Universidad Politécnica de Cartagena

TEORÍA DE REDES DE TELECOMUNICACIONES

Grado en Ingeniería Telemática
Grado en Ingeniería en Sistemas de Telecomunicación

Curso 2018-2019

Práctica 9. Problema de localización de nodos

(1 sesión)

Author:

Pablo Pavón Mariño

1 Objetivo

Los objetivos de este laboratorio son

1. Crear algoritmos con Net2Plan para resolver las formulaciones de diversos problemas de localización de nodos, utilizando la librería *Java Optimization Modeler* (JOM).
2. Adquirir experiencia con las diferentes formas de implementar problemas de optimización con JOM, con la representación vectorial.

2 Duracion

Esta práctica esta diseñada para realizarse en una sesión de prácticas de 2 horas.

3 Evaluación

Esta práctica ha sido diseñada para guiar a los alumnos en el aprendizaje de Net2Plan. Las anotaciones que los alumnos hagan en el documento serviran como parte del material utilizado para el estudio de la asignatura, no teniendo que entregar nada al profesor para su evaluación.

4 Documentación

Para realizar la práctica, los alumnos pueden consultar:

- Documentación de la librería JOM (ver `http://www.net2plan.com/jom`).
- Documentación de Net2Plan (ver `http://www.net2plan.com/`).
- Instrucciones en este documento.

5 Trabajo previo

- Leer la sección 7.2 de [1], y los apuntes de la asignatura sobre problema de localización de nodos.
- Repasar la documentación de JOM `http://www.net2plan.com/jom`.

6 Problema de localización de nodos

Sea $\mathcal{N}$ un conjunto de localizaciones. En cada localización se ubica un *nodo de acceso* , entidad que inyecta tráfico a la red. Cada localización puede albergar también uno o ningún *nodo troncal*. Cada nodo de acceso debe estar conectado a un nodo troncal. Y cada nodo troncal puede tener conectados hasta K nodos de acceso. Se denota c_{ij} el coste de conectar un nodo de acceso ubicado en una localización i a un nodo troncal ubicado en una localización j. C es el coste de adquirir un nodo troncal (donde sea que se ubique).

Estamos interesados en determinar cuantos nodos troncales serán necesarios y el emplazamiento de los mismos asi como resolver como los nodos de acceso se conectan a ellos, tal que el coste total de la red sea mínimo. El problema se formula como sigue:

- Parametros de entrada (Constantes conocidas):
  - $\mathcal{N}$: Conjunto de localizaciones.
  - $c_{ij}, i, j \in \mathcal{N}$: Coste de conectar un nodo de acceso i a un nodo troncal j.
  - C: Coste de un nodo tronacl.
  - K: Número máximo de nodos de acceso que se pueden conectar a un mismo nodo troncal.
- Variables de decisión:
  - $z_j, j \in \mathcal{N}$: 1 si el nodo troncal esta situado en la localización j, 0 en caso contrario.
  - $e_{ij}, i, jin\mathcal{N}$: 1 si el nodo de acceso situado en i esta conectado al nodo troncal situado en j, 0 en caso contrario.
- Formulación:

$$\min \quad C\sum_j z_j + \sum_{ij} c_{ij}e_{ij}, \quad \text{sujeto a:} \tag{1a}$$

$$\sum_j e_{ij} = 1, \quad \forall i \in \mathcal{N} \tag{1b}$$

$$\sum_i e_{ij} \leq K z_j, \quad \forall j \in \mathcal{N} \tag{1c}$$

La función objetivo (1a) suma el coste de la red (nodos troncales y enlaces con los nodos de acceso). Las restricciones (1b) hacen que cada nodo de acceso este conectado exáctamente a un nodo troncal. Finalmente, (1c) fuerza a que si una localización no tiene ningún nodo troncal ($z_j = 0$), entonces ningún nodo de acceso puede estar conectado a el. Si la localización tiene un nodo troncal ($z_j = 1$), entonces, como mucho hasta K nodos de acceso pueden estar conectados a el.

7 Algoritmo Net2Plan

El alumno debe desarrollar el algoritmo Net2Plan para resolver el problema (1). El algoritmo recibirá una topología con nodos como entrada. Los nodos se asumen que son las localizaciones de los nodos de acceso, que al mismo tiempo son localizaciones potenciales para los nodos troncales. El algoritmo debe producir como resultado un diseño que refleje donde situar los nodos troncales y como se conectan los nodos de acceso a ellos. Para desarrollar el algoritmo debe seguir los siguientes pasos:

1. Copiar el fichero `AlgorithmTemplate.java` del Aula Virtual y renombrarlo como `NodeLocation.java`. Editar el fichero.
2. El algoritmo debe tener un parámetro de entrada llamado `C`, con valor por defecto 10. Éste es el C de la ecuación (1). También debe haber un parámetro de entrada llamado `K`, con valor por defecto 5. Éste es el K de la ecuación (1).
3. Eliminar todos los enlaces de la red.
4. Crear un objeto tipo `OptimizationProblem` (p.ej. de nombre `op`).

5. Añadir las variables de decisión `z_j`: una por cada nodo de la red. La coordenada j-th corresponde al objeto `Node` de índice j.

6. Añadir la variable de decisión con nombre `e_ij`: un array 2D de variables con una variable por cada par de nodos de la red. La coordenada (i, j)-th corresponde a la existencia o no de una conexión entre el nodo de acceso de índice i y el nodo troncal de índice j.

7. Establezca los parámetros de entrada de JOM:

   - `C`.
   - `K`.
   - Consideramos que los costes c_{ij} son la distancia euclídea entre las localizaciones (nodos) con índice (i, j). Utilizar el método:

     `getMatrixNode2NodeEuclideanDistance`

     del objeto textttNetPlan para obtener la matriz con tantas filas y columnas como nodos, siendo la coordenada (i, j) de la matriz, la distancia euclídea entre los nodos con índice (i, j).

8. Establezca la función objetivo del problema. Recuerde que en JOM, el operador `.*` se puede utilizar para hacer la multiplicación de dos arrays, elemento a elemento y que la función de JOM `sum` (sin argumentos),suma todos los elementos de un array.

9. Utilice un bucle `for` con tantas iteraciones como nodos, y añada las restricciones (1b). Establezca como parámetro de entrada de JOM el variable `i` del bucle, con el indice del nodo de acceso. Utilice la función de JOM `sum` sobre la fila i-th del array `e_ij`.

10. Utilice un bucle `for` con tantas iteraciones como nodos, y añada las restricciones (1c). Establezca como parámetro de entrada de JOM la variable `j` del buble, con el índice del posible nodo troncal. Utilice la función de JOM `sum` sobre la columna j-th del array `e_ij`.

11. Llame al solver para encontrar la solución numérica. Como es un problema lineal con variables de decisión enteras, el solver a utilizar será `glpk`.

12. Obtenga la solución primal.

13. Guarde el resultado: enlaces de nodos de acceso a troncales en el diseño (no como enlaces bidireccionales). Todos los enlaces serán de capacidad 1, con una longitud igual a la distancia euclídea de los nodos que une (nodo de acceso a troncal), (utilice el método `getNodePairEuclideanDistance`), y velocidad de propagación de 200000 km/s.

14. Devuelva un mensaje que contenga (i) el número total de nodos troncales, (ii) el coste de la topología obtenida (utilice el método `getOptimalCost` del objeto `OptimizationProblem`).

7.1 Chequear el algoritmo

Cargue la topología `NSFNet_N14_E42.n2p`. El algoritmo deberia producir una solución con un total de 6 nodos troncales y un coste de 103.8 unidades.

Quiz 1. Rellene la siguiente tabla, ejecutando el algoritmo sobre la topología `NSFNet_N14_E42.n2p` para diferentes valores de `K` (máxima conectividad) y `C` (coste de ubicar un nodo troncal).

test de topología NSFNET

`C`	# nodos troncales con `K=5`	# nodos troncales con `K=14`
0	14	14
5	11	11
10	6	6
12	5	4
15	4	3
20	3	3
1000	3	3

Responda a las siguientes preguntas:

- ¿Cómo afecta K en el número de nodos troncales?

  Answer: Lower K make that in some occasions more core nodes are needed, since the existing ones are already congested (connected to K access nodes).

- ¿Por qué el número de nodos troncales en `K=5` no es inferior a 3?

  Answer: Because then I need at least 3 core nodes to connected to 14 access nodes.

- ¿Por qué el número de nodos troncales no aumenta con C?

  Answer: Because a higher cost of the core node is never an incentive to buy more of them.

8 Variaciones del problema

Quiz 2. Modifica el algoritmo añadiendo la restriccion de que el máximo número de nodos troncales está limitado por el parámetro de entrada `M`. La restricción a añadir a (1) es:

$$\sum_j z_j \leq M$$

Tenga en cuenta que si $M < N/K$ el problema no tiene solución, ya que no hay suficientes nodos troncales para conectar a todos los nodos de acceso.

9 Forma matricial de las restricciones (opcional)

Asumiendo que las variables de decisión z_j son definidas en JOM como un vector fila de $1 \times |\mathcal{N}|$, y e_{ij} como una matriz $|\mathcal{N}| \times |\mathcal{N}|$.

Entonces, de acuerdo a la sintaxis de JOM:

- `sum (e_ij , 2)` suma los valores en la segunda dimensión de `e_ij` (las columns). Entonces, produce un vector columna ($|\mathcal{N}| \times 1$), donde la coordenada i-th es $\sum_j e_{ij}$.
- `sum (e_ij , 1)` suma los valores en la primera dimensión de `e_ij` (las filas). Entonces, produce el vector fila ($1 \times |\mathcal{N}|$), donde la coordenada j-th es $\sum_i e_{ij}$.

De acuerdo a esto, las restricciones (1b) se pueden introducir en una única llamada al método `addConstraint`, como una restricción vectorial, tal y como se indica:

```
sum (e_ij , 2) == 1
```

Asimismo, las restricciones (1c) se pueden introdudir con una única llamada al método `addConstraint`, como una restricción vectorial, tal y como se indica:

```
sum (e_ij , 1) <= K * z_j
```

Quiz 3. Reescriba el algoritmo utilizando la forma matricial.

10 Trabajo en casa tras la práctica

El alumno puede completar todos los *Quizs* que no ha podido finalizar durante la sesión de prácticas

Bibliography

[1] *P. Pavón Mariño, "Optimization of computer networks. Modeling and algorithms. A hands-on approach", Wiley 2016.*

Bibliography