Design of Multi-layer networks with traffic grooming and statistical multiplexing[*]

P. Belotti[1]**, A. Capone**[2]**, G. Carello**[2]**, F. Malucelli**[2]**, F. Senaldi**[2]**, A. Totaro**[2]

[1]*Carnegie Mellon University, Pittsburgh PA,*

[2]*Politecnico di Milano, Italy*

Keywords: Network design, MPLS, statistical multiplexing, traffic grooming.

1 Introduction

Consider the following problem: given a Telecommunication network with a set of point-to-point traffic demands, an installation of network capacity is sought that minimizes the total network cost while routing all demands on single paths [4]. Moreover, the volume of each traffic demand is constant during most of the network operations, but is subject to rare peaks which are assumed non-simultaneous. Today's Multi-Protocol Label Switching (MPLS) networks are an application of this network model as MPLS nodes exploit the *statistical multiplexing* property, where demands that have sudden variations in the volume requested can be accommodated in the network capacity [3]. This key feature allows us to save large amounts of network capacity by "aggregating" traffic flow and thus sharing resources allocated to different traffic demands.

More specifically, the data exchanged between two MPLS nodes consists in the nominal amount of traffic for all demands plus, possibly, a peak of one of the demands. Multiplexed data transmitted by an MPLS node can only be de-multiplexed by another MPLS node. Hence, although a network does not need to have MPLS nodes only, an MPLS path accommodating a set of traffic demands must have two MPLS nodes as its endnodes. As MPLS nodes are quite expensive, one has to find the optimal tradeoff between the additional cost of MPLS nodes and the savings in the link bandwidth allocation due to the *statistical multiplexing* effect. If link capacities can be selected according to a small discrete set of values, MPLS traffic aggregation can also improve capacity utilization further improving cost savings (grooming effect).

In Section 2 we describe a traffic model for MPLS networks, and provide a path-based, Mixed Integer Linear Programming (MILP) formulation for the problem of designing a two-layer MPLS network where the total cost depends on the number of MPLS nodes and the link capacities. We apply Lagrangian Relaxation and the Subgradient method to obtain a lower bound of the network cost. As the number of path variables used to model the routing grows exponentially with the graph size, we use an initially limited number of variables and a column generation approach. We have also devised a heuristic approach, outlined in Section 4, to get a good feasible solution. Computational results are reported in Section 5 for real-world instances.

2 A model for network design with statistical multiplexing

We do not provide here, for reasons of space, a detailed description of the traffic model underlying our method. The interested reader can refer to [1]. We have adopted a traffic model where each demand q is described by an average rate f'_q and an additional rate f''_q: throughout network operations, the rate of demand q is f'_q, and it occasionally bursts at $f'_q + f''_q$. Considering a set of demands $\{(f'_1, f''_1), (f'_2, f''_2) \ldots, (f'_r, f''_r)\}$ routed on a link of the network, statistical multiplexing affects the capacity required by the aggregated flow. If no MPLS node was installed, none of the flows would be aggregated on a link, therefore the amount of the capacity to be installed would be:

$$\sum_{q=1}^{r} (f'_q + f''_q). \tag{1}$$

On the other hand, if the flows are aggregated on a single link, we can take advantage of statistical multiplexing and the capacity requirement on the link decreases. Only one demand is assumed to take on its highest

[*]This work has been partially supported by *ALCATEL Italia* and by the European Network of Excellence EURO-NGI.

value $f'_q + f''_q$. Therefore, the capacity needed on a link is:

$$\sum_{q=1}^{r} f'_q + \max_{q=1..r} f''_q. \tag{2}$$

As we point out above, the *aggregation* of traffic flows onto network arcs, as given in (2), is performed by MPLS nodes, and the inverse process, i.e. decompression, also needs a MPLS node. However, aggregated flows can be routed through non-MPLS nodes, which maintain the encapsulation and forward the aggregated traffic to other nodes. As a result, some traffic is routed without taking advantage of statistical multiplexing, and some is compressed and routed between MPLS nodes in order to share capacity resources. How can we model both forms of traffic to be routed on the same network? By introducing a paradigm that is specifically designed to separate distinct routing protocols in a network, i.e., a *multi-layer* model: a network is seen as a superposition of two identical networks that work with different protocols and where the corresponding nodes are coupled to provide a flexible way of switching data between the two layers[5].

In our case, one layer routes the non-aggregated traffic and allocates capacity on edges according to (1), while a second layer accommodates the capacity for aggregated traffic according to (2). However, in the latter – which we call the MPLS layer – traffic demands are only routed between MPLS nodes, which only constitute a subset of the network nodes. Therefore, further notation is needed to describe this limit in the routing of aggregated traffic: we define a set of *subpaths* between every node pair and require that traffic can only be aggregated if sent through subpaths defined between pairs of MPLS nodes. A subpath $s(i, j)$ or simply s (subpaths are also called *logical links* or *hops* in the Telecommunication literature) between two MPLS nodes i and j carries an aggregated flow B through edges of the network, thus it must be allocated B units of capacity on all edges contained in s. Let us denote with S the set of all subpaths.

The network is represented as an undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. Given a set Q of traffic demands, each traffic demand $q \in Q$ is defined by an origin s_q, a destination t_q (with $s_q, t_q \in V$), a mean flow f'_q (in Mbit/s) and an additional one f''_q (in Mbit/s). Let us also denote with F the maximum f''_q, i.e. $F = \max_{q\in Q} f''_q$. We consider an *augmented* set of edges, where both edges of E and subpaths are comprised. Let us denote P as the set of all the paths between pairs of nodes of V which may also contain subpaths. Given a traffic demand q, P_q denotes the set of paths connecting s_q and t_q. Each origin-destination path is composed by a sequence of hops. For each node $v \in V$, $I(v)$ and $F(v)$ are the sets of hops that start and end in v, respectively. Similarly, we denote with $\overline{I}(v)$ and $\overline{F}(v)$ the sets of arcs incident in v. An MPLS node is assumed to have a fixed cost θ, while each unit of capacity installed on edge e has a cost of c_e. The cost of a subpath s is given by the cost of all edges contained in s, i.e., $c_s = \sum_{e\in s} c_e$. Finally, let us define a per-channel capacity λ and an upper bound $\psi >> 0$ on the number of subpaths that origin or end in i.

We define a binary variable x_p for each path $p \in P$ connecting node s_q to t_q, which is 1 if traffic demand q is routed on path p, and zero otherwise. For each $v \in V$, a binary variable m_v tells us whether v is an MPLS node, and an integer variable y_e corresponds to the total capacity installed on edge e. For each $s \in S$, variables z_s represent the maximum peak rate among the demands routed on s. Using the variables described above, the model of the problem can be formulated as follows:

$$\min \quad \theta \sum_{v\in V} m_v + \sum_{e\in E} c_e y_e + \sum_{s\in S} c_s y_s \tag{3}$$

$$\sum_{p\in P_q} x_p \geq 1 \quad \forall q \in Q \tag{4}$$

$$\sum_{s\in S\cap(I(v)\cup F(v))} y_s \leq \psi m_v \quad \forall v \in V \tag{5}$$

$$\sum_{q\in Q}\sum_{p\in P_q:s\in p} f'_q x_p + z_s \quad\leq\quad \lambda y_s \quad \forall s\in S \tag{6}$$

$$\sum_{q\in Q}\sum_{p\in P_q:e\in p} (f'_q + f''_q) x_p \quad\leq\quad \lambda y_e \quad \forall e\in E \tag{7}$$

$$f''_q \sum_{p\in P_q:s\in p} x_p \quad\leq\quad z_s \quad \forall q\in Q, \forall s\in S \tag{8}$$

$$x_p, m_i \in \{0,1\}, y_s, y_e \in \mathbb{Z}^+, z_s \geq 0 \tag{9}$$

The objective function (3) is the sum of edge- and subpath capacity costs and node installation costs. Constraint (4) imposes that each traffic demand is routed on exactly one path. Constraints (5) guarantee that i is an MPLS if at least one subpath originating or ending in i is used. Constraints (6) and (7) define the relationship between routing path x_p and the capacity to be installed on all edges and subpaths. For the latter, the capacity is given by the sum of the mean values of the flows routed on it and the maximum of the peak rates, which is defined by constraint (8). Constraints (9) specify the type of all variables included.

Two remarks are due here. First, the model can be completed with multifacility features both on nodes and edges. This extension is particularly important in the case of practical applications but does not change the main aspects of the problem. We omit this part of the model for the sake of readability, and to better focus on the relevant point of this problem, i.e., statistical multiplexing. Second, the formulation above has a potentially huge number of variables, as the cardinality of the sets S and P of subpaths and paths is exponential in the number of nodes of G. Therefore, for practical purposes we instead consider initially limited sets S_0 and P_0 and apply column generation in order to include those variables x, y, and z with negative reduced cost.

3 Computing a lower bound through Lagrangian Relaxation

We apply Lagrangian Relaxation to constraints (6), (7), and (8). Model (3) - (9) is thus decomposed into $|Q|+1$ subproblems which are easier to solve and which give us a lower bound to the optimal solution to our problem.

Let us denote as π_s, σ_e, and ρ_s^q the Lagrangian multipliers associated with the relaxed constraints. The objective function of the Lagrangian dual is:

$$L(\boldsymbol{\pi},\boldsymbol{\sigma},\boldsymbol{\rho}) = L_0(\boldsymbol{\pi},\boldsymbol{\sigma},\boldsymbol{\rho}) + \sum_{q\in Q} L_q(\boldsymbol{\pi},\boldsymbol{\sigma},\boldsymbol{\rho}). \tag{10}$$

The first sub-problem minimizes the following objective function:

$$L_0(\boldsymbol{\pi},\boldsymbol{\sigma},\boldsymbol{\rho}) = \min \quad \sum_{e\in E}(c_e - \lambda\sigma_e)y_e + \sum_{s\in S}(c_s - \lambda\pi_s)y_s + \theta\sum_{i\in V} m_i + \sum_{s\in S}(\pi_s - \sum_{q\in Q}\sigma_s^q)z_s \tag{11}$$

subject to constraints (5) only. In this sub-problem, the z variables are not related with the y's or the m's, hence they can be set to zero or to their maximum value F depending on their coefficient $\pi_s - \sum_{q\in Q}\sigma_s^q$. Due to constraint (5), the problem needs to be solved by a Linear Programming (LP) solver.

Each sub-problem $L_q(\boldsymbol{\pi},\boldsymbol{\sigma},\boldsymbol{\rho}), q\in Q$, has the following form:

$$L_q(\boldsymbol{\pi},\boldsymbol{\sigma},\boldsymbol{\rho}) = \min \quad \sum_{s\in S}\sum_{p\in P_q:s\in p}(\pi_s f'_q + \rho_s^q f''_q)x_p + \sum_{e\in E}\sigma_e \sum_{p\in P_q:e\in p}(f'_q + f''_q)x_p \tag{12}$$

subject to constraint (4) and to $0 \leq x_p \leq 1$. These $|Q|$ sub-problems are shortest path problems on an oriented graph where the set of the arcs is the set $S\cup E$ and each arc is assigned a cost given by $\sigma_e(f'_q + f''_q)$ for all $e\in E$ and $(\pi_s f'_q + \rho_s^q f''_q)$ for all $s\in S$. Solving each of the $|Q|$ sub-problems with any shortest-path algorithm will give an integer solution even though integrality of x variables is not explicitly imposed.

4 A heuristic upper bounding procedure

To find a good feasible solution, we propose a heuristic algorithm based on a local search framework. The local search is applied to two initial solutions, leading to two versions of the algorithm. The first initial solution is obtained from the routing provided by the $|Q|$ shortest path problems at each subgradient iteration. According to such routing, resources are allocated on nodes and links. The local search is then applied to the set of K best feasible solutions obtained from such procedure. The second procedure builds the initial solution with a greedy algorithm that routes each commodity on the path with the minimum number of hops. Only those hops are considered that cross at most two links of the lower layer. Obviously, the set of potential LSPs available for the local search is different depending on the chosen initial solution.

Our local search is based on two neighborhoods. The local search alternately visits the two neighborhoods, swapping from one to the other when a local minimum is reached. Two key features of the problem solution are the routing of each traffic demand and the routing of each hop on the lower layer network. The two neighborhoods are generated by modifying such features of a given solution.

The first neighborhood is generated by routing each traffic demand on all possible LSPs connecting its source to its destination. As the number of possible LSPs connecting the origin and the destination node of a given traffic demand is huge, visiting the whole neighborhood would be expensive from a computational viewpoint. Therefore, we propose a heuristic way to visit such neighborhood based on the approach proposed in [2]. Instead of enumerating all possible alternative routing, we look for the routing providing the best improvement of the objective function and search heuristically for such routing, exploiting an auxiliary graph, built for each traffic demand. In the auxiliary graph, a cost is associated with each arc and represents the incremental cost, positive (or negative), of routing (removing) the considered traffic demand on (from) the arc itself. An improving move is then associated with a negative cost cycle. Thus looking for the best neighborhood turns out to be a minimum negative cost cycle problem, which is an NP-hard problem that can be solved heuristically as proposed in [2]. However, due to the technological feature of the problem, computing the auxiliary graph cost is computationally expensive as it requires to evaluate the link capacity to add or remove at the lower layer and the node capacity to add or remove at both layers.

The second neighborhood is built by changing the routing of all the traffic flows routed on the considered hop. The dimension of the second neighborhood is huge as well, thus we use a heuristic approach to find the best move. We generate an auxiliary graph with incremental arc costs using the same criteria adopted to explore the first neighborhood. Then we delete the only arc with a negative incremental cost (that represents the considered hop) and search the shortest path to connect the endpoints of the considered hop. If the length of this path is less than the cost of the deleted arc, considering the absolute value, a potentially improving solution is found.

5 Computational results

We have implemented our algorithms with MATLAB 7.0 and run them on an a workstation with a 1.8 Ghz processor and 1 Gbyte of RAM.

We have tested the formulation on real-world instances provided by ALCATEL Italia, with 25 (A instances) and 39 nodes (B instances) and 63 and 86 links, respectively. Two traffic demands, with 270 (60 of which have null f'') and 300 commodities, (40 with null f'') are given. We have solved different instances obtained by considering different ratios between mean and additional values: *low peak* instances, in which the ratio between peak and mean rate is [0.1-0.2], *medium peak* instances, in which the ratio is between [0.3-0.4] and *high peak* instances, with ratio ranging in [0.5-0.6]. In these three cases, for each traffic demand, the sum of mean and peak rates is the same.

To evaluate the effect of MPLS node costs, we have considered also two networks with 35 nodes (C and D instances) and one with 45 (E instances). These three instances have respectively 63, 106 and 102 edges, and two traffic matrices with 340 commodities (40 of which have null f'') and 575 commodities (75 with null f''). We have considered three nodes costs: one with the overall nodes cost equal to the overall edge cost

Network	Peak rate	LB	LH	LS_{LH}	Gap	Gain
		10^6 (€)	10^6 (€)	10^6 (€)	$\frac{LS_{LH}-LB}{LB}$	$\frac{LH-LS_{LH}}{LH}$
	low	12.979	20.244	18.587	43.2%	8.2%
A	medium	12.125	19.110	17.542	44.7%	8.2%
	high	11.334	18.120	16.489	45.5%	9.0%
	low	4.247	6.960	6.103	43.7%	12.3%
B	medium	3.681	6.444	5.528	50.2%	14.2%
	high	3.480	6.184	5.094	46.4%	17.6%

Table 1: Results on instances with different values of the ratio between mean and peak rates. Instances A and B have 25 and 39 nodes and 63 and 86 edges, respectively.

Network	Nodes cost	Local search from LH		Local search from greedy		$\frac{LS_G-LS_{LH}}{LS_{LH}}$
	(respect to the	LS_{LH}	time	LS_G	time	
	links cost)		(minutes)		(minutes)	
	cheap	6.769	197	7.248	13	7.1%
C	equal	14.000	213	14.211	10	1.5%
	expensive	622.439	269	678.147	15	9.0%
	cheap	5.478	362	5.715	15	4.3%
D	equal	12.334	524	12.682	14	2.8%
	expensive	635.901	841	691.454	17	8.7%
	cheap	8.117	1056	8.817	33	8.6%
E	equal	18.707	1855	18.944	36	1.3%
	expensive	948.010	1812	1021.900	38	7.8%

Table 2: Results on instances with different node costs. Instances C, D, and E have 35, 35, and 45 nodes and 63, 106, and 102 edges, respectively.

Network	Nodes cost	after LB		after Local search from LB	
	(respect to the	used /	capacity	used /	capacity
	links cost)	potential	(Mbit/s)	potential	(Mbit/s)
	cheap	15 / 15	227.810	13 / 15	242.269
C	equal	13 / 15	91.123	9 / 15	111.649
	expensive	6 / 15	39.497	2 / 15	4.976
	cheap	11 / 15	74.640	9 / 15	92.523
D	equal	6 / 15	11.195	5 / 15	44.007
	expensive	7 / 15	17.261	2 / 15	7.153
	cheap	18 / 20	440.998	18 / 20	475.830
E	equal	16 / 20	194.686	13 / 20	242.891
	expensive	12 / 20	30.478	2 / 20	5.287

Table 3: Use of MPLS core nodes on instances with different node costs. Instances C, D, and E have 35, 35, and 45 nodes and 63, 106, and 102 edges, respectively.

(referred to as *equal* in the tables below), one with the first 100 times greater than the second (*expensive*) and finally one with the first 100 times smaller than the second (*cheap*).

Table 1 reports results for instances A and B (first column) for different values of the ratio between mean and peak rate (second column): in the third column the objective function value (LB) of the Lagrangian relaxation, in the fourth column and in the fifth one both the initial solution (LH) and the final solution (LS_{LH}) of the local search respectively , in the sixth column and in the seventh one the gap between upper and lower bound ($\frac{LS_{LH}-LB}{LB}$) and the gain obtained on the starting solution by using the local search ($\frac{LS_{LH}-LH}{LH}$).

We observe that the gap between upper and lower bound is high (45% on average), owing to the use of Lagrangian Relaxation solved by a continuous relaxation. The mean time to obtain these values (LB, IS_{LB} and FS_{LB}) was 562 minutes. Most of this time (84%) is spent for the local search since building the auxiliary graphs is computationally complex. Both neighborhoods require to create a new auxiliary graph each time a commodity q or an hop of a LSP s is considered. However, the gains obtained with the local search, shown in the seventh column (mean value 11%), prove that this phase of the algorithm is quite effective.

Table 2 is devoted to the results of the instances with different node cost. For each of the three instances (first column) and the three cost settings (second column) Table 2 reports: in the third and fourth column the local search solution (LS_{LH}) obtained starting from the lower bound and the time needed to obtain it, respectively, in the seventh and eighth columns the local search solution (LS_G) obtained starting from the greedy solution and the time needed to obtain it, and finally in the last column the gap between LS_G and LS_{LH}. Table 3 reports the number of MPLS nodes in the different solutions. For each of the three instances (first column) and the three cost settings (second column) Table 2 gives: in the third and fourth column the number of MPLS core nodes installed and their total capacity in the lower bound solution, in the last two columns the number of MPLS core nodes installed and their total capacity in the final heuristic solution.

We observe that, as expected, the number of core nodes equipped with multiplexing devices decreases as the node cost increases. The main reason of such behavior is that the bandwidth saving achieved with the statistical multiplexing and traffic grooming is balanced by the node cost. However, even when the nodes cost is high the solution includes a few MPLS nodes. This is not due to the bandwidth saving we achieve on the links, since links cost is negligible with respect to nodes cost. Whereas, the reason is that MPLS nodes allow to reduce the total amount of traffic into the transport nodes that can be equipped with a cheaper technology. In the opposite scenario where MPLS nodes are very cheap not all nodes are installed. This is due to the effect of statistical multiplexing on MPLS flows routing that favors route aggregation. As a result, some nodes are not crossed by MPLS traffic and do not need MPLS support.

The time needed to compute the upper and the lower bound increases when the network size increases. The mean time in the case of the network with 35 nodes and 63 links is equal to 226 minutes, 575 minutes in the case of the network with 35 nodes and 106 links, and 1574 minutes for the network with 45 nodes.

The network cost we achieved with the local search starting from the feasible solution derived with the greedy algorithm is quite good. On average it is 5.7% worse than LS_{LH}, with a maximum of 9% and a minimum of 1.3%. The mean time needed to get this results is only 21 minutes therefore the greedy algorithm seems to be the best way to find a good solution in a short time. However, we have to point out that the greedy algorithm is based on a min-hop routing. If the obtained routing corresponds to a non feasible solution, this method cannot proceed. The solution is not feasible when a node is not able to manage all the flows crossing it due to capacity constraints. On the contrary, the algorithm that obtains the starting feasible solution of the local search from the lower bound is more robust since the subgradient method used to solve the Lagrangian relaxation drives the algorithm towards feasible solutions.

Acknowledgments. The authors wish to thank E. Pedrinelli and M. Fumagalli, Alcatel Italia, for the support provided to problem definition.

References

[1] P. Belotti, A. Capone, G. Carello, F. Maffioli, F. Malucelli, Network Design with statistical Multiplexing: a MIP Approach, Technical Report, Politecnico di Milano, 2005.

[2] I. Ghamlouche, M. Gendreau, T.G. Crainic. Cycle-based neighbourhood for fixed-charge capacitated multicommodity network design. Operations Research, 51, 2003, pp. 655-667.

[3] E.Rosen, A.Viswanathan, R.Callon, Multiprotocol Label Switching Architecture, RFC 3031, January 2001.

[4] M. Pióro, D. Medhi, Routing, Flow, and Capacity Design in Communication and Computer Networks, Elsevier, 2004.

[5] G. Dahl, A. Martin, M. Stoer, Routing through Virtual Paths in Layered Telecommunication Networks, Operations Research 47, 1999, pp. 693–702.