Universidad Politécnica de Cartagena

PLANIFICACIÓN Y GESTIÓN DE REDES

Grado en Ingeniería Telemática
Curso 2015-2016

Caso de estudio. Algoritmos heurísticos para el problema de ubicación de nodos y enlaces

Autor:

Pablo Pavón Mariño

Antecedentes

En este caso de estudio los alumnos deberán desarrollar un algoritmo para resolver una versión del problema de localización de nodos que se describe a continuación.

Sea $i = 1, \ldots, N$ un conjunto de ubicaciones. En cada ubicación i se encuentra un nodo de acceso, y un número $z_i \in \{0, 1\}$ de nodos troncales. El coste c_j de ubicar un nodo troncal en una ubicación j se denota como c, y se asume constante (no dependiente de la ubicación donde se vaya a situar). Cada nodo de acceso i se debe conectar a un y sólo un nodo troncal, utilizando un *enlace de acceso*. El coste de un enlace de acceso del nodo de acceso en i con un nodo troncal en la ubicación j, lo denotamos como $c_{ij} \geq 0$. El número máximo de nodos de acceso que se pueden conectar a un nodo troncal es M. Finalmente, todos los nodos troncales están conectados entre sí (todos con todos). Esto implica que añadir un nodo troncal a la red, trae consigo la necesidad de conectarlo a todos los nodos troncales ya existentes. El coste de un enlace troncal originado en la ubicación i y terminando en la ubicación j se denota como c'_{ij}.

El objetivo del problema es encontrar (i) cuántos nodos troncales se adquieren y en qué ubicaciones se localizan (esto determina implícitamente los enlaces troncal-troncal), y (ii) cómo se conectan los nodos de acceso a los nodos troncales, tal que se minimice el coste total (sumando el coste de los nodos troncales, los enlaces acceso-troncal y troncal-troncal).

Objetivos

El objetivo de la práctica es realizar un algoritmo para ejecutar en `net2plan`, que resuelva instancias del problema de localización de nodos descrito anteriormente. Se hacen las siguientes consideraciones:

- El algoritmo debe recibir como entrada un diseño consistente en un conjunto de nodos. Cada uno de los nodos es una ubicación donde existe ya un nodo de acceso. Cada una de estas ubicaciones, podrá alojar o no un nodo troncal.
- El algoritmo deberá aceptar los siguientes parámetros de entrada:
  - `maxExecTimeSecs`: Tiempo máximo (en segundos) de ejecución del algoritmo. El alumno deberá implementar internamente el mecanismo que haga que la ejecución del algoritmo se detenga tras, a lo sumo, `maxExecTimeSecs` segundos[1]. Al detenerse, el algoritmo deberá devolver la mejor solución encontrada hasta el momento. El valor por defecto del parámetro será `maxExecTimeSecs = 60`.
  - `maxNumAccessNodesPerCoreNode`: El número máximo de nodos de acceso que pueden estar conectados con un nodo troncal (M en la sección anterior. El valor por defecto será `maxNumAccessNodesPerCoreNode = 5`. Nótese que dentro de este número se incluye el nodo de acceso que estaría en la misma ubicación que el nodo troncal.
  - `coreNodeCost`: Coste monetario de un nodo troncal, independientemente de su ubicación (c en la sección anterior). El valor por defecto será `coreNodeCost = 1000`.
  - `linkCostPerKm`: Coste monetario de cada km de los enlaces desde los nodos de acceso a los troncales, y por cada enlace entre dos nodos troncales. Es decir:
    - El coste de un enlace entre un nodo de acceso en i y un nodo troncal en j, que están separados d km es de: `linkCostPerKm` $\times\, d$ (ya se supone que el enlace es bidireccional, no hay que imputar el coste dos veces).

[1] Para medir el tiempo de ejecución del algoritmo, debe utilizar el método `System.nanoTime`

    * El coste de un enlace entre un nodo de troncal en i y un nodo troncal en j, que están separados d km es de: `linkCostPerKm` $\times d$ (ya se supone que el enlace es bidireccional, no hay que imputar el coste dos veces).

  El valor por defecto será `linkCostPerKm = 1`. La distancia en km entre dos ubicaciones se asume que es la proporcionada por el método `getNodePairPhysicalDistance` de la clase `NetPlan`.

- `linkCapacity`: Capacidad a fijar para todos los enlaces acceso-troncal que se planifiquen. El valor por defecto será `linkCapacity = 1`.

Una vez calculadas las ubicaciones de los nodos troncales (que determina los enlaces troncal-troncal) y los enlaces acceso-troncal, el algoritmo debe:

- Borrar todos los posibles enlaces existentes en el diseño anterior, por ejemplo llamando al método `removeAllLinks` de la clase `NetPlan`.
- Añadir un enlace, desde la ubicación del nodo de acceso a la ubicación del nodo troncal al que está asociado (*no se debe añadir un enlace en sentido contrario*), de capacidad igual a `linkCapacity`. Si un nodo de acceso se conecta con un nodo troncal en su misma ubicación, no se debe añadir un enlace al diseño `netPlan`. Los enlaces troncal-troncal TAMPOCO SE DEBEN AÑADIR AL DISEÑO, aunque su coste se habrá tenido en cuenta al tomar las decisiones de ubicación.

Nótese que se asumirá que en aquellas ubicaciones donde termine al menos un enlace acceso-troncal, es porque se ha ubicado un nodo troncal. Además, si de una ubicación no sale ningún enlace, se asume que en esa ubicación hay un nodo troncal, y que el nodo de acceso en esa ubicación se conecta con él.

El diseño obtenido deberá tener el mínimo coste total, y cumplir las restricciones del problema.

Entrega

Esta práctica se realizará en parejas. La práctica se implementará a través de un fichero en Java, de nombre `TCA_NodeAndLinkLocation_XXXX_YYYY.java`, donde XXXX será el primer y segundo apellido del primer miembro del grupo de prácticas, e YYYY primer y segundo apellido del segundo alumno del grupo.

Los alumnos deberán enviar por correo electrónico a `pablo.pavon@upct.es` este fichero, con fecha tope el Jueves 26 de Mayo de 2016 a las 23:59 horas. Se tomará como fecha de recepción de este trabajo la fecha de llegada en el servidor de correo. Cada hora de retraso en la entrega, se penalizará con un punto en la nota de este trabajo.

Los alumnos no tienen que entregar ningún material extra más allá de este fichero `.java`. No se admitirá repartir el código en varios ficheros `.java`. Si el alumno necesita definir varias clases Java, deberán estar todas las clases integradas dentro de un único fichero.

Evaluación

La evaluación de la práctica se hará de la siguiente manera:

1. 50%: El coste medio obtenido en una serie de instancias de red, que estarán disponibles en Aula Virtual, comparada en un ranking entre las soluciones de los grupos de prácticas. La nota

máxima en este apartado la llevará el grupo con la mejor solución, y la nota del resto de grupos irá decreciendo según criterio del profesor.

2. 10%: Claridad y limpieza de código. Se valorará un código más estructurado, limpio y con una cantidad adecuada de comentarios. Las primeras líneas de código deberán incluir como comentarios, los nombres completos de los alumnos del grupo.

3. 40%: Una presentación de un máximo de 6 minutos (3 minutos cada miembro del grupo), a realizar el Miércoles 1 de Junio durante la clase de teoría. Posteriormente, el fichero .PPT con la presentación debe ser enviada por correo electrónico a pablo.pavon@upct.es (fecha tope el Jueves 2 de Junio a las 20:00). La presentación realizada debe describir claramente:

   - El algoritmo implementado, justificando las decisiones tomadas. Se valorará especialmente la profundidad técnica del algoritmo, y su originalidad (siempre que dé resultados razonablemente buenos).
   - Las referencias consultadas, y cómo se han empleado.

   La presentación no debe dedicar tiempo a explicar el contexto del problema de localización de nodos, ni el enunciado de la práctica. Sólo los aspectos que informan sobre el algoritmo desarrollado.

Se penalizarán los casos que incumplan las restricciones del problema, o sobrepasen en más de un segundo el límite de tiempo fijado con el parámetro `maxExecTimeSecs`.