Universidad Politécnica de Cartagena

TEORÍA DE REDES DE TELECOMUNICACIONES

Grado en Ingeniería Telemática
Grado en Ingeniería en Sistemas de Telecomunicación

Curso 2018-2019

Práctica 7. Optimización del encaminamiento y la capacidad modulares utilizando una formulación destino-enlace

(1 sesión)

Autor:

Pablo Pavón Mariño

1. Objetivos

Los objetivos de esta práctica son:

1. Crear algoritmos de Net2Plan que resuelvan un problema CFA, donde las capacidades y el encaminamiento se optimizan de forma conjunta, resolviendo una formulación destino-enlace del problema utilizando la librería *Java Optimization Modeler* (JOM).
2. Conseguir práctica en las diferentes formas de escribir problemas de optimización con JOM, aprovechando sus capacidades de representación vectorial.
3. Explorar el potencial de la representación matricial de las restricciones de encaminamiento en JOM y Net2Plan.

2. Duración

Esta práctica está diseñada para una sesión de 2 horas.

3. Evaluación

Esta práctica ha sido diseñada para guiar al estudiante en su aprendizaje en Net2Plan. Las anotaciones que los estudiantes hagan en este documento son para su uso cuando repasen la asignatura y no es necesario que se entregue al profesor para su evaluación.

4. Documentación

Los recursos necesarios para este trabajo de laboratorio son:

- Documentación de librería de JOM (`http://www.net2plan.com/jom`).
- La herramienta Net2Plan y su documentación (`http://www.net2plan.com/`).
- Instrucciones de este guión de prácticas.

5. Trabajo previo antes de venir al laboratorio

- Leer la sección 4.4 de [1], y los apuntes de teoría relacionados con la formulación destino-enlace.
- Refrescar las lecturas de la documentación de JOM en `http://www.net2plan.com/jom`, y en particular, como se manejan los vectores de variables y restricciones.

6. Encaminamiento y asignación de capacidad con capacidades modulares

Sea $\mathcal{G}(\mathcal{N}, \mathcal{E})$ una red con un conjunto de nodos $\mathcal{N}$ y un conjunto de enlaces $\mathcal{E}$. El tráfico ofrecido se compone de un conjunto de demandas unicast $\mathcal{D}$, y el tráfico ofrecido h_d para cada demanda $d \in \mathcal{D}$

es conocido.

La red se basa en equipos que reenvían el tráfico de acuerdo a su nodo destino. Las capacidades de los enlaces están restringidas a ser múltiplos de enteros de C unidades de tráfico.

Estamos interesados en crear un algoritmo de Net2Plan que encuentre el encaminamiento basado en destino y las capacidades de los enlaces que minimizan el coste total de la red, calculado como el número de módulos de C unidades de capacidad a instalar en la red. La formulación a resolver con JOM se define como:

- Parámetros de entrada (constantes conocidas):
  - $\mathcal{N}$: conjunto de nodos de la red.
  - $\mathcal{E}$: conjunto de enlaces de la red.
  - $\mathcal{D}$: conjunto de demandas unicast ofrecidas.
  - $h_d, d \in \mathcal{D}$: tráfico ofrecido por la demanda d.
  - C: Unidades de tráfico de un módulo de capacidad.
- Variables de decisión:
  - $a_e, e \in \mathcal{E}$: número de módulos de capacidad a instalar en el enlace e.
  - $x_{te}, t \in \mathcal{N}, e \in \mathcal{E}$: Cantidad de tráfico con destino el nodo t, que atraviesa el enlace e.
- Formulación:

$$\text{mín} \quad \sum a_e, \quad \text{sujeto a:} \tag{1a}$$

$$\sum_{e \in \delta^+(n)} x_{te} - \sum_{e \in \delta^-(n)} x_{te} = \begin{cases} h_{nt} \text{ if } n \neq t \\ -\sum_{n'} h_{n't} \text{ if } n = t \end{cases}, \quad \forall n, t \in \mathcal{N} \tag{1b}$$

$$\sum_t x_{te} \leq a_e C, \quad \forall e \in \mathcal{E} \tag{1c}$$

$$a_e \in \{0, 1, 2, \ldots\}, \quad \forall e \in \mathcal{E} \tag{1d}$$

$$x_{te} \geq 0, \quad \forall t \in \mathcal{N}, e \in \mathcal{E} \tag{1e}$$

La función objetivo (1a) minimiza el número total de módulos de capacidad (nuestra estimación del coste de la red). Las restricciones (1b) son las estándar de conservación de flujo para la formulación destino-enlace. Las restricciones (1c) representan qeu, para cada enlace, el tráfico cursado en el enlace es menor o igual a su capacidad (y por tanto el enlace no se satura). Finalmente, la restricciones (1de) prohiben instalar una capacidad negativa en los enlaces, o llevar una cantidad de tráfico negativo, ya que no tiene sentido físico.

7. Algoritmo en Net2Plan

El estudiante debe implementar el algoritmo de Net2Plan que resuelva el problema (1) siguiendo los siguientes pasos:

1. Copia el fichero `AlgorithmTemplate.java` que se encuentra en el Aula Virtual y renombrarlo como `DestinationLinkModularCapacities.java`.

2. El algoritmo debe tener un parámetro de entrada llamado `capacityModule`, con valor por defecto 10, y una descripción *Size of the capacity module measured in the same units as the traffic*. Este parámetro es la constante C en (1).

3. Este algoritmo funcionará para cualquier tipo de encaminamiento del diseño de entrada. Se tiene que eliminar cualquier encaminamiento cursado de la red (rutas en *source routing*, o reglas de encaminamiento en el encaminamiento salto a salto). Se puede hacer llamando al método `removeAllUnicastRoutingInformation` del objeto de entrada `NetPlan`.

4. Calcular la matríz de tráfico (con tantas filas y columnas como nodos) a partir de las demandas ofrecidas, utilizando el método

   getMatrixNode2NodeOfferedTraffic

   de la clase `NetPlan`. Tenga en cuenta que se devuelve un objeto `DoubleMatrix2D`. Éste se puede converitr en un `double [][]` usando el método *toArray* de `DoubleMatrix2D`.

5. Crear un objeto del tipo `OptimizationProblem` (p.ej. de nombre `op`).

6. Añadir las variables de decisión del problema, con nombre `x_te`: una variable por nodo y enlace. El valor mínimo que pueden tomar esas variables es 0 y el máximo `Double.MAX_VALUE`.

7. Añadir las variables de decisión del probleme con nombre `a_e`: una variable por enlace. El valor mínimo que pueden tomar esas variables es 0 y el máximo `Integer.MAX_VALUE`, ya que estas variables están restringidas a tomar solo valores enteros.

8. Establecer la función objetivo del problema utilizando la función de JOM `sum`.

9. Utilizar dos bucles anidados para añadir las restricciones de conservación de flujo. El bucle `for` externo itera los nodos de los que estamos haciendo la conservación (n in (1b)), y el interno en los nodos destino(t in (1)b). Para añadir la restricción de conservación de un nodo `n` a un nodo destino `t`:

   - Se establecen los parámetros de entrada de JOM:
     - `deltaPlus` con los índices de los enlaces de salida del nodo actual. Para ello, utilizar el método *getOutgoingLinks* del objeto `Node` para conocer los enlaces de salida, y el método *NetPlan.getIndexes* para convertir la colección de enlaces a sus índices.
     - `deltaMinus` con los índices de los enlaces entrantes del nodo actual. `n`. Para ello, utiliza el método *getIncomingLinks* del objeto `Node` para obtener los enlaces entrantes, y el método *NetPlan.getIndexes* para convertir la colección de nelaces en sus índices.
     - `h_nt` con la coordenada de la matríz de tráfico con el tráfico del nodo `n` al nodo destino `t`.
     - La entrada `h_t` ($h_t = \sum_{n'} h_{n't}$) se puede generar facilmente utilizando el método:

       `getVectorNodeEgressUnicastTraffic`

     de la clase `NetPlan`, que devuelve la cantidad total de tráfico ofrecido dirigido a un nodo dado.
   - Establecer la restricción utilizando la función `sum`, sobrer `x_te`, pero restringido a la suma de los elementos en la filat y las columnas en`deltaPlus` o `deltaMinus`.

10. Añadir el parámetro de entrada de JOM `C`, con el valor dado por el parámetro de entrada del algoritmo `capacityModule` (la constante C en (1)).

11. Utiliar un bucle for con un iterador por enlace, para añadir las restricciones de (1c). Se añade una restriccion en cada iteración del bucle. Para añadir la restriccion de un enlace`e` se utiliza la función `sum`, sobre todos los nodos destino `t` (utilizando la palabra clave de JOM `all` en la coordenada de destino), y el índice del enlaceindex `e`.

12. Llamar al solver para encontrar la solución numérca. Utilizar la opción `maxSolverTimeInSeconds` para establecer un tiempo máximo del solver de 10 segundos. el solver devolverá la mejor solución encontrada después de 10 segundos, siempre y cuando no se haya encontrado la mejor solución antes. Puede ocurrir que el solver no encuentre una solución factible. Para comprobar esa situación, se debe utilizar el método `solutionIsFeasible` de `OptimizationProblem`, y lanzar una excepción si el solver no encuentra solución.

13. Recuperar la solución primal encontrada, y convertirla a un objeto *DoubleMatrix2D*, utilizando el método *view2D*.

14. Utilizar el método de *NetPlan* llamado*setRoutingFromDestinationLinkCarriedTraffic*. Este método crea automáticamente las rutas de la red que son constistentes con lo que aparece en la matriz 2D x_{te} recuperada. Ya que la formulación puede crear soluciones óptimas con bucles (bucles en enlaces con ancho de banda disponible que no causan la necesidad de más módulos de capacidad), establecer la opción en este método que automáticamente las elimina.

15. Establecer las capacidades en los enlaces en el diseño `netPlan` como el número de módulos para ese enlace multiplicado por la capacidad del módulo.

7.1. Comprobar el algoritmo

Cargue la red `example7nodesWithTraffic.n2p` en Net2Plan. Ejecute el algoritmo desarrollado sobre esta red. Éste debería producir una solución con un total de 200 unidades de capacidad instaladas en la red (20 módulos) cuando el tamaño de los módulos es de 10.

Reduzca el tamaño de los módulos a 1 y ejecute de nuevo el algoritmo. En este caso la capacidad óptima instalada es de 158.

Quiz 1. ¿Por qué la capacidad total instalada es menor cuando se reduce el tamaño de los módulos?

8. Variaciones del problema

Quiz 2. Modifique el problema original (1) tal que ahora el coste de cada módulo de capacidad sea proporcional a la longitud del enlace en km. En este caso, la función objetivo cambia, y toma la siguiente forma:

$$\text{mín} \quad \sum_e d_e a_e$$

donde d_e es la longitud del enlace, que lo devuelve el método `getLengthInKm()` del objeto `Link`.

9. Forma matricial de las restricciones del problema(opcional)

9.1. Restricciones de conservación de flujo

Las restricciones de conservación de flujo en (1b) consisten en una restriccion por nodo n y una por nodo destino t ($|\mathcal{N}|^2$ constraints).

Todas las restricciones se pueden representar como una única igualdad matricial como sigue:

$$A_{ne} \times x'_{te} = H_{st} \tag{2}$$

donde:

- A_{ne} es la matriz de incidencia de enlace. Esto es, $|\mathcal{N}| \times |\mathcal{E}|$ con una fila por nodo y una columna por enlace. La coordenada (n, e) es 1 si el enlace e es un enlace saliente del nodo n, -1 si es un enlace entrante de n, y 0 en caso contrario.

    - La matriz de incidencia de enlace se puede obtener como una matriz sparse en Net2Plan utilizando el siguiente método de la clase `NetPlan`:

      `getMatrixNodeLinkIncidence`

- x_{te} es una matriz $|\mathcal{N}| \times |\mathcal{E}|$, y x'_{te} es su traspuesta, con las variables de decisión.
- H_{st} es la matriz de tráfico $|\mathcal{N}| \times |\mathcal{N}|$, con una ligera modificación: las coordenadas de (n, n) en la diagonal contienen los valores $-\sum_{n'} h_{n'n}$, por tanto, la suma de cada columna de H_{st} es 0.

9.2. Restricciones de capacidad enlace- caso fraccional

Las restricciones de capacidad de enlace en (1c) consisten en una restriccion por enlace ($|\mathcal{E}|$ restricciones):

Todas las restricciones pueden representarse como una única desigualdad vectorial como sigue:

$$\sum_t x_{te} \leq Ca \tag{3}$$

wheredonde

- $\sum_t x_{te}$ cambia ahora a un vector $1 \times |\mathcal{E}|$, con el tráfico cursado de cada enlace (utilizar el comando `sum` de JOM sobre la primera dimensión de `x_te` para ello).
- a representa el vector $1 \times |\mathcal{E}|$ con el número de módulos a instalar en cada enlace.

Quiz 3. Reescribe las restricciones de conservación de flujo y capacidad de los enlaces utilizando la forma matricial.

10. Trabajo en casa después de la sesión de prácticas

El estudiante debería completar todos los *Quizs* que no haya podido finalizar durante la sesión de prácticas.

Bibliografía

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