Integer Linear Programming Models for Optical Transport Networks Dimensioning (2/2)

Redes Óticas/Optical Networks 2016/2017

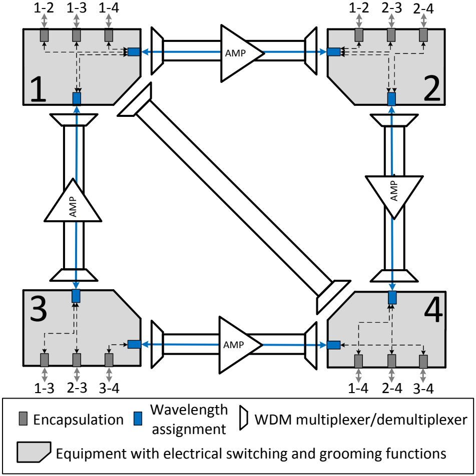
Departamento de Eletrónica Telecomunicações e Informática, Universidade de Aveiro

Rui Manuel Morais
e-mail:rui.m.morais@coriant.com

Contents

Opaque Networks

- In **opaque networks**, an **OEO conversion** of the signals is performed at the end of each transmission system, thus a **link-by-link grooming** scheme is employed.
- In the link-by-link grooming scheme **every client signal can be groomed with any other that share the same transmission system**.



Opaque Networks Dimensioning - Notation

- **Indexes:**
  - Origin node, o, and destination node, d, of a demand.
  - Origin node, i, and destination node, j, of a link (transmission system).
  - Client traffic type, c.
- **Inputs:**
  - Network topology in the form of an adjacency matrix $[G_{ij}]$.
  - Client traffic demands in the form of a 3-dimensional matrix $[D_{odc}]$.
- **Variables:**
  - f_{ij}^{od} - Binary variable indicating if link (i, j) is used in the path between nodes (o, d).
  - W_{ij} - Integer variable indicating the number of 100 Gbit/s optical channels between the nodes i and j.

Opaque Networks Dimensioning - ILP

$$min \quad \sum_{(i,j)} \sum_{(o,d)} f_{ij}^{od} + \sum_{(i,j)} W_{ij} \tag{1}$$

subject to

$$\sum_{j \setminus \{o\}} f_{ij}^{od} = 1 \qquad \forall (o,d) : o < d, \forall i : i = o \tag{2}$$

$$\sum_{j \setminus \{o\}} f_{ij}^{od} = \sum_{j \setminus \{d\}} f_{ji}^{od} \qquad \forall (o,d) : o < d, \forall i : i \neq o, d \tag{3}$$

$$\sum_{j \setminus \{d\}} f_{ji}^{od} = 1 \qquad \forall (o,d) : o < d, \forall i : i = d \tag{4}$$

$$\sum_{(o,d):o<d} (f_{ij}^{od} + f_{ji}^{od}) \sum_{c \in C} B(c) D_{odc} \leq 100 W_{ij} G_{ij} \qquad \forall (i,j) : i < j \tag{5}$$

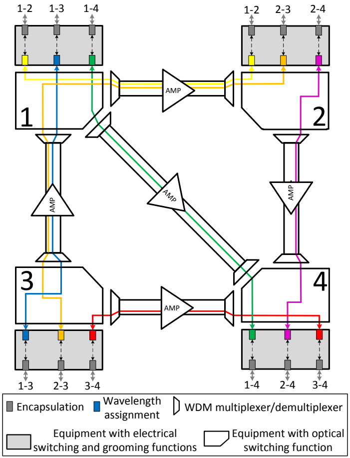
$$W_{ij} \leq 80 \qquad \forall (i,j) : i < j \tag{6}$$

$$f_{ij}^{od}, f_{ji}^{od} \in \{0,1\} \qquad \forall (i,j) : i < j, \forall (o,d) : o < d \tag{7}$$

$$W_{ij} \in \mathbb{N} \qquad \forall (i,j) : i < j \tag{8}$$

Transparent Networks

- In **transparent networks**, the signal is kept in the optical domain at every intermediate node of the path. Since **OEO conversion is only present** at the source and destination nodes, a **single-hop grooming** is employed.
- In the single-hop grooming scheme **only client signals with the same source and destination can be groomed into the same optical channel**.



Transparent Networks Dimensioning - Notation

- **Indexes:**
  - Origin node, o, and destination node, d, of a demand.
  - Origin node, i, and destination node, j, of a link (transmission system).
  - Client traffic type, c.
- **Inputs:**
  - Network topology in the form of an adjacency matrix $[G_{ij}]$.
  - Client traffic demands in the form of a 3-dimensional matrix $[D_{odc}]$.
- **Variables:**
  - f_{ij}^{od} - **Integer** variable indicating the **number of 100 Gbit/s optical channels** between the nodes o and d that uses link (i, j).
  - $W_{\mathbf{od}}$ - Integer variable indicating the number of 100 Gbit/s optical channels between the nodes **o** and **d**.

$$min \quad \sum_{(i,j)} \sum_{(o,d)} f_{ij}^{od} + \sum_{(o,d)} W_{od} \tag{9}$$

subject to

$$100 W_{od} \geq \sum_{c \in C} B(c) D_{odc} \qquad \forall (o,d) : o < d \tag{10}$$

$$\sum_{j \setminus \{o\}} f_{ij}^{od} = W_{od} \qquad \forall (o,d) : o < d, \forall i : i = o \tag{11}$$

$$\sum_{j \setminus \{o\}} f_{ij}^{od} = \sum_{j \setminus \{d\}} f_{ji}^{od} \qquad \forall (o,d) : o < d, \forall i : i \neq o, d \tag{12}$$

$$\sum_{j \setminus \{d\}} f_{ji}^{od} = W_{od} \qquad \forall (o,d) : o < d, \forall i : i = d \tag{13}$$

$$\sum_{(o,d):o<d} (f_{ij}^{od} + f_{ji}^{od}) \leq 80 G_{ij} \qquad \forall (i,j) : i < j \tag{14}$$

$$f_{ij}^{od}, f_{ji}^{od} \in \mathbb{N} \qquad \forall (i,j) : i < j, \forall (o,d) : o < d \tag{15}$$

$$W_{od} \in \mathbb{N} \qquad \forall (o,d) : o < d \tag{16}$$

MATLAB+LP Solve - Installing

- Help: `http://web.mit.edu/lpsolve/doc/MATLAB.htm`

- LP Solve MATLAB Extensions:
  `http://sourceforge.net/projects/lpsolve/files/lpsolve/5.5.2.0/lp_solve_5.5.2.0_MATLAB_exe_win64.zip/download`
- The files **mxlpsolve.mexw64** and **mxlpsolve.m** should be included in the **same folder as the .m**

- Library: `http://sourceforge.net/projects/lpsolve/files/lpsolve/5.5.2.0/lp_solve_5.5.2.0_dev_win64.zip/download`
- **Must be included in the Windows PATH environment**.

MATLAB+LP Solve - Syntax

- **Create** and initialize **a new model** with X variables:

```
p=mxlpsolve('make_lp', 0, X);
```

- By default is a minimization model. To change to **maximization**:

```
xlpsolve('set_sense', lp, 1);
```

- **Set** the coefficients c of the **objective function**:

```
xlpsolve('set_obj_fn', lp, c);
```

- **Add a $\leq$ constraint** row with coefficients a and independent term b to the lp (1 for $\leq$; 2 for $\geq$; 3 for $=$):

```
xlpsolve('add_constraint', lp, a, 1, b);
```

- **Set** the type of the variable j **to integer**:

```
xlpsolve('set_int', lp, j, 1);
```

- **Set** the type of the variable j **to binary**:

```
xlpsolve('set_binary', lp, j, 1);
```

MATLAB+LP Solve - Syntax

- **Write** an lp **model** to a file:

```
xlpsolve('write_lp', lp, 'model_name.lp');
```

- **Solve the model**:

```
xlpsolve('solve', lp);
```

- **Returns** the value of the **objective function**:

```
xlpsolve('get_objective', lp);
```

- **Returns** the values of **the variables**:

```
xlpsolve('get_variables', lp);
```

MATLAB+LP Solve - Example

- Let's consider our first simple example:

$$\begin{aligned} max \quad & z = 120x_1 + 160x_2 \\ s.t. \quad & \\ & x_1 + 1,5x_2 \leq 150 \\ & 4x_1 + 3x_2 \leq 360 \\ & x_1, x_2 \in \mathbb{N} \end{aligned}$$

- In MATLAB it comes:

```
lp=mxlpsolve('make_lp', 0, 2);
mxlpsolve('set_sense', lp, 1);
mxlpsolve('set_obj_fn', lp, [120, 160]);
mxlpsolve('add_constraint', lp, [1, 1.5], 1, 150);
mxlpsolve('add_constraint', lp, [4, 3], 1, 360);
mxlpsolve('set_int', lp, 1, 1);
mxlpsolve('set_int', lp, 2, 1);
mxlpsolve('write_lp', lp, 'Example.lp');
mxlpsolve('solve', lp);
obj = mxlpsolve('get_objective', lp);
var = mxlpsolve('get_variables', lp);
```

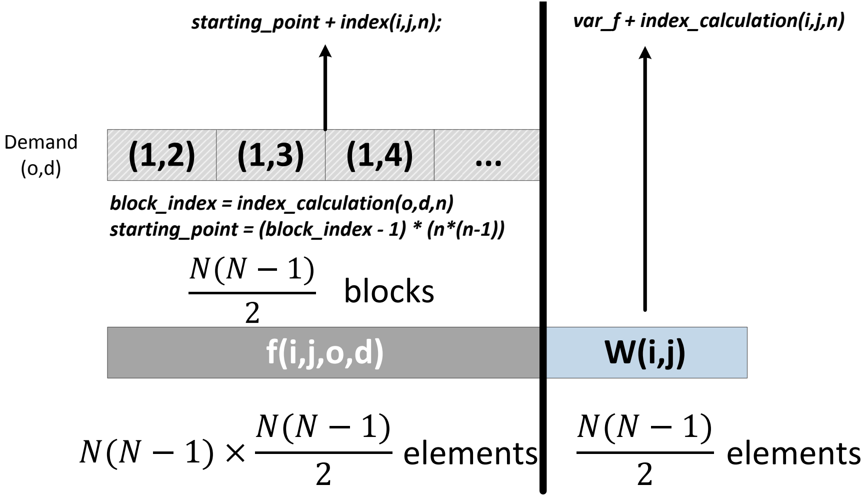
- Function that calculates the index of the element (i,j) of the upper half triangular matrix in a 1D representation:

$$index = \frac{(i-1)(2N-(i-1)-1)}{2} + ((j-1)-(i-1)-1)+1. \tag{17}$$

- Function that calculates the index of the element (i,j) of a matrix without diagonal in a 1D representation:

$$index = ((i-1)(n-1)+(j-1))+1. \tag{18}$$

MATLAB+LP Solve - Variable Index Search



What you need to do...

1. Determine the CapEx per node and link making use of the value of the variables f_{ij}^{od} and W_{od} obtained by solving the ILP model.

2. Compare the results with the ones obtained using the Net2Plan and the analytical formulation.