Universidad Politécnica de Cartagena

TEORÍA DE REDES DE TELECOMUNICACIONES

Grado en Ingeniería Telemática
Grado en Ingeniería en Sistemas de Telecomunicación

Curso 2015-2016

Lab work #4. Introduction to Java Optimization Modeler (JOM) library in Net2Plan

(1 session)

Author:

Pablo Pavón Mariño

1 Objectives

The goals of this lab work are:

1. Introduce the utilization of the *Java Optimization Modeler* (JMO) library, to solve mathematical optimization problems.
2. Model and solve with JOM simple network problems implemented in Net2Plan algorithms, and review some theoretical concepts as a by-product.

2 Duration

This lab work is designed for one session of two hours.

3 Evaluation

This lab work has been designed to guide the students in their learning of Net2Plan. The annotations the students make in this document are for their use when studying the course, and do not have to be delivered to the teacher for evaluation.

4 Documentation

The resources needed for this lab work are:

- JOM library documentation (see `http://www.net2plan.com/jom`).
- Net2Plan tool and their documentation (see `http://www.net2plan.com/`).
- Instructions in this wording.

5 Previous work before coming to the lab

- Read the JOM documentation in `http://www.net2plan.com/jom`, in particular, the examples in the main page, sections *Getting started* and *JOM syntax at a glance*.
- Get familiar with the methods of the class `OptimizationProblem`, reading its Javadoc.

6 Installing the JOM solvers

Important: JOM is an interface from Java, to a set of numerical solvers. These solvers must be previously installed in the system.

In this course we will use the GLPK solver for linear programs (integer or not) and IPOPT for differentiable convex problems. **Both solvers are already installed in your computer in the lab.**

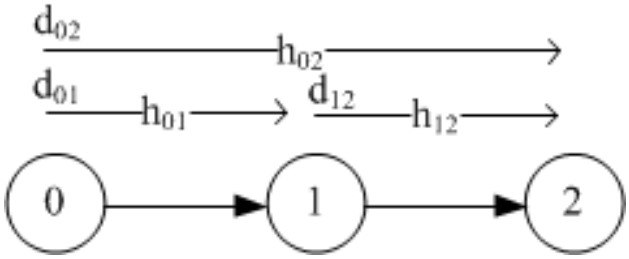

Figure 1: Three node network

For home installations: Follow the instructions in:

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

A summary follows (for Windows):

- Check if your Java Virtual Machine is of 32 or 64 bits, using the command `java -version`.
- Download the appropriate *DLL* files for GLPK and IPOPT (32 or 64 bits).
- To make them immediately available from Net2Plan default configuration, rename the files as `glpk.dll` and `ipopt.dll` and place them in the `c:\windows\system32` folder or your computer.

7 Starting template

This lab session is based on several problems that intend to optimize the amount of injected traffic for three demands, in a network topology as the one shown in Fig. 1.

Such case study is created in the template file `ThreeNodesBATemplate.java` available in Aula Virtual. Take a look at the file to see its structure. The following sections modify the file in different forms.

8 Maximizing the throughput

This section is intended to create an algorithm that finds using JOM the optimum bandwidth assignment, which maximizes the total offered traffic in the network.

The optimization problem to solve is:

- Input parameters:
  - U: The capacity of the two links (value of parameter `linkCapacity`)
- Decision variables:
  - h_{12}: The traffic injected by demand d_{12}.
  - h_{23}: The traffic injected by demand d_{23}.
  - h_{13}: The traffic injected by demand d_{13}.

- Formulation:

$$\max_{h} \quad h_{12} + h_{23} + h_{13}, \quad \text{subject to:} \tag{1a}$$

$$h_{12} + h_{13} \leq U \tag{1b}$$

$$h_{23} + h_{13} \leq U \tag{1c}$$

$$h_{12} \geq 0 \tag{1d}$$

$$h_{23} \geq 0 \tag{1e}$$

$$h_{13} \geq 0 \tag{1f}$$

To solve this problem, follow the next steps:

1. Copy the `ThreeNodesBATemplate.java` file and rename it as `ThreeNodesBATemplate_maxThroughput.java`. Start modifying it in the place mentioned in the template.
2. Create an object of the class *OptimizationProblem*. See the JOM API Javadoc accessible from *Help* menu in Net2Plan tool.
3. Use the method *setInputParameter* to create a JOM parameter of name U, with the same value as the algorithm parameter *linkCapacity*.
4. Use the method *addDecisionVariable* to create three decision variables, of names *h12*, *h23* and *h13*. Since they are scalar variables, they have a size parameter of *new int [] 1 , 1*. Set the variables lower possible values to zero, and the maximum possible value to *Double.MAX_ VALUE* (no upper value). Note that the variables should not be constrained to be integer.
5. Use the method *setObjectiveFunction* to set the problem objective function (*h12 + h23 + h13*).
6. Use the method *addConstraint* to add the constraints (1b) and (1c). Note that non-negativity constraints are already enforced by the lower limits set to the decision variables.
7. Since the problem is linear, without integer constraints, it can be solved both by GLPK or IPOPT solver. Use the method *solve* to solve it using the GLPK solver.
8. Use the method *solutionIsOptimal* to check if an optimum solution has been reached. If not, raise a *Net2PlanException* with the message: "An optimal solution was not found" (*throw new Net2PlanException ("An optimal solution was not found");*)
9. If an optimal solution was found, retrieve it using the method *getPrimalSolution*, that returns an object of the class *DoubleMatrixND*. This class can handle arrays of an arbitrary number of dimensions in an efficient form. Now the decision variables are arrays of size $(1, 1)$ (scalars). In this case, it is possible to use the method *toValue ()* to convert the object to a *double*.
10. Set the computed injected traffics in the design: set the offered traffic of the demands, and the routes' carried traffic and occupied link capacities accordingly.

Check. Run the algorithm, and check that the solution is that the demand d_{13} traversing two links injects zero traffic, and the other demands (d_{12}, d_{23}) inject as much traffic as the link capacity. The total offered traffic is twice the link capacity.

Quiz 1. Change the algorithm to force that the traffic of demands d_{12} and d_{23} must be an integer multiple of 2 traffic units. To do so, note that d_{12} and d_{23} should be now restricted to be integer, and contain the number of 2-unit modules of traffic injected (e.g. the actual amount of traffic injected by d_{12} would be given by its decision variable multiplied by two). To check the results: if *linkCapacity* is set to 3, then optimum solution is that d_{13} injects one traffic units, and d_{23} and d_{12} inject two traffic units.

9 Maximizing the logarithm utility

The optimization problem to solve in this section is:

- Input parameters:
  - U_{12}: The capacity of the link 1-2 (parameter `linkCapacity12`)
  - U_{23}: The capacity of the link 2-3 (parameter `linkCapacity23`)
- Decision variables:
  - h_{12}: The traffic injected by demand d_{12}.
  - h_{23}: The traffic injected by demand d_{23}.
  - h_{13}: The traffic injected by demand d_{13}.
- Formulation:

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

Here, $\log(x)$ stands for the natural logarithm of x. To solve problem (2), modify the algorithm in the previous section. First, copy and rename the algorithm as `ThreeNodesBATemplate_maxLog.java`. In the objective function, use the function to maximize:

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

Since in JOM, the natural logarithm is written as *ln*. Note that since the problem is not linear, the solver to use is IPOPT.

Also, the problem should put the identifier "Constraint12" to the constraint (2b). This is done setting the identifier parameter in the method *addConstraint*. The identifier changes nothing in the constraint, but permits later retrieving the optimum multiplier associated to that constraint.

The algorithm should print in the output message:

- The optimum benefit obtained: $\log h_{12} + \log h_{23} + \log h_{13}$.

- The optimum multiplier associated to constraint (2b). This is obtained using the method *getMultiplierOfConstraint*. Note that the multiplier may be negative, in contrast to what theory states. This is an internal IPOPT convention, that returns negative multipliers in maximization problems for $\leq$-constraints.

Check. Run the algorithm with link capacity equal to one. Check that the solution is that the demand d_{13} traversing two links injects 1/3 traffic units, and the other demands (d_{12}, d_{23}) inject 2/3 units.

9.1 Vectorial form of the decision variables

JOM library permits organizing the decision variables and the constraints in arrays of an arbitrary number of dimensions.

Remake the algorithm `ThreeNodesBATemplate_maxLog.java`, changing its name to `ThreeNodesBATemplate_maxLog_vect.java`, with the following modifications:

- The three decision variables are created in a single call to *addDecisionVariable* method, and are now arranged in a single 1×3 vector called h.Now *h12* will be *h(0)*, *h23* will be *h(1)* and *h13* will be *h(2)*.
- The objective function can be written in two forms:
  - Expanding the sum: *ln(h(0)) + ln(h(1)) + ln(h(2))* or also *ln(h(0,0)) + ln(h(0,1)) + ln(h(0,2))*.
  - Using the *sum* function: *sum(ln(h))*. In this case, *ln(h)* is a vector 1×3, with the natural logarithm of each coordinate of h, and *sum* produces the sum of all the elements of the array $ln(h)$.
- Rewrite the constraints and the rest of the algorithm appropriately.
- To retrieve the optimum value of the objective function, get the *DoubleMatrixND* object with the method *getPrimalSolution ("h")*, and then convert it into a *double []* with the method *to1DArray*.

Check. Run the algorithm and check that the results do not change.

9.2 Perturbation function

In this section, we are interested in (i) observing how the optimum benefit changes if we change the capacity of the link 1-2, keeping the link 1-3 unchanged, and (ii) check that the optimum multiplier of constraint (2b) permits constructing an optimistic estimator of how the optimum benefit changes (the slope of the tangent line to the perturbation function).

Follow the next steps:

1. Run the optimization algorithm for the values:

$$U_{12} = \{0.5, 0.6, \ldots, 1.5\}$$

2. Draw a graph with the perturbation function: the points $(U_{12}, p^*(U_{12}))$, where $p^*(U_{12})$ is the optimum benefit obtained when the problem is solved with such U_{12} value.

3. In the same graph, draw a line that touches the point $(1, p^*(1))$, and has the slope given by minus the multiplier of the constraint. Observe that, as predicted by theory, the line is tangent to the perturbation function (which is concave since we are in a maximization problem), and thus becomes an optimistic estimator for it.

Quiz 2. Write the theoretical result that supports this observation.

10 Work at home after the lab work

The student is encouraged to complete all the *Quizs* that he/she could not finish during the lab session.