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 #6. Traffic routing. Flow-link formulations

(1 session)

Author:

Pablo Pavón Mariño

1 Objectives

The goals of this lab work are:

1. Create Net2Plan algorithms that solve several variants of the traffic routing problem, solving flow-link formulations using the *Java Optimization Modeler* (JOM) library.
2. Gain experience with the different forms of writing optimization problems in JOM, making benefit of its vectorial representation capabilities.
3. Explore the potential of the matricial representation of the routing constraints in JOM and Net2Plan.

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 Section 4.3, 4.6.3 and 4.6.4 of [1], and the lecture notes regarding flow-link formulations.
- Refresh your reading in the JOM documentation in `http://www.net2plan.com/jom`, in particular, how vector of variables and constraints are handled.

6 Minimum bandwidth routing

We are interested in creating Net2Plan algorithms that solves the minimum bandwidth routing problem using JOM. The problem is defined as follows:

- Input parameters (known constants):
  - $\mathcal{N}$: Set of network nodes.
  - $\mathcal{E}$: Set of network links. From this information, $\delta^+(n)$ denotes the set of links outgoing from node n, and $\delta^-(n)$ the set of incoming links to n.
  - $u_e, e \in \mathcal{E}$: Capacity of a link e.
  - $\mathcal{D}$: Set of offered unicast demands.
  - $h_d, d \in \mathcal{D}$: Offered traffic of a demand d.
- Decision variables:
  - $x_{de}, d \in \mathcal{D}, e \in \mathcal{E}$: Traffic of demand d that traverses link e.
- Formulation:

$$\min \quad \sum_{de} x_{de}, \quad \text{subject to:} \tag{1a}$$

$$\sum_{e \in \delta^+(n)} x_{de} - \sum_{e \in \delta^-(n)} x_{de} = \begin{cases} h_d, \text{ if } n = a(d) \\ -h_d, \text{ if } n = b(d) \\ 0, \text{ otherwise} \end{cases}, \quad \forall d \in \mathcal{D}, \forall n \in \mathcal{N} \tag{1b}$$

$$\sum_{d} x_{de} \leq u_e, \quad \forall e \in \mathcal{E} \tag{1c}$$

$$x_{de} \geq 0, \quad \forall d \in \mathcal{D}, e \in \mathcal{E} \tag{1d}$$

The objective function (1a) minimizes the total amount of traffic in the links. Constraints (1b) are the flow conservation constraints. Constraints (1c) mean that for each link, the traffic carried in the link is less or equal than its capacity (and thus, no link is oversubscribed). Finally, (1d) forbids that a link carries a negative amount of traffic of a demand, since this has no physical meaning.

7 Net2Plan algorithm

The student should develop a Net2Plan algorithm solving problem (1) following the next steps:

1. Copy the `AlgorithmTemplate.java` file in Aula Virtual and rename it as `FlowLinkUnicast.java`.
2. The algorithm has no input parameters.
3. This algorithm will work whatever is the routing type of the input design. We just want to remove any carried routing of the network (routes in source routing, or forwarding rules in hop-by-hop routing). We can do that calling to the method `removeAllUnicastRoutingInformation` of the input `NetPlan` object.
4. Create an object of the type `OptimizationProblem` (e.g. of name `op`).
5. Add the problem decision variables, with name `x_de`: one variable per demand and link. The minimum value of the variables is set to zero, the maximum to *Double.MAX_ VALUE*.
6. Set the problem objective function. Use the JOM function `sum`.

7. Use a double for loop with one iteration per network node, and then one inner iteration per demand, to add the constraints (1b). For adding the conservation constraint of a demand `d` and node `n`:

   - Set the JOM input parameters:
     - `deltaPlus` with the indexes of the output links of the current node. For this, use the method *getOutgoingLinks* of the node object to get the output links, and the method *NetPlan.getIndexes* to convert the collection of links to their indexes.
     - `deltaMinus` with the indexes of the incoming links of the current node. For this, use the method *getIncomingLinks* of the node object to get the incoming links, and the method *NetPlan.getIndexes* to convert the collection of links to their indexes.
     - `h_d` with the current demand offered traffic.
     - `d` with the current demand index.
   - Set the constraint using the function `sum`, over `x_de`, but restricting the sum to the elements in row `d` and the columns in `deltaPlus` or `deltaMinus`.

8. Use a for loop with one iteration per link, to add the constraints (1c). One constraint is added inside each iteration of the loop. For adding the constraint of a link `e`:

   - Set a JOM input parameter of name `u_e`, with a value equal to the current link capacity, and the input parameter `e` with the current link index.
   - Set the constraint using the function `sum`, over all the demands (using the JOM keyword `all` in the demand coordinate), and the link of index `e`.

9. Call the solver to find a numerical solution.

10. Retrieve the primal solution obtained, and convert it into a *DoubleMatrix2D* object using the method *view2D*. An object of the class *DoubleMatrix2D* contains a 2D matrix and permits making operations with it efficiently.

11. Use the method of *NetPlan* called *setRoutingFromDemandLinkCarriedTraffic*. This method automatically creates the routes in the network that are consistent to what appears in the x_{de} 2D matrix computed. Note that in the x_{de} values obtained, each (d, e) coordinate contains the amount of traffic of d in link e, and *not* the fraction of traffic respect to h_d. This is important when calling the method *setRoutingFromDemandLinkCarriedTraffic*. Since, this formulation cannot create loops (solutions with loops are never optimal, since they are strictly worse than the same routes without loops), set the option that automatically eliminates them to false.

7.1 Check the algorithm

Load the network `example7nodesWithTraffic.n2p`. The algorithm should produce a solution with an amount of consumed bandwidth in the links of 155.5.

If we reduce the capacity of the links to 15 units (instead of 50), the optimum solution will consume 174.1 units.

If we reduce the capacity of the links to 10 units, the problem will be unfeasible (there is not enough capacity in the links to carry the traffic in any form).

Quiz 1. Is the traffic going through the shortest paths for all demands in the first case ($u_e = 50$)? does it hold again for $u_e = 15$?

8 Problem variations

Quiz 2. Modify formulation (1) so that now the x_{de} variables are renamed as $\hat{x}_{de}$, and measure the **fraction** ($\in [0;1]$) of the traffic of demand d, that is carried through link e. That is, $x_{de} = h_d \hat{x}_{de}$.

- Decision variables:
  - $\hat{x}_{de}, d \in \mathcal{D}, e \in \mathcal{E}$: Fraction $\in [0,1]$ of the offered traffic of demand d that goes through link e
- Formulation:

$$\min \quad \sum_{de} h_d \hat{x}_{de}, \quad \text{subject to:} \tag{2a}$$

$$\sum_{e \in \delta^+(n)} \hat{x}_{de} - \sum_{e \in \delta^-(n)} \hat{x}_{de} = \begin{cases} 1, \text{ if } n = a(d) \\ -1, \text{ if } n = b(d) \\ 0, \text{ otherwise} \end{cases}, \quad \forall d \in \mathcal{D}, \forall n \in \mathcal{N} \tag{2b}$$

$$\sum_{d} h_d \hat{x}_{de} \leq u_e, \quad \forall e \in \mathcal{E} \tag{2c}$$

$$\hat{x}_{de} \geq 0, \quad \forall d \in \mathcal{D}, e \in \mathcal{E} \tag{2d}$$

Remake the algorithm with the new decision variables (changing their name in JOM to `xx_de`). For the multiplication in the objective function use the JOM expression:

```
sum (h * xx_de)
```

where:

- `h` is a row vector with as many coordinates as demands, containing the demand offered traffic. The offered traffic vector can be obtained in Net2Plan using the method of `NetPlan` class:

```
getVectorDemandOfferedTraffic
```

- Then, `h * xx_de` is a vector by matrix multiplication, that produces a vector with one coordinate per link, containing the link carried traffic.
- `sum (h * xx_de)` is then the sum in all the links of the carried traffic.

In its turn, for the link capacity constraints, use the expression `h * xx_de (all,e)` to obtain the traffic carried in link `e`. Note that this is the multiplication of vector `h` with one-coordinate per demand, with vector `xx_de(all,e)`, with also one coordeinate per-demand, containing the $\hat{x}_{de}$ values in link `e`.

Quiz 3. In formulation (2), note that if the `xx_de` decision variables are constrained to be integer (that is, or 0 or 1), then the routing is constrained to be non-bifurcated. Add an input parameter to the problem called `isNonBifurcated`, with default value `false`. Use this parameter to permit the user choosing if the routing is constrained or not to be bifurcated. Check that the example topology chosen in the tests has no solution when link capacities are set to 15 units, when the routing is constrained to be non-bifurcated (while there is a solution with bifurcated routing).

Quiz 4. Implement a Net2Plan algorithm that solves a flow-link formulation to find the routing that maximimizes the idle bandwidth in the bottleneck. Take the formulation from the lecture notes, or Section 4.3 of [1]:

$$\max \quad u, \quad \text{subject to:} \tag{3a}$$

$$\sum_{e \in \delta^+(n)} x_{de} - \sum_{e \in \delta^-(n)} x_{de} = \begin{cases} h_d, \text{ if } n = a(d) \\ -h_d, \text{ if } n = b(d) \\ 0, \text{ otherwise} \end{cases}, \quad \forall d \in \mathcal{D}, \forall n \in \mathcal{N} \tag{3b}$$

$$\sum_{d} x_{de} \leq u_e - u, \quad \forall e \in \mathcal{E} \tag{3c}$$

$$x_{de} \geq 0, \quad \forall d \in \mathcal{D}, e \in \mathcal{E} \tag{3d}$$

Check that the solution that maximizes the idle bandwidth in the bottleneck can be different to the one that minimizes the average consumed bandwidth.

9 Matricial form of problem constraints (optional)

9.1 Flow conservation constraints - fractional case

The flow conservation constraints in (2b) (using $\hat{x}_{de}$ variables) consist of one constraint per demand, and per network node ($|\mathcal{D}| \times |\mathcal{E}|$ constraints).

All the constraints can be represented by a single matricial equality as follows:

$$A_{ne} \times x'_{de} = A_{nd} \tag{4}$$

where:

- A_{ne} is the link incidence matrix. This is a $|\mathcal{N}| \times |\mathcal{E}|$ matrix with one row per node, and one column per link. Coordinate (n, e) is 1 if the link e is an outgoing link of n, -1 if it is an incoming link to n, and 0 otherwise.
  - The link incidence matrix can be obtained as an sparse matrix in Net2Plan using the method of `NetPlan` class:

    `getMatrixNodeLinkIncidence`

- x_{de} is a $|\mathcal{D}| \times |\mathcal{E}|$ matrix, and x'_{de} its transpose, with the decision variables.
- A_{nd} is the demand incidence matrix. This is a $|\mathcal{N}| \times |\mathcal{D}|$ matrix with one row per node, and one column per demand. Coordinate (n, d) is 1 if the demand d is an outgoing demand of n, -1 if it is an incoming demand to n, and 0 otherwise.
  - The demand incidence matrix can be obtained as an sparse matrix in Net2Plan using the method of `NetPlan` class:

    `getMatrixNodeDemandIncidence`

9.2 Link capacity constraints - fractional case

The link capacity constraints in (2) (using $\hat{x}_{de}$ variables) consists of one constraint per link ($|\mathcal{E}|$ constraints):

All the constraints can be represented by a single vectorial inequality as follows:

$$h \times x_{de} \leq u \tag{5}$$

where:

- h is a row vector $1 \times |\mathcal{D}|$ with the offered traffic of each demand.
  - The offered traffic vector can be obtained in Net2Plan using the method of `NetPlan` class:

    `getVectorDemandOfferedTraffic`

- u is a row vector $1 \times |\mathcal{E}|$ with the capacity of each link.
  - The link capacity vector can be obtained in Net2Plan using the method of `NetPlan` class:

    `getVectorLinkCapacity`

Quiz 5. Rewrite the flow conservation and link capacity constraints in Quiz 3 using their matricial form. Recall that JOM operator `*` implements the standard matrix multiplication.

Quiz 6. Rewrite the flow conservation and link capacity constraints in (1) (with x_{de} variables) using matricial expressions with JOM.

- For flow conservation constraints, use the expression:

  $$A_{ne} \times x'_{de} = A_{nd} \times \mathbf{diag}(h)$$

  where $\mathbf{diag}(h)$ is a diagonal matrix with the demand offered traffics in the diagonal.

- For link capacity constraints use the expression:

  $$\sum_d x_{de} \leq u$$

  and note that the JOM expression `sum (x_de , 1)`, sums along the rows of the matrix, and produces a $1 \times |\mathcal{E}|$ vector with the carried traffic in each link.

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.

Bibliography

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