Práctica Greedy

Práctica 3 de Algorítmica

Curso 2017 - 2018

ugr Universidad de Granada

DECSAI

Reparaciones

Un electricista necesita hacer n reparaciones urgentes, y sabe de antemano el tiempo que le va a llevar cada una de ellas: en la tarea i-ésima tardará t_i minutos. Como en su empresa le pagan dependiendo de la satisfacción del cliente, necesita decidir el orden en el que atenderá los avisos para minimizar el tiempo medio de atención de los clientes (desde el inicio hasta que su reparación es efectuada).

- Diseñar un algoritmo greedy para resolver esta tarea.
- Demostrar que el algoritmo obtiene la solución óptima.
- Modificar el algoritmo anterior para el caso en que se disponga de más de un electricista.

El problema del viajante de comercio

El problema del viajante de comercio (TSP, por Traveling Salesman Problem) se define como: dado un conjunto de ciudades y una matriz con las distancias entre todas ellas, un viajante debe recorrer todas las ciudades exactamente una vez, regresando al punto de partida, de forma tal que la distancia recorrida sea mínima.
Mas formalmente, dado un grafo G, conexo y ponderado, se trata de hallar el ciclo hamiltoniano de mínimo peso de ese grafo.
Una solución (un tour) para TSP es una permutación del conjunto de ciudades que indica el orden en que se deben recorrer. Para el cálculo de la longitud del tour no debemos olvidar sumar la distancia que existe entre la última ciudad y la primera (hay que cerrar el ciclo).

Nos centraremos en una serie de algoritmos aproximados de tipo greedy y evaluaremos su rendimiento en un conjunto de instancias del TSP. Para el diseño de estos algoritmos, podemos hablar de dos enfoques subyacentes: a) estrategias basadas en alguna noción de cercanía, y b) estrategias de inserción.

Para el primer caso, tenemos la heurística del *vecino más cercano*, cuyo funcionamiento es extremadamente simple: dada una ciudad inicial v_0, se agrega como ciudad siguiente aquella v_i (no incluída en el tour) que se encuentre más cercana a v_0. El procedimiento se repite hasta que todas las ciudades se hayan incluido.

Para el caso de las estrategias de inserción, la idea es comenzar con un recorrido parcial, que incluya algunas de las ciudades, y luego extender este recorrido insertando las ciudades restantes mediante algún criterio. Para poder implementar este tipo de estrategia, deben definirse tres elementos:

1. Cómo se construye el recorrido inicial parcial
2. Cuál es el nodo siguiente a insertar
3. Dónde se inserta

El recorrido inicial se puede construir a partir de las 3 ciudades que formen el triángulo más grande: por ejemplo, eligiendo la ciudad que está más a la izquierda, la que está más a la derecha, y la que está más arriba.

Una vez seleccionada la ciudad a insertar, ésta será ubicada en el punto que provoque el mínimo incremento en la longitud del tour. Es decir, hay que comprobar, para cada posible posición, el incremento que se produce.
Para decidir cuál es la ciudad a insertar, podemos aplicar el siguiente procedimiento, denominado *inserción más económica*: entre todas las ciudades no visitadas, elegir aquella que provoque el mínimo incremento de la longitud. En otras palabras, para cada ciudad debemos insertarla en cada posición posible y quedarnos con la posición que obtiene el mínimo. Seleccionaremos aquella ciudad que nos de el mínimo entre todos los mínimos calculados.

Tareas a realizar

- Construir un programa que proporcione soluciones para el problema del viajante de comercio empleando las dos heurísticas descritas anteriormente, así como otra adicional propuesta por el propio equipo.
  Aunque el programa debe ser capaz de devolver el recorrido obtenido, en principio su salida principal es únicamente la longitud de dicho recorrido.
- Se debe realizar un estudio comparativo de las tres estrategias empleando un conjunto de datos de prueba.

Datos de prueba

Están disponibles en la plataforma de docencia de la asignatura, y han sido obtenidos y adaptados de la librería TSPLIB. El formato de los ficheros es el siguiente:

```
DIMENSION: 51
1 565.0 575.0
2 25.0 185.0
3 345.0 750.0
..............
50 595.0 360.0
51 1340.0 725.0
```

La primera fila indica la cantidad de ciudades en el fichero. A continuación, aparece una fila por cada ciudad conteniendo tres valores: el número de ciudad, la coordenada X y la coordenada Y.

En cada ejecución del algoritmo, deberá calcular la matriz de distancias entre todas las ciudades teniendo en cuenta lo siguiente: sean (x_i, y_i), (x_j, y_j) las coordenadas en el plano de las ciudades c_i, c_j . En primer lugar se calcula la distancia euclídea d entre ambos puntos como

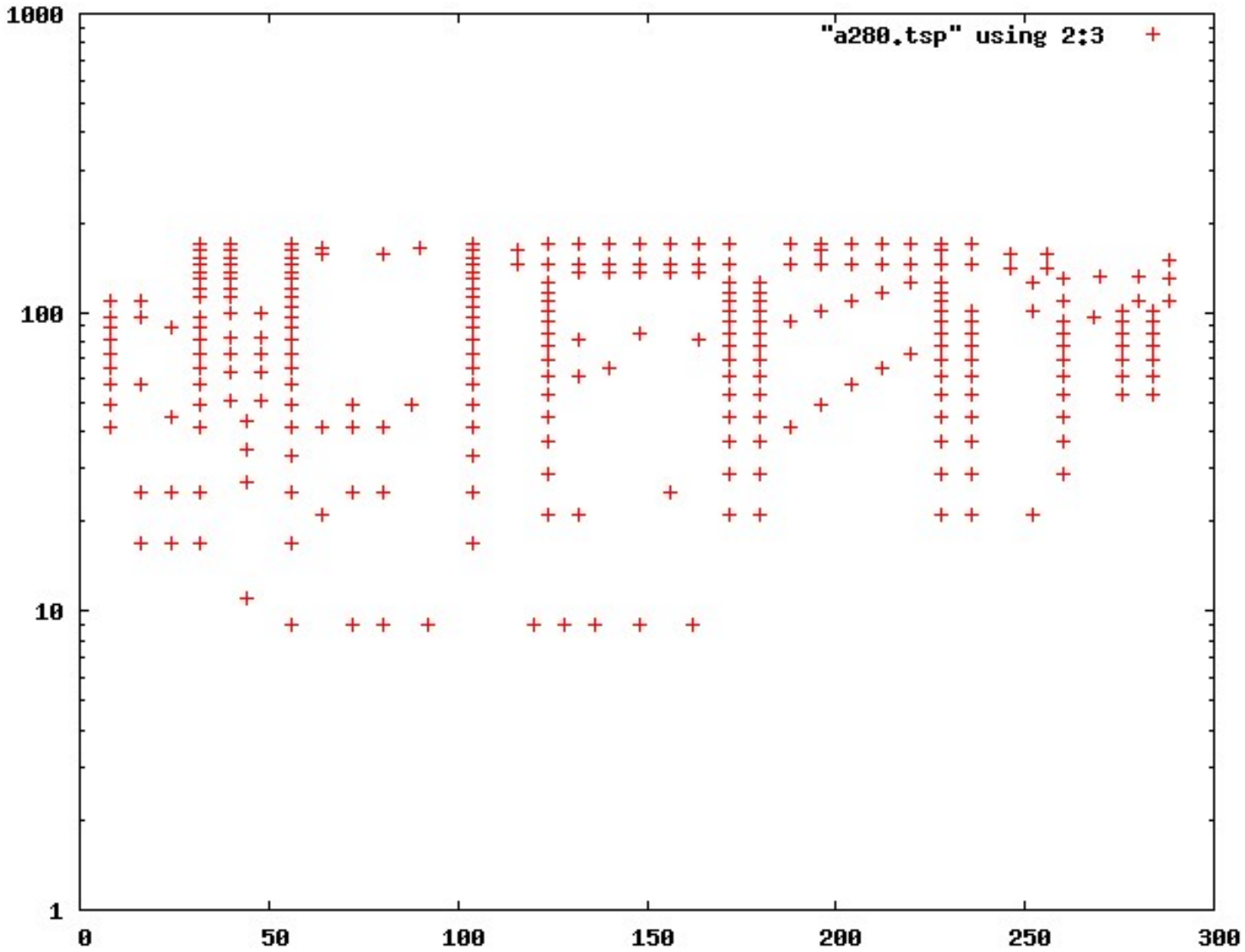
$$d = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$

Después la distancia d debe redondearse al entero más próximo.

Visualización

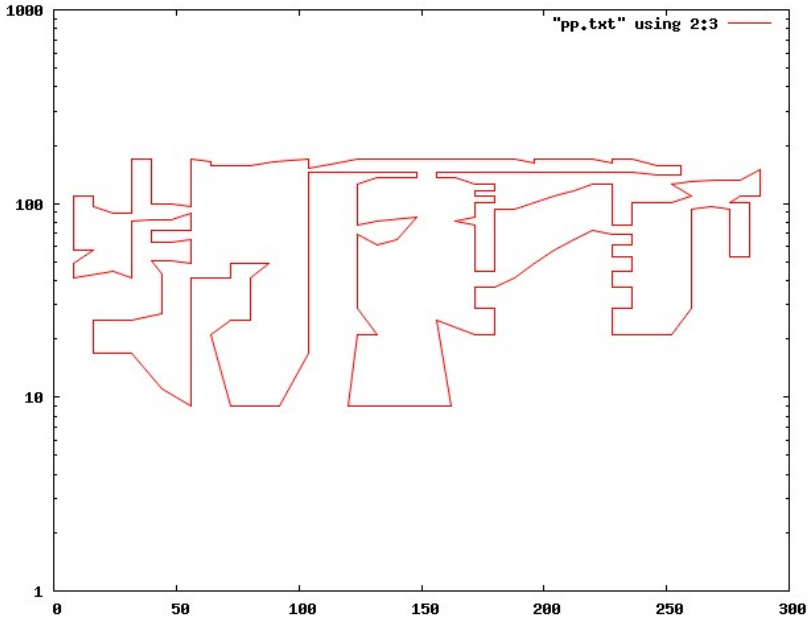
Si se desean visualizar los diferentes problemas, se puede utilizar gnuplot. Simplemente, si `fichero.tsp` es el fichero que contiene las coordenadas de las ciudades (en el formato anterior), basta con utilizar el comando siguiente (para indicarle a gnuplot que utilice los datos de la columna 2 del fichero como abscisas y los datos de la columna 3 como ordenadas en la representación gráfica):

```
gnuplot> plot "fichero.tsp" using 2:3 with points
```


"a280.tsp" using 2:3
1000
100
10
1
0
50
100
150
200
250
300

las diferentes ciudades (filas del fichero) se reordenan de acuerdo al orden de un tour concreto (por ejemplo el generado por nuestro algoritmo, o el óptimo) entonces es posible hacer también con gnuplot una representación gráfica del tour. Simplemente, si el fichero de coordenadas reordenado es `ficheroreord.tsp`, basta con ejecutar el comando siguiente (al usar la opción `with lines` se unen mediante una línea recta los puntos consecutivos, que al estar en el orden del tour, generan una representación visual del recorrido, a falta de cerrar el ciclo)):

```
gnuplot> plot "ficheroreord.tsp" using 2:3 with lines
```


"pp.txt" using 2:3
1000
100
10
1
0
50
100
150
200
250
300