relacionan con 4, los tenemos que relacionar con los vecinos de 4 (en este caso, todos los nodos). Y tenemos la gráfica de la cerradura transitiva.

3. Ejecición en una matriz

Las relaciones también pueden representarse como matrices (y de hecho, el algoritmo originalmente se hace sobre una matriz). A continuación, veremos la ejecución del algoritmo en la siguiente matriz de relación.

Paso 1 Tomamos al primer elemento como pivote. Primero nos fijamos en la fila 1. Esta sólo tiene 1 en la columna 4 y 0 en las demás. Luego nos fijamos en las demás filas. A las que tengan un 1 en la columna 1, les agregamos todos los 1's de la fila 1(en este caso, sólo agregamos el 1 de la columna 4). Después de la primera iteración, la matriz queda así: Nótese que aunque la fila 3 tiene un 1 en la columna 1, no se le agregan 1's porque ya tiene el 1 de la columna 4.

Paso 2 Ahora tomamos al segundo elemento como pivote. La segunda fila tiene 1's en las columnas 1, 3 y 4. Nos fijamos en las demás filas y de nuevo, a las que tengan un 1 en la columna 2, les agregamos 1's en las columnas 1, 3 y 4. La matriz queda así: Como ninguna fila tiene un 1 en la columna 2, no hubo cambios.

Paso 3 Tomamos al tercer elemento como pivote. La fila 3 tiene 1's en las columnas 1 y 4. Ahora nos fijamos en las filas que tengan 1 en la columna 3 y les agregamos los 1's de las columnas 1 y 4.

Paso 4 Por último, tomamos al cuarto elemento como pivote. La cuarta fila tiene 1's en las columnas 1, 3 y 4. Nos fijamos en todas las filas que tengan un 1 en la columna 4 y les agregamos estos elementos: Como ya no quedan elementos para usar como pivotes, ya tenemos la cerradura transitiva: