Universidad Politécnica de Cartagena

TEORÍA DE REDES DE TELECOMUNICACIONES

Grado en Ingeniería Telemática
Grado en Ingeniería en Sistemas de Telecomunicación

Curso 2018-2019

Práctica 4. Introducción a la librería Java Optimization Modeler (JOM) con Net2Plan

(1 sesión)

Autor:

Pablo Pavón Mariño
Juan Pedro Muñoz Gea
María Victoria Bueno Delgado
Ángel Antonio Pintado Sedano
Juan Carlos Sánchez Aarnoutse

1. Objetivos

Los objetivos de esta práctica son:

1. Introducir al alumno en la utilización de la librería *Java Optimization Modeler* (JOM) para la resolución de problemas de optimización matemática.
2. Modelar y resolver con JOM problemas sencillos de red implementados como algoritmos de Net2Plan así como revisar algunos conceptos teóricos y sus derivadas.

2. Duración

Esta práctica está diseñada para una sesión de dos 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:

- La documentación de librería JOM (ver `http://www.net2plan.com/jom`).
- La herramienta Net2Plan y su documentación (ver `http://www.net2plan.com/`).
- Las instrucciones de esta práctica.

5. Trabajo previo antes de venir al laboratorio

- Leer la documentación de JOM en `http://www.net2plan.com/jom`, en particular los ejemplos de la página principal, secciones *Getting started* y *JOM syntax at a glance*.
- Familiarizarse con los métodos de la clase `OptimizationProblem` que se pueden consultar en el correspondiente Javadoc.

6. Instalación del los resolvers de JOM

Importante: JOM es una interface de Java para un conjunto de solvers numéricos. Estos solvers deben haber sido previamente instalados en el sistema.

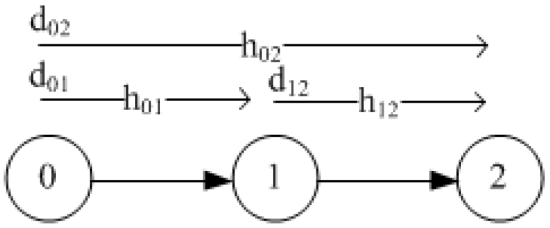

Figura 1: Red de tres nodos

En este curso se empleará el solver GLPK para los problemas de programación lineal (enteros o no enteros) y el solver IPOPT para los problemas convexos diferenciables. **Ambos solvers están ya instalados en los equipos del laboratorio.**

Para la instalación en casa: Siga las siguientes instrucciones:

`http://www.net2plan.com/jom/installation.php`

A modo de resumen (para Windows):

- Compruebe si su Máquina Virtual de Java (*Java Virtual Machine*) es de 32 o 64 bits por medio del comando `java -version`. Debe ejecutar este comando en una consola terminal de Windows (inicio->ejecutar->cmd).
- Descargue las versiones de los archivos *DLL* para GLPK e IPOPT que correspondan con su versión (32 o 64 bits).
- Para permitir que estén inmediatamente disponibles para la configuración por defecto de Net2Plan, renombre los archivos como `glpk.dll` e `ipopt.dll` y colóquelos en el directorio `c:\windows\system32` de su equipo.

7. Plantilla de partida

Esta sesión de laboratorio está basada en varios problemas planteados para optimizar la cantidad de tráfico inyectado en la red por tres demandas en una topología de red como la mostrada en la Figura ??.

Este caso de estudio se encuentra disponible en el aula virtual, en la plantilla `ThreeNodesBATemplate.java`. Observe el archivo para comprobar su estructura. En las siguientes secciones modificará el archivo de diferentes formas.

8. Maximizando el throughput

Esta sección está planteada para que cree un algoritmo que, por medio de JOM, encuentre la asignación de ancho de banda óptima, la cual maximiza el total de tráfico ofrecido en la red.

El problema de optimización a resolver es el siguiente:

- Parámetros de entrada:

- U: La capacidad de los dos enlaces (valor del parámetro `linkCapacity`)

- Variables de decisión:
  - h_{12}: El tráfico inyectado por la demanda d_{12}.
  - h_{23}: El tráfico inyectado por la demanda d_{23}.
  - h_{13}: El tráfico inyectado por la demanda d_{13}.

- Formulación:

$$\max_{h} \quad h_{12} + h_{23} + h_{13}, \quad \text{sujeto a:} \tag{1a}$$
$$h_{12} + h_{13} \leq U \tag{1b}$$
$$h_{23} + h_{13} \leq U \tag{1c}$$
$$h_{12} \geq 0 \tag{1d}$$
$$h_{23} \geq 0 \tag{1e}$$
$$h_{13} \geq 0 \tag{1f}$$

Para resolver este problema siga los siguientes pasos:

1. Copie el archivo `ThreeNodesBATemplate.java` y renómbrelo como `ThreeNodesBATemplate_maxThroughput.java`. Comience modificándolo en el lugar mencionado en el archivo plantilla (en el lugar que dice "** The optimization of the offered/carried traffic should start here **").

2. Cree un objeto de la clase *OptimizationProblem*. Vea el Javadoc de la API de JOM disponible en el menú de ayuda (*Help*) de la herramienta Net2Plan.

3. Use el método *setInputParameter* para crear un parámetro de JOM con el nombre U, con el mismo valor que el parámetro del algoritmo *linkCapacity*.

4. Use el método *addDecisionVariable* para crear tres variables de decisión con los nombres *h12*, *h23* y *h13*. Dado que son variables escalares, en el parámetro que indica el tamaño (*size*) tendrán el valor *new int [] 1 , 1*. Establezca el valor más bajo posible de las variables a cero, y el máximo valor posible a *Double.MAX_ VALUE* (no hay un valor más grande). Observe que las variables no deben ser restringidas a entero.

5. Use el método *setObjectiveFunction* para establecer la función objetivo del problema como (*h12 + h23 + h13*).

6. Use el método *addConstraint* para añadir las restricciones (**??**b) y (**??**c). Observe que las restricciones no negativas ya están forzadas por el límite inferior establecido en las variables de decisión.

7. Dado que el problema es linear, sin restricciones enteras, puede ser resuelto con ambos solvers, tanto GLPK como IPOPT. Utilice el método *solve* para resolverlo usando el solver GLPK.

8. Utilice el método *solutionIsOptimal* para comprobar si la solución óptima ha sido alcanzada. En caso negativo, lanza una excepción *Net2PlanException* con el mensaje: "No se ha encontrado una solución óptima" (*throw new Net2PlanException ("An optimal solution was not found");*)

9. Si la solución óptima ha sido encontrada, recupérela usando el método *getPrimalSolution*, que devuelve un objeto de la clase *DoubleMatrixND*. Esta clase puede manejar arrays de un número de dimensiones arbitrario de una forma eficiente. Ahora las variables de decisión son arrays del tamaño $(1, 1)$ (escalares). En este caso, es posible usar el método *toValue ()* para convertir el objeto a un *double*.

10. Establezca los tráficos inyectados computados en el diseño: establezca el tráfico ofrecido de las demandas y el tráfico cursado por las rutas y las capacidades de los enlaces ocupados.

Comprobación. Ejecute el algoritmo y compruebe que la solución hace que la demanda d_{13} que atraviesa dos enlaces inyecta cero tráfico y que las otras dos demandas (d_{12}, d_{23}) inyectan tanto tráfico como capacidad tiene el enlace. El tráfico total ofrecido es dos veces la capacidad del enlace.

Quiz 1. Cambie el algoritmo para que fuerce a que el tráfico de las demandas d_{12} y d_{23} deba ser un entero múltiplo de 2 unidades de tráfico. Para hacer esto, observe que d_{12} y d_{23} deberían ahora estar restringidas a entero y contener un número módulo de 2 unidades de tráfico inyectado (e.g. la cantidad de tráfico inyectada actualmente por d_{12} debe venir dada por su variable de decisión multiplicada por dos). Para comprobar los resultados: si *linkCapacity* es puesto a 3, entonces las solución óptima es aquella en la que d_{13} inyecta una unidad de tráfico, y d_{23} y d_{12} inyectan dos unidades de tráfico.

9. Maximizando la utilidad del algoritmo

El problema de optimización a resolver en esta sección es:

- Parámetros de entrada:
  - U_{12}: La capacidad del enlace 1-2 (parámetro `linkCapacity12`)
  - U_{23}: La capacidad del enlace 2-3 (parámetro `linkCapacity23`)
- Variables de decisión:
  - h_{12}: El tráfico inyectado por la demanda d_{12}.
  - h_{23}: El tráfico inyectado por la demanda d_{23}.
  - h_{13}: El tráfico inyectado por la demanda d_{13}.
- Formulación:

$$\max_{h} \quad \log h_{12} + \log h_{23} + \log h_{13}, \quad \text{sujeto a:} \tag{2a}$$
$$h_{12} + h_{13} \leq U_{12} \tag{2b}$$
$$h_{23} + h_{13} \leq U_{13} \tag{2c}$$
$$h_{12} \geq 0 \tag{2d}$$
$$h_{23} \geq 0 \tag{2e}$$
$$h_{13} \geq 0 \tag{2f}$$

Aquí $\log(x)$ representa el logaritmo natural de x. Para resolver el problema (**??**), modifique el algoritmo de la sección anterior. Primero copie y renombre el algoritmo a `ThreeNodesBATemplate_maxLog.java`. En la función objetivo, use la función para maximizar:

```
ln(h12) + ln(h23)+ ln(h13)
```

Dado que en JOM los logaritmos naturales se escriben como *ln*.Observe que puesto que el problema es no lineal se debe emplear el solver IPOPT.

Además, el problema debe poner el identificador "Constraint12" para la restricción (**??**b). Esto se hace poniendo el parámetro identificador en el método *addConstraint*. El identificador no cambia en la restricción pero permite recuperar posteriormente el multiplicador óptimo asociado a esa restricción.

El algoritmo debe imprimir en la salida el siguiente mensaje:

- El beneficio óptimo obtenido: $\log h_{12} + \log h_{23} + \log h_{13}$.
- El multiplicador óptimo asociado a la restricción (**??**b). Éste es obtenido empleando el método *getMultiplierOfConstraint*. Observe que el multiplicador puede ser negativo, en contra de lo que dice la teoría. Eso es debido a una convención interna de IPOPT que devuelve multiplicadores negativos en problemas de maximización con restricciones $\leq$-.

Comprobación Ejecute el algoritmo con la capacidad de enlace igual a uno. Compruebe que la solución es aquella en la que la demanda d_{13} atravesando dos enlaces inyecta $1/3$ unidades de tráfico, y que las otras demandas (d_{12}, d_{23}) inyectan $2/3$ unidades.

9.1. Forma vectorial de las variables de decisión

La librería JOM permite organizar las variables de decisión y las restricciones en arrays de un número arbitrario de dimensiones.

Rehaga el algoritmo `ThreeNodesBATemplate_maxLog.java`, cambiando su nombre a `ThreeNodesBATemplate_maxLog_vect.java`, con las siguientes modificaciones:

- Las tres variables de decisión son creadas en una única llamada al método *addDecisionVariable*, ahora son dispuestas en un único vector de 1×3 con el nombre *h*. Ahora *h12* será *h(0)*, *h23* será *h(1)* y *h13* será *h(2)*.
- La función objetivo puede ser escrita de dos formas:
  - Expandiendo la suma: *ln(h(0)) + ln(h(1)) + ln(h(2))* o también *ln(h(0,0)) + ln(h(0,1)) + ln(h(0,2))*.
  - Usando la función suma (*sum*): *sum(ln(h))*. En este caso, *ln(h)* es un vector 1×3, con el logaritmo natural de cada coordenada de h, y *sum* produce la suma de todos los elementos del array $ln(h)$.
- Reescriba las restricciones y el resto del algoritmo adecuadamente.
- Para recuperar el valor óptimo de la función objetivo puede obtener el objeto *DoubleMatrixND* con el método *getPrimalSolution ("h")*, a continuación puede convertirlo en un *double []* con el método *to1DArray*.

Comprobación. Ejecute el algoritmo y compruebe que el resultado no cambia.

9.2. Función de perturbación

En esta sección estamos interesados en (i) observar cómo el beneficio óptimo cambia si cambiamos la capacidad del enlace 1-2, manteniendo inalterado en enlace 1-3, y (ii) comprobar que el multiplicador óptimo de la restricción (**??**b) permite contruir un estimador optimista de cómo el beneficio óptimo cambia (la pendiente de la recta tangente a la función de perturbación).

Siga los siguientes pasos:

1. Ejecute el algoritmo de optimización para los valores:

$$U_{12} = \{0{,}5, 0{,}6, \ldots, 1{,}5\}$$

2. Represente una gráfica de la función de perturbación: los puntos $(U_{12}, p^*(U_{12}))$, donde $p^*(U_{12})$ es el beneficio óptimo obtenido cuando el problema se resuelve con el valor de dicho U_{12}.

3. En la misma gráfica, dibuje una línea que pase por el punto $(1, p^*(1))$, y que tenga la pendiente dada por el negativo del multiplicador de la restricción. Obsérvese que, como predice la teoría, la línea es tangente a la función de perturbación (que es cóncava ya que estamos en un problema de maximización), y por lo tanto se convierte en un estimador optimista para ella.

Quiz 2. Escriba el resultado teórico que apoya esta observación.

10. Trabajo para casa después de la sesión de laboratorio

El estudiante debería completar todos los *Quizs* que no haya podido finalizar durante la sesión en el laboratorio.