TEORÍA DE REDES DE TELECOMUNICACIONES

Grado en Ingeniería Telemática
Grado en Ingeniería en Sistemas de Telecomunicación

Curso 2015-2016

Lab work #7. Modular capacities and routing optimization using a destination-link formulation

(1 session)

Author:

Pablo Pavón Mariño

1 Objectives

The goals of this lab work are:

1. Create Net2Plan algorithms that solve a CFA problem, where the link capacities and the routing are jointly optimized, solving a destination-link formulation 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 4.4 of [1], and the lecture notes regarding destiation-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 Joint routing and capacity allocation with modular capacities

Let $\mathcal{G}(\mathcal{N}, \mathcal{E})$ be a network, where the set of nodes $\mathcal{N}$ and the set of links $\mathcal{E}$ are given. The offered traffic is composed of a set of unicast demands $\mathcal{D}$, and the offered traffic h_d for each demand $d \in \mathcal{D}$ is known.

The network is based on equipment that forwards the traffic according to its destination node. The link capacities are restricted to be integer multiples of C traffic units.

We are interested in creating a Net2Plan algorithm that finds the destination-based routing and the link capacities that minimize the total network cost, computed as the number of modules of C capacity units to install in the network. The formulation to solve with JOM is defined below:

- Input parameters (known constants):
  - $\mathcal{N}$: Set of network nodes.
  - $\mathcal{E}$: Set of network links.
  - $\mathcal{D}$: Set of offered unicast demands.
  - $h_d, d \in \mathcal{D}$: Offered traffic of a demand d.
  - C: Traffic units of one capacity module.
- Decision variables:
  - $a_e, e \in \mathcal{E}$: Number of capacity modules to install in link e.
  - $x_{te}, t \in \mathcal{N}, e \in \mathcal{E}$: Amount of traffic targeted to destination node t, that traverses link e.
- Formulation:

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

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

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

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

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

The objective function (1a) minimizes the total number of capacity modules (our estimation of the network cost). Constraints (1b) are the standard flow conservation constraints for the destination-link formulation. 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, (1de) forbid buying a negative amount of capacity modules for a link, or carrying 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 `DestinationLinkModularCapacitie`
2. The algorithm should have one input parameter named `capacityModule`, with default value 10, and description *Size of the capacity module measured in the same units as the traffic.* This is the C constant in (1).

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. Compute the traffic matrix (with as many rows and columns as nodes) from the offered demands, using the method

   getMatrixNode2NodeOfferedTraffic

   of the the `NetPlan` class. Note that a `DoubleMatrix2D` object is returned. It can be converted into a `double [][]` using the *toArray* method of `DoubleMatrix2D`.

5. Create an object of the type `OptimizationProblem` (e.g. of name `op`).

6. Add the problem decision variables, with name `x_te`: one variable per node and per link. The minimum value of the variables is set to zero, the maximum to `Double.MAX_VALUE`.

7. Add the problem decision variables, with name `a_e`: one variable per link. The minimum value of the variables is set to zero, the maximum to `Integer.MAX_VALUE`, the variables are restricted to be integer.

8. Set the problem objective function using the JOM function `sum`.

9. Use two nested for loops to add the flow conservation constraints. Outer for loop iterates in the nodes of which we are making the conservation (n in (1b)), and the inner in the destination nodes (t in (1)b). For adding the conservation constraint of a node `n` and destination node `t`:

   - Set the JOM input parameters:
     - `deltaPlus` with the indexes of the output links of the current node `n`. 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 `n`. 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_nt` with the coordinate of the traffic matrix with the traffic from node `n` to node `t`.
     - `h_t` input ($h_t = \sum_{n'} h_{n't}$) can be easily generated using the method:

       `getVectorNodeEgressUnicastTraffic`

     of the `NetPlan` class, that returns the total offered traffic targeted to a given node.
   - Set the constraint using the function `sum`, over `x_te`, but restricting the sum to the elements in row `t` and the columns in `deltaPlus` or `deltaMinus`.

10. Add the JOM input parameter `C`, with the value given by the algorithm input parameter `capacityModule` (the C constant in (1)).

11. 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` use the function `sum`, over all the desintation nodes `t` (using the JOM keyword `all` in the destination coordinate), and the link of index `e`.

12. Call the solver to find a numerical solution. Use the option `maxSolverTimeInSeconds` to set the maximum solver time to 10 seconds. The solver will return the best solution found so far after 10 seconds, if an optimum solution was not found before. It may happen that the solver could not find any feasible solution. To check this situation use the method `solutionIsFeasible` of the `OptimizationProblem`, and raise an exception if a feasible solution was not found by the solver.

13. Retrieve the primal solution obtained, and convert it into a *DoubleMatrix2D* object using the method *view2D*.
14. Use the method of *NetPlan* called *setRoutingFromDestinationLinkCarriedTraffic*. This method automatically creates the routes in the network that are consistent to what appears in the x_{te} 2D matrix retrieved. Since the formulation can create optimal solutions with loops (loops in links with idle bandwidth, that do not cause the need of more capacity modules), set the option in this method that automatically eliminates them.
15. Set the capacities of all the links in the `netPlan` design as the number of modules for that link multiplied by the module capacity.

7.1 Check the algorithm

Load the network `example7nodesWithTraffic.n2p`. The algorithm should produce a solution with a total of 200 capacity units installed in the network (20 modules) when the capacity module size is 10.

Reducing the capacity module size to 1, the total optimal capacity installed becomes 158.

Quiz 1. Why the total capacity installed is lower when we reduce the capacity module size?

8 Problem variations

Quiz 2. Modify the problem in (1) so now the cost of each capacity module is proportional to the link length in km. Then, the new objective function becomes:

$$\min \quad \sum_e d_e a_e$$

where d_e is the link lenght as returned by the method `getLengthInKm()` of the link object.

9 Matricial form of problem constraints (optional)

9.1 Flow conservation constraints

The flow conservation constraints in (1b) consist of one constraint per node n and one constraint per destination node t ($|\mathcal{N}|^2$ constraints).

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

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

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_{te} is a $|\mathcal{N}| \times |\mathcal{E}|$ matrix, and x'_{te} its transpose, with the decision variables.
- H_{st} is the $|\mathcal{N}| \times |\mathcal{N}|$ traffic matrix, with a slight modification: the (n, n) coordinates in the diagonal contain the values $-\sum_{n'} h_{n'n}$, so the sum of each column of H_{st} is zero.

9.2 Link capacity constraints - fractional case

The link capacity constraints in (1c) consists of one constraint per link ($|\mathcal{E}|$ constraints):

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

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

where:

- $\sum_t x_{te}$ becomes now a $1 \times |\mathcal{E}|$ vector, with the carried traffic of each link (use the JOM command `sum` over the first dimension of `x_te` for that).
- a represents the $1 \times |\mathcal{E}|$ vector with the number of modules to install per link.

Quiz 3. Rewrite the flow conservation and link capacity constraints using their matricial form.

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.*