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 #5. Traffic routing. Flow-path 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-path formulations of the problem 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 4.2 and 4.6.2 to 4.6.4 of [1], and the lecture notes regarding flow-path 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.
  - $u_e, e \in \mathcal{E}$: Capacity of a link e.
  - $\mathcal{D}$: Set of offered unicast demands.
  - $h_d, d \in \mathcal{D}$: Offered traffic of a demand d.
  - $\mathcal{P}_d, d \in \mathcal{D}$: For each demand, list of the admissible paths. In our case, the k loopless shortest paths between the demand end nodes, in number of hops. These are the paths accepted as potential routes to carry traffic of the demand.
  - $\mathcal{P}$: the set of all the admissible paths in the network ($\mathcal{P} = \bigcup_{d \in \mathcal{D}} \mathcal{P}_d$). Given a path $p \in \mathcal{P}$, we denote as $d(p)$ to the demand that p is associated to. We denote as l_p to the number of links traversed by path p. Given a link e, we denote as $\mathcal{P}_e$ to the set of admissible paths (i.e. in $\mathcal{P}$) traversing e.
- Decision variables:
  - $x_p, p \in \mathcal{P}$: Carried traffic by path p
- Formulation:

$$\min \quad \sum l_p x_p, \quad \text{subject to:} \tag{1a}$$

$$\sum_{d \in \mathcal{P}_d} x_p = h_d, \quad \forall d \in \mathcal{D} \tag{1b}$$

$$\sum_{e \in \mathcal{P}_e} x_p \leq u_e, \quad \forall e \in \mathcal{E} \tag{1c}$$

$$x_p \geq 0, \quad \forall p \in \mathcal{P} \tag{1d}$$

The objective function (1a) minimizes the total amount of traffic in the links: for each path, its carried traffic multiplied by the number of traversed links. Constraints (1b) mean that for each demand, all the traffic is carried (the sum of the traffic carried by the demand's route sums the demand's offered traffic). 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 path carries a negative amount of traffic, 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 `FlowPathUnicast.java`.
2. The algorithm should have one input parameter named `k`, with default value 10, and description *Maximum number of loopless admissible paths per demand.*
3. Set the routing type of the input `NetPlan` object to source routing.
4. Remove all the existing routes in the input design.

5. For each demand, compute the `k` loopless shortest paths between the demand end nodes (`k` is the input parameter), in number of hops. For this, use the function `GraphUtils.getKLooplessShortestPaths`. All these paths should be added to the input `NetPlan` design as routes with zero carried traffic. They are the admissible paths. Note than then:
   - Given a Net2Plan demand `d`, the method `d.getRoutes` returns the routes of the demand (and thus the $\mathcal{P}_d$ set).
   - Given a Net2Plan link `e`, the method `e.getTraversingRoutes` returns the routes traversing the link (and thus the $\mathcal{P}_e$ set).
6. Create an object of the type `OptimizationProblem` (e.g. of name `op`).
7. Add the problem decision variables, with name `x_p`: one variable per admissible path (and thus, one per route in the `NetPlan` object). The minimum value of the variables is set to zero, the maximum to *Double.MAX_ VALUE*.
8. Set the problem objective function. For this:
   - Set a JOM input parameter (using the method `setInputParameter`) of name `l_p`, containing a row vector with as many elements as admissible routes, and coordinate p containing the number of hops of the admissible route of index p. Use the method *netPlan.getVectorRouteNumberOfLinks()* to obtain automatically such vector.
   - Set the objective function using any of these two forms:
     - Use the `sum` function of JOM combined with the operator `.*` (element-by-element multiplication).
     - Use the operators `*` (matrix multiplication) and `'` (transpose), since the objective function equals the row vector l_p multiplied by the vector x_p in a column form.
9. Use a for loop with one iteration per demand, to add the constraints (1b). One constraint is added inside each iteration of the loop. For adding the constraint of a demand `d`:
   - Set a JOM input parameter of name `h_d`, with a value equal to the demand offered traffic.
   - Set a JOM input parameter of name `P_d`, with a vector array containing the indexes of the routes carrying traffic of the demand. Use the method *getRoutes* of the demand object for getting the routes, and the static method *NetPlan.getIndexes* to convert the collection of demand objects to a collection of their indexes.
   - Set the constraint using the function `sum`, over `x_p`, but restricting the sum to the elements in `P_d`.
10. 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 link capacity.
    - Set a JOM input parameter of name `P_e`, with a vector array containing the indexes of the routes traversing the link. Use the method *getTraversingRoutes* of the link object for getting the routes, and the static method *NetPlan.getIndexes* to convert the collection of demand objects to a collection of their indexes.
    - Set the constraint using the function `sum`, over `x_p`, but restricting the sum to the elements in `P_e`.
11. Call the solver to find a numerical solution.
12. Retrieve the primal solution obtained. Then, for each route in the network, set its carried traffic and occupied link capacity according to the optimum solution found.

13. Use the method `removeAllRoutesUnused` of the `NetPlan` object to remove those unused routes. We consider an unused route, the ones carrying less then 0.001 traffic units (routes with non-zero carried traffic, but with a traffic so small can appear because of numerical errors in the solver).

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 the admissible list computation, by considering as admissible the k loopless shortest paths for each demand, measure in total length (that is, summing the length in km of the traversed links). Note: This means passing to the *getKLooplessShortestPaths* method, a link cost map where the cost of each link equals to its length in km.

Quiz 3. Add an input parameter to the problem, so that the user can limit the maximum number of hops of any admissible path. Parameter name: `maxNumHops`, default `4`, description "Maximum number of hops of an admissible path".

Quiz 4. Modify formulation (1) so that now the x_p variables are renamed as $\hat{x}_p$, and measure the **fraction** ($\in [0; 1]$) of the traffic of demand $d(p)$, that is carried through path p.

- Decision variables:
  - $\hat{x}_p, p \in \mathcal{P}$: Fraction $\in [0, 1]$ of demand $d(p)$ carried traffic that goes through path p
- Formulation:

$$\min \quad \sum l_p h_p \hat{x}_p, \quad \text{subject to:} \tag{2a}$$

$$\sum_{d \in \mathcal{P}_d} \hat{x}_p = 1, \quad \forall d \in \mathcal{D} \tag{2b}$$

$$\sum_{e \in \mathcal{P}_e} h_p \hat{x}_p \leq u_e, \quad \forall e \in \mathcal{E} \tag{2c}$$

$$0 \leq \hat{x}_p \leq 1, \quad \forall p \in \mathcal{P} \tag{2d}$$

Remake the algorithm with the new decision variables (changing their name in JOM to `xx_p`). The constants h_p provide for each path p, the offered traffic of its associated demand $d(p)$. The vector h_p can be obtained in Net2Plan using the method of *NetPlan*:

`getVectorRouteOfferedTrafficOfAssociatedDemand`

Note that if the `xx_p` 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.

Quiz 5. Implement a Net2Plan algorithm that solves a flow-path formulation to find the routing that maximimizes the idle bandwidth in the bottleneck. Take the formulation from the lecture notes, or Section 4.2 of [1]:

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

$$\sum_{d \in \mathcal{P}_d} x_p = h_d, \quad \forall d \in \mathcal{D} \tag{3b}$$

$$\sum_{e \in \mathcal{P}_e} x_p \leq u_e - u, \quad \forall e \in \mathcal{E} \tag{3c}$$

$$x_p \geq 0, \quad \forall p \in \mathcal{P} \tag{3d}$$

9 Matricial form of problem constraints (optional)

9.1 Demand satisfaction constraints

The demand satisfaction constraints in (1) take the form:

$$\sum_{p \in \mathcal{P}_d} x_p = h_d, \forall d \in \mathcal{D}$$

All the $|\mathcal{D}|$ constraints can be represented by a single vectorial equality as follows:

$$A_{dp} \times x'_p = h' \tag{4}$$

where:

- A_{dp} is demand-to-route assignment matrix. This is a $|\mathcal{D}| \times |\mathcal{P}|$ matrix with one row per demand, and one column per admissible path. Coordinate (d, p) is 1 if the path p is an admissible path associated to demand d, and 0 otherwise.
  - The demand-to-route assignment matrix can be obtained as an sparse matrix in Net2Plan using the method of `NetPlan` class:

    `getMatrixDemand2RouteAssignment`

- x_p is a $1 \times |\mathcal{P}|$ row vector, with the decision variables (the traffic carried by path p).
- 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`

9.2 Link capacity constraints

The link capacity constraints in (1) take the form:

$$\sum_{p \in \mathcal{P}_e} x_p \leq u_e, \forall e \in \mathcal{E}$$

All the $|\mathcal{E}|$ constraints can be represented by a single vectorial inequality as follows:

$$A_{ep} \times x'_p \leq u' \tag{5}$$

where:

- A_{ep} is link-to-route assignment matrix. This is a $|\mathcal{E}| \times |\mathcal{P}|$ matrix with one row per link, and one column per admissible path. Coordinate (e, p) is the number of times that path p traverses link e.
  - The link-to-route assignment matrix can be obtained as an sparse matrix in Net2Plan using the method of `NetPlan` class:

    `getMatrixLink2RouteAssignment`

- x_p is a $1 \times |\mathcal{P}|$ row vector, with the decision variables (the traffic carried by path p).
- 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 6. Rewrite the demand satisfaction and link capacity constraints using their matricial form. Recall that JOM operator `*` implements the standard matrix multiplication.

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.*