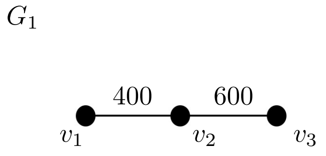
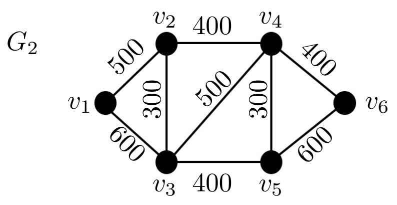
JOINT ROUTING AND WAVELENGTH ASSIGNMENT FOR SINGLE RATE AND MULTIRATE OPTICAL NETWORKS CERI MASTER I – OPTICAL NETWORKS – 2018/2019

F. DE PELLEGRINI

The joint routing and wavelength assignment problem (RWA) it is a core problem in resource optimization for Optical networks. In this TP we shall study numerically some heuristics to perform resources allocation for RWA problems.

There are three reference optical network topologies you should use. The first two networks are described by weighted graph and the traffic demand matrix in Fig. 1. The third topology is the one reported in Fig 2.





$$\Lambda_1 = \begin{pmatrix} 0 & 30 & 50 \\ 30 & 0 & 50 \\ 30 & 50 & 0 \end{pmatrix}$$

$$\Lambda_1 = \begin{pmatrix} 0 & 100 & 50 & 200 & 150 & 100 \\ 100 & 0 & 50 & 200 & 150 & 100 \\ 100 & 50 & 0 & 200 & 150 & 100 \\ 100 & 50 & 200 & 0 & 150 & 100 \\ 100 & 50 & 200 & 150 & 0 & 100 \\ 100 & 50 & 200 & 150 & 100 & 0 \end{pmatrix}$$

FIGURE 1. Two test fiber topologies and the respective traffic matrices.

Task 1: routing and coloring algorithms [5 pts]. Preliminarily, you need two algorithms which work on a weighted graph $G = (V, E)$. The optical network is represented by a graph. The graph is a weighed graph, where each edge $e \in E$ has a weight $d(e)$ which represents the length of the optical connection. Hence, a path $p = e_1 e_2 \dots e_K$ formed by K consecutive links has lenght $d(p) = \sum_{i=1}^{K} d(e_i)$.

(1) **Shortest path routing algorithm:** in order to perform routing, you shall use a shortest path approach. You can use the Djikstra algorithm: it is a standard algorithm: for your commodity the pseudocode is attached in the Appendix of this document. The Dijkstra algorithm returns as output a shortest path tree and you can use that as the reference lightpath tree sourcing at the root node. Test it on the sample topologies in Fig. 1.

(2) **Greedy coloring algorithm:** in order to perform wavelength assignment, you should use a graph coloring algorithm. A standard algorithm to this respect is the greedy algorithm we have been studying in class. Test it on the sample topologies in Fig. 1.

Provide proof of correctness: before moving on, you should demonstrate that the algorithms you have been implementing are working correctly. Be critic!

TABLE 1. Multirate problem parameters

Line	Rate [Gbit/s]	Coverage [Km]	Transponder Cost [Units]
R1	10	1750	1
R2	40	1800	2.5
R3	100	900	3.5

Task 2. RWA with single rates [5 pts]. The fiber topology is modeled as a graph. In this task, only one rate is provided, and you assume that each lightpath is able to satisfy the traffic demand for each source-destination pair. Hence, you should write a program able to do the following tasks: this will let you obtain a heuristic providing a feasible (yet likely not optimal) RWA solution.

- for every pair of source destination pairs (s, d) determine a shortest path route which will host the lightpath;
- construct the interference graph $P(G)$: two lightpaths are neighbors in $P(G)$ when they share a link;
- assign the wavelengths using the greedy coloring algorithm over $P(G)$;
- calculate, for every link, the capacity provided by the system and the throughput sustained.

You have to perform this task on graphs $G1$ and graph $G2$ depicted in Fig. 1; again in this step you can ignore the traffic matrix.

Provide proof of correctness: for graph $G1$, you should be able to compare the procedure performed by your program with the calculation you can do by hand.

Task 3. RWA-MLR [5 pts]. In this task you will deal with the multirate case, where the system parameters are described in Tab. 1. Perform the following steps (they are similar to the steps of Task 1, but this time you should refer also to the demand matrices reported in Fig. 1):

- The first step requires to consider again every source-destination path you have calculated in the first task: you need for each path to to determine the optimal rate allocation in order to minimize the cost of transponders while guaranteeing coverage;
- construct the lightpath graph $P(G)$: this time, you need to account for multiple lightpaths sourcing at each node;
- assign the wavelengths to the lightpaths using the greedy coloring algorithm over $P(G)$;
- calculate, for every link, the capacity provided by the system and the throughput sustained.

Provide proof of correctness: for graph $G1$, you should be able to compare the procedure performed by your program with the calculation you can do by hand.

Task 3. RWA-MLR on a real topology [5 pts]. If you have time, in this task you can pursue the same work done for in Task 2, but you should work on the topology $G3$. You can generate a traffic matrix for all pairs of source destinations with entries drawn from a uniform distribution in $[0, 200]$ Gbit/s.
Note: in this case, several source-destination pairs cannot be covered using a single lightpath: propose a way to overcome the reachability problem and implement your solution.

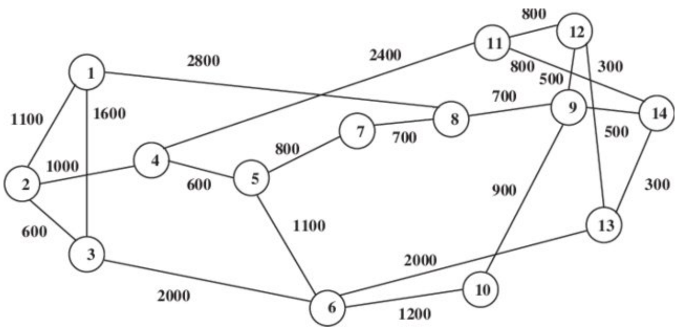


FIGURE 2. The NSF topology.

APPENDIX: PSEUDOCODE OF THE ALGORITHMS.

0.1. **Dijkstra.**

: the pseudocode for the construction of a SPT according to the Dijkstra algorithm is reported below:

```
Dijkstra($G$,$s$)
 1  for each vertex $v \in V_G$              // Initialize all nodes
 2       $d[v] = \infty$                     // Source at distance $\infty$
 3       $p[v] =$ NIL                        // Nodes are isolated
 4  $d[s] = 0$                              // Source at distance 0 from itself
 5  $Q = V_G$                               // $Q$ is a sorted Queue: sort nodes according
 6                                          // to the distance from the source
 7  while ($Q \neq \emptyset$)               // Untill all nodes visited
 8       $u =$ EXTRACT-MIN$Q$                 // Estract the node at minimum distance
 9       for each edge $e = (u, v)$          // Check all its neighbours
10            if $d[v] > d[u] + w[e]$        // Check if you can improve $d(s, v)$ via $u$
11                 $d[v] = d[u] + w[e]$      // Update $d(s, v)$
12                 $p[v] = u$                // $u$ becomes parent of $v$
13  $T = (V_G, \emptyset)$                   // Build $T$ from $p[]$
14  for each vertex $v \in V_G,\ v \neq s$    // for each node in the graph
15       $E_T = E_T \cup \{(p[v], v)\}$       // insert edge to parent
16
17  return $T = (V, E_T), p, d$              // return tree and distance
```

Note that the parent vector p completely specifies the tree.

0.2. **Greedy Coloring.**

: the pseudocode for the greedy coloring algorithm is listed below. The set $N_g(i)$ denotes all neighbors of node i.

```
GreedyColor($G$)
 1  label nodes $V = \{1, 2, \dots, N\}$, $N = |V|$      // an arbitrary labelling
 2  $K = 0$                              // the number of colors, null at the beginning
 3  for i=1, . . . , N                   // visit all nodes
 4       // assign the smallest color not used by neighbors:
 5       let $c(i)$ be the smallest positive integer such that
 6       $c(i) \notin \min\{c(j) : j < i, j \in N_g(i)\}$
 7  if $K < c(i)$                 // if the colors have been all used up to $K$
 8       $K = c(i)$               // increase $K$ of one unit
 9
10  return c, $K$
```

Note: any labelling at step 1 works, but typically a good heuristic for a greedy coloring is to label nodes for increasing degrees.

The Greedy coloring algorithm and the Dijkstra algorithms are classical algorithms. For a reference to Djikstra algorithm, look into: *Cormen, Thomas H., Charles E. Leiserson and Ronald L. Rivest, Introduction to Algorithms, MIT Press and McGraw-Hill.*