An Efficient Algorithm for Solving Traffic Grooming Problems in Optical Networks

Hui Wang, George N. Rouskas
Operations Research and Department of Computer Science, North Carolina State University, Raleigh, NC 27695-8206 USA

***Abstract*—We consider the virtual topology and traffic routing (*VTTR*) problem, a subproblem of traffic grooming that arises as a fundamental network design problem in optical networks. The objective of *VTTR* is to determine the minimum number of lightpaths so as to satisfy a set of traffic demands, and does not take into account physical layer constraints; a routing and wavelength assignment (RWA) algorithm must reconcile the virtual topology obtained by *VTTR* with the physical topology. We propose an efficient algorithms that uses a partial LP relaxation technique with lazy constraints to improve substantially the scalability of *VTTR*, and, hence, of traffic grooming. Our approach delivers a desirable tradeoff between running time and quality of solution.**

I. Introduction

To accommodate the exponential growth of demand in communications, infrastructure that can support ever increasing amounts of traffic is highly needed. Optical networks have been commonly used as the backbone infrastructure of Internet services, since they deliver high performance in terms of both throughput and QoS. With the help of WDM technology, it is possible to transmit traffic on different wavelengths within the same optical fiber simultaneously. Currently, the data rate of a single wavelength is in the order of 10-40 Gbps, while higher rates are becoming commercially available.

The capacity of each wavelength can be significantly higher than the magnitude of individual traffic demands. The key idea of traffic grooming is to aggregate individual traffic requests onto wavelengths so as to improve bandwidth utilization across the network and minimize the use of network resources. Many variants of traffic grooming have been studied in the literature. Online versions of the problem target network environments in which traffic demands arrive in real time. Since future demands are not known, the main objective of online problems is to minimize blocking probability or maximize throughput.

Offline traffic grooming is a fundamental network design problem that has been shown to be NP-hard [4]. Such network design problems have been formulated as integer linear programs (ILPs) and assume the existence of a traffic matrix representing the demands between node pairs. Basic ILP formulations of the problem are available in [6] and [17]. Typically, the objective is to minimize the total network cost while satisfying all demands (e.g., as in [2], [8]), or to maximize the total revenue by satisfying as many traffic demands as possible given certain capacity (wavelength) constraints (e.g., as in [17]). Therefore, the number of lightpaths established to carry the traffic demands is usually taken as the metric to minimize [8]. Other cost functions have also been considered; for instance, the electronic switching cost of grooming traffic between lightpaths at intermediate switches [5] or power consumption in optical networks [16].

One essential concern about the ILP formulations is that they are solvable only for small network topologies [16]. For larger topologies representative of realistic networks, the ILP formulation cannot be solved to optimality within reasonable amounts of time (e.g., several hours). Therefore, the offline problem has generally been addressed using heuristic algorithms [1], [18]. Other approaches tackle the problem by manipulating the ILP formulation using decomposition or column generation techniques.

In [7], the original ILP is decomposed into two simpler ILPs that are solved sequentially: one that addresses only the traffic routing and lightpath routing subproblems and is solved first; and another that addresses the wavelength assignment problem only and takes as input the solution of the first ILP. In [14], a multi-level decomposition method is introduced to address the multi-layered routing and multi-rate connection characteristics of traffic grooming. In [11], the objective is to design a ring network that is able to satisfy any request graph with maximum degree at most δ. The cases of $\delta = 2$ and $\delta = 3$ were solved by graph decomposition.

Column generation techniques were developed in [3], [12]. A heuristic algorithm using column generation for a path-based formulation of the problem was developed in [3]. The key idea was to generate an optimal subset of paths efficiently. A hierarchical optimization method was proposed in [12]. The method first deals with the grooming and routing decisions using column generation to find the dual bounds and a rounding heuristic to find integral solution; wavelength assignment was then carried out in a second step.

In this paper, we define the *VTTR* problem in Section II, and in Section III, we present an algorithm based on partial LP relaxation to solve it. In Section IV, we present numerical results, and we conclude the paper in Section V.

II. The *VTTR* Problem

In [13], we proposed a decomposition of the traffic grooming (TG) problem, where the objective is to minimize the number of lightpaths, into two subproblems, the *virtual topology and traffic routing (VTTR)* subproblem and the *routing and wavelength assignment (RWA)* subproblem, that are then solved sequentially. We have shown in [13] that, whenever

This work was supported by the National Science Foundation under Grant CNS-1113191.

the network is not wavelength (bandwidth) limited, this sequential solution yields an optimal solution to the original traffic grooming problem. We have also developed scalable optimal RWA algorithms for ring and mesh topologies in [15] and [10], respectively. Therefore, our focus here is on the *VTTR* subproblem.

The *VTTR* subproblem is defined as follows:

Definition 2.1 (VTTR): Given the number N of nodes in the network graph G, the wavelength capacity C, and traffic demand matrix T, establish the minimum number of lightpaths to carry all traffic demands.

Note that the *VTTR* problem does not take as input the network graph G, only the traffic demand matrix T (and, hence, the number of nodes, N). Consequently, the output of the problem is simply the set of lightpaths to be established but *not* the (physical) paths that these lightpaths take in the network. The physical paths must be determined by the RWA algorithm in a second step of solving the original traffic grooming problem on the given network graph G.

Due to space constraints, we do not provide the ILP formulation for *VTTR*. However, *VTTR* has the same objective function as the original traffic grooming problem in minimizing the number of lightpaths, but only a subset of the constraints, namely, those related to routing the traffic demands over the lightpaths in the virtual topology (e.g., refer to [16]).

The *VTTR* subproblem is similar in concept to the virtual topology problem studied in the context of multihop broadcast-and-select (BAS) networks [9]. In multihop BAS networks it is not possible to establish direct connections between every pair of nodes, hence some traffic demands may need to be routed via intermediate nodes – just as traffic demands need to be routed over multiple lightpaths in our problem. Furthermore, just as BAS networks do not impose any physical topology constraints on the formation of direct connections due to their all-to-all broadcast nature, the *VTTR* subproblem does not impose any physical topology constraints on the formation of the virtual topology of lightpaths. The virtual topology determined by *VTTR* is reconciled with the physical topology by solving the second subproblem [13].

Ignoring the physical topology constraints in the definition of the *VTTR* subproblem has two major benefits. First, the running time for finding an optimal solution depends only on the size (i.e., number N of nodes) of the network, not its topology. Hence, the running time of a problem instance with a given demand matrix T would be identical for a sparse ring network and a dense mesh network of the same size. Second, the problem formulation does not include any binary variables. Therefore, it is possible to employ partial LP relaxation techniques so as to reduce the time required to find solutions that are close to the optimal; we describe an iterative algorithm that uses such techniques in the following section.

A. Partial LP Relaxation of VTTR

Linear programming (LP) relaxation of an ILP is the problem that arises by relaxing the integrality constraints on the relevant decision variables of the original problem. Since the original ILP formulation has stronger constraints than its LP relaxation, in the case of minimization problems such as the one we consider in this work, the optimal value of the LP relaxation provides a lower bound for the original ILP formulation. Although LP relaxation sacrifices optimality, the relaxed problem can be solved as a linear program in time that may be orders of magnitude lower than the time to solve the original ILP.

TABLE I
CPU TIME COMPARISON OF *VTTR* AND *VTTR-rlx*, $N = 16$

t_{max}	CPU Time (sec)	
	VTTR	*VTTR-rlx*
10	21629.1	0.184
20	21626.7	0.199
30	21626.6	0.200
40	21732.8	0.242
50	21740.4	0.259
60	21625.9	0.188

Definition 2.2: (VTTR-rlx) Given the number N of nodes in the graph G of *TG*, the wavelength capacity C, and traffic demand matrix T, establish the minimum number of lightpaths to carry all traffic demands while allowing *fractional* lightpaths to exist between any pair of nodes.

Let $\{b_{ij}\}$ be integer variables denoting the number of lightpaths from node i to node j in the virtual topology. *VTTR-rlx* can be derived from *VTTR* by replacing the integer variables $\{b_{ij}\}$ with non-negative real variables $\{\bar{b}_{ij}\}$, while maintaining the integrality constraints on all other integer variables in the formulation. Then, *VTTR-rlx* represents a *partial* LP relaxation of *VTTR*, and can be formulated as a mixed integer linear program (MILP). Also, if $\{\bar{b}_{ij}\}$ is a feasible solution to *VTTR-rlx*, then $\{\lceil \bar{b}_{ij} \rceil\}$ is a feasible solution to *VTTR*.

We compared the *VTTR* and *VTTR-rlx* on problem instances defined on a 16-node network. For the comparison, we generated traffic instances by setting each traffic demand t_{sd} as a random integer in the range $[0, t_{max}]$. We let parameter $t_{max} = 10, 20, 30, 40, 50, 60$, and for each value of t_{max} we generated ten traffic matrices (i.e., problem instances) that were used to solve both *VTTR* and *VTTR-rlx*.

Table I presents the CPU time (in sec), averaged over the ten random instances, that CPLEX needs to solve the *VTTR* and *VTTR-rlx* problems for each value of t_{max}. We imposed a six-hour limit on running time; if an instance did not complete within this time limit, we recorded the best available solution found until that time and terminated the CPLEX process. We note that the CPU times do not vary much across the values of t_{max}, but solving the partial LP relaxation *VTTR-rlx* takes a fraction of a second whereas solving the *VTTR* ILP takes longer than the six hour limit we imposed.

Table II compares the best available solutions to *VTTR* obtained within the six-hour limit, to the optimal solutions to *VTTR-rlx*, in terms of the objective value (i.e., number of lightpaths). For each row of the table (i.e., a specific value of t_{max}), the values shown are averages over the corresponding

TABLE II
OBJECTIVE VALUE COMPARISON OF *VTTR* AND *VTTR-rlx*, $N = 16$

t_{max}	Objective Value (# of lightpaths)		
	VTTR	*VTTR-rlx*	
	(best available)	(optimal)	(rounded-up)
10	101.7	74.1	217.9
20	173.2	150.0	274.2
30	250.6	226.6	340.1
40	327.1	302.6	423.6
50	389.3	366.4	480.1
60	468.2	443.5	558.5

ten traffic instances. However, the optimal solution to partial LP relaxation *VTTR-rlx* is a lower bound, but not necessarily a feasible solution to *VTTR*. Therefore, we also present the objective value of the feasible solution obtained by rounding up the real values $\bar{b}^{\star}_{ij}$ of the optimal solution to *VTTR-rlx*.

From the two tables we make two important observations. First, the integral constraints of the lightpath variables b_{ij} play an important role in increasing the complexity of the branch-and-bound process of the ILP solver. Second, rounding up the real lightpath values $\bar{b}^{\star}_{ij}$ results in a large optimality gap. Based on these observations, in the next section we develop an iterative algorithm that strikes a good balance between running time and quality of solution.

III. AN ITERATIVE ALGORITHM FOR *VTTR*

Consider the optimal solution $\{\bar{b}^{\star}_{ij}\}$ to the *VTTR-rlx* problem and the corresponding feasible solution $\{\lceil \bar{b}^{\star}_{ij} \rceil\}$ to *VTTR*, obtained by rounding. Let us define:

$$U_{ij} = \frac{\bar{b}^{\star}_{ij}}{\lceil \bar{b}^{\star}_{ij} \rceil}, \quad \bar{b}^{\star}_{ij} > 0. \tag{1}$$

The quantity U_{ij} represents the utilization of the lightpaths from node i to node j in the rounded-up feasible solution. When the utilization is high (i.e., U_{ij} is close to 1.0), the corresponding lightpath resources are used effectively in the solution; furthermore, rounding up the corresponding lightpath variable to obtain a feasible solution makes only a small contribution to the optimality gap. The opposite is true when the utilization of a set of lightpaths is low.

We define U_l and U_h, $0 \leq U_l \leq U_h \leq 1$, as a low and high threshold, respectively on lightpath utilization. The key idea of the iterative algorithm for *VTTR* is to treat the integer constraints on lightpath variables b_{ij} as *lazy* constraints, and activate only a subset of them at each iteration, based on how they relate to this pair of utilization thresholds.

The algorithm starts by solving the partial LP relaxation *VTTR-rlx* in which none of the integrality constraints on $\{b_{ij}\}$ are activated. If all lightpath variables in the optimal solution are integer, then this is a feasible (and optimal) solution to *VTTR*. Otherwise, we examine the solution to identify all lightpath variables $\bar{b}_{ij}$ with a utilization $U_{ij} \leq U_l$ or $U_{ij} \geq U_h$. We then activate two sets of *equality constraints* on the identified variables, i.e., we solve a modified version of *VTTR-rlx* in which the identified variables $\bar{b}_{ij}$ are set to be equal to the floor (respectively, ceiling) of the corresponding optimal solution obtained from *VTTR-rlx* if $U_{ij} \leq U_l$ (respectively, $U_{ij} \geq U_h$). We repeat this process, increasing the threshold value U_l and decreasing U_h at each iteration, until one of the following stopping criteria is satisfied:

1) all lightpath variables in the solution are integer, and hence represent an optimal solution to *VTTR*;
2) the threshold pair (U_l, U_h) reaches a predetermined value, or
3) the improvement in the value of the objective function over the previous iteration is less than a predetermined minimum value δ.

A combination of the above criteria may be used, e.g., stop whenever the thresholds have reached a predetermined value or the improvement over the previous iteration is less than δ, whichever is satisfied first.

The algorithm can be described by these steps:

1) Initialization: $i \leftarrow 0; U_l \leftarrow 0.1$; $U_h \leftarrow 0.9$.
2) Solve *VTTR-rlx* with no integer constraints on $\{b_{ij}\}$ activated. Calculate and record U_{ij} for lightpaths between all node pairs (i, j) in the optimal solution.
3) Re-solve *VTTR-rlx* with the following two sets of equality constraints activated:
$$\bar{b}_{ij} = \lceil \bar{b}_{ij} \rceil \quad \forall\ i, j:\ U_{ij} \geq U_h$$
$$\bar{b}_{ij} = \lfloor \bar{b}_{ij} \rfloor \quad \forall\ i, j:\ U_{ij} \leq U_l$$
Find the new optimal solution, and determine the objective value of the corresponding feasible solution obtained by rounding up all non-integer lightpath variables.
4) If the stopping criterion is satisfied, return the current solution; otherwise set $i \leftarrow i + 1; U_l \leftarrow U_l + 0.1$; $U_h \leftarrow U_h - 0.1$ and repeat from Step 2.

We note that, at each iteration of the algorithm, a tighter partial LP relaxation of *VTTR* with a larger number of equality constraints is considered, generally requiring longer time to solve. On the other hand, the objective value of the solution improves with each iteration. By selecting an appropriate stopping criterion, especially in terms of the threshold values on lightpath utilization, this algorithm may be designed to deliver a desirable tradeoff between running time and quality of the final solution.

IV. NUMERICAL RESULTS

In this section, we present the results of a simulation study we conducted to investigate the performance of the *VTTR* problem and the iterative algorithm described in the previous section for the VTTR problem in terms of scalability (i.e., running time) and quality of solution. All results were obtained by running the IBM Ilog CPLEX 12 optimization tool on a cluster of identical compute nodes with dual Woodcrest Xeon CPU at 2.33GHz with 1333MHz memory bus, 4GB of memory and 4MB L2 cache.

Our study involves a large set of problem instances defined on several network sizes[1] with various random traffic loads.

[1]We remind the reader that the *VTTR* problem does not account for the physical topology of the network.

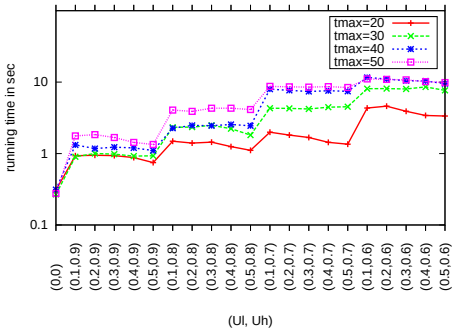

Fig. 1. CPU time of the iterative algorithm, $N = 16$

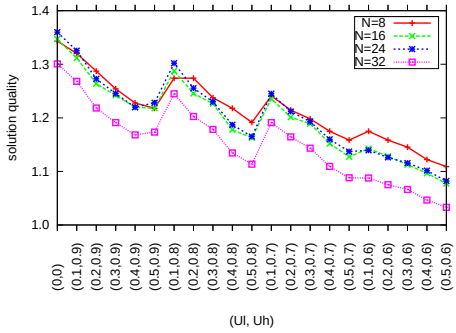

Fig. 2. Solution quality of the iterative algorithm, $t_{max} = 30$

In particular, we consider networks with $N = 8$, 16, 24, and 32 nodes. In all the simulations, we set the wavelength capacity $C = 16$. For each network topology, we consider several problem instances. For each problem instance, the traffic demand matrix $T = [t_{sd}]$ is generated by drawing the (integer) traffic demands uniformly at random in the interval $[0, t_{max}]$. The values of t_{max} we used in the simulations are 20, 30, 40, and 50. Each data point in the figures we present in this section represents the average of 10 random problem instances for the stated values of the input parameters.

In this section, we set the relative optimality gap to 2% for all simulations. Consequently, CPLEX terminates when it finds a solution that is within 2% of the optimal for the problem at hand, rather than continuing until the problem is solved to optimality. Later in this section we will investigate how the running time required to solve the *VTTR* problem is affected by this optimality gap.

Figure 1 plots the running time of the iterative algorithm for *VTTR* as a function of the thresholds U_l and U_h; note that the pair (0,0) in the figure corresponds to the solution of the relaxed problem *VTTR-rlx*. The figure plots results for networks with $N = 16$ nodes and various values of t_{max}. As expected, the running time generally increases as U_l increases or U_h decreases, since in both cases equality constraints are imposed on a larger number of lightpath variables. We also see that the running time is not significantly affected by the value of parameter t_{max}. When t_{max} increases from 20 to 50, the running time only increases by a few seconds. This shows that the algorithm is relatively stable to the change of t_{max}, and hence, is effective across a range of traffic loads.

Based on the last observation, for the simulations in the remainder of this section we have fixed the value of $t_{max} = 30$; with this value of t_{max}, the average size of demands between any source-destination pair is close to the capacity $C = 16$ of a wavelength. Results for other values of t_{max} exhibit the same behavior and are omitted.

Figure 2 plots the quality of the solution of the iterative algorithm as a function of the pair of thresholds (U_l, U_h) and for various network sizes. The quality of the solution is defined as: $\sum \lceil \bar{b}^{\star}_{ij} \rceil / \sum b^{\star}_{ij}$. The numerator in this expression is the value of the feasible solution to *VTTR* obtained by rounding up the lightpath variables in the optimal solution to the modified version of *VTTR-rlx* problem for the given value of a pair of thresholds. The denominator is the value of the objective function for the optimal solution to the *VTTR* problem (obtained within a 2% relative optimality gap, as we explained earlier). A low value of the above expression denotes a higher quality solution. As expected, the quality of the solution starts away from optimal for *VTTR-rlx* (i.e., the point (0,0) in the figure) since all integer lightpath variables are relaxed; the solution quality then improves as U_l increases or U_h decreases. The best result is achieved for the threshold pair $(U_l, U_h) = 0.5, 0.6$. Importantly, for all network sizes shown in the figure, the solution is at most 11% of the optimal as soon as $(U_l, U_h) = (0.5, 0.6)$; this worst case occurs for $N = 8$, and the gap decreases as the network size N increases. For the 32-node network, the gap is as small as 3%. Note that this pair of values for (U_l, U_h) ensures that no wavelength is under-utilized (i.e., it is filled to 50% at minimum) while also leaving some room to accommodate future demands without necessarily setting up additional lightpaths.

Figure 3 plots the running time of the algorithm as a function of the network size N. For this simulation, we set $(U_l, U_h) = (0.5, 0.6)$, the value that achieves the solution of highest quality (i.e., the smallest number of lightpaths) as we observed earlier. As we can see, the algorithm makes it possible to solve the *VTTR* problem for network sizes up to $N = 32$ within one hour without sacrificing much in terms of optimality (as Figure 2 indicates). Such problem instances are impossible to solve by directly using the original ILP formulation for the *VTTR* problem, even applying a 2% optimality gap.

Finally, Figure 4 compares the running time as a function of network size of three methods for solving the *VTTR* problem:

1) solving *VTTR* to optimality;
2) solving *VTTR* with a 2% relative optimality gap; and
3) solving *VTTR-rlx* using $(U_l, U_h) = (0.5, 0.6)$.

All simulations are allowed to run for as long as 24 hours. As we can see, the running time of the first method (i.e., directly solving the ILP formulation for the VTTR problem to optimality), grows consistently with the netwok size N. Within the 24-hour time limit, this method can only solve network sizes up to 16 nodes. Setting the optimality gap to 2% (i.e., using the second method listed above) reduces the

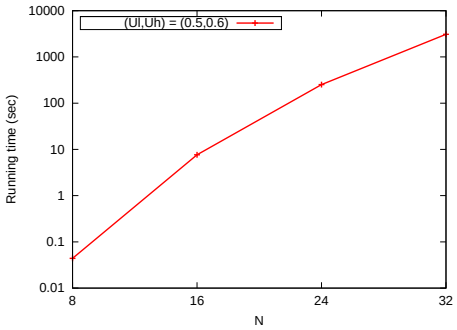

Fig. 3. CPU time of iterative algorithm, $(U_l, U_h) = (0.5, 0.6), t_{max} = 30$

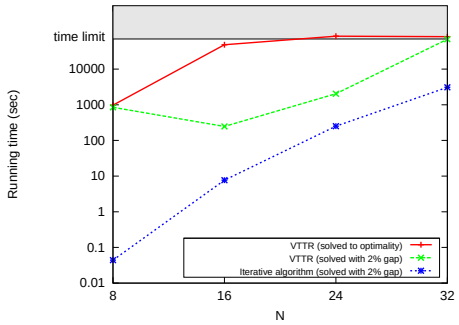

Fig. 4. CPU time comparison as a function of network size N, $t_{max} = 30$

running time of finding a solution by about two orders of magnitude, but even in this case it is not possible to solve 32-node networks within the 24-hour time limit. Notice that with the second method, the running time for the 8-node network is longer than the time it takes to solve the 16-node network. This drop in running time when the network size increases from 8 to 16 can be explained by making the observation that, with a fixed value of $t_{max} = 30$, as the network size grows so does the offered traffic and the optimal objective value (i.e., number of lightpaths). Hence, CPLEX may take a shorter time to reach a solution that is 2% from the optimal solution as the network size increases, as the absolute difference from the optimal solution is larger. Of course, as the network size increases further, the increase in the number of variables and constraints becomes once again the factor determining the running time; hence the increase as network size grows to 24 and beyond.

Finally, the iterative algorithm we described in the previous section is significantly faster over all network sizes, and reduces the running time by more than one order of magnitude compared to solving *VTTR* directly within a 2% optimality gap. In particular, the iterative algorithm solves the *VTTR* problem on the 32-node network in about 3000 seconds (i.e., less than 1 hour), while also obtaining a solution that is within 3% of the best solution (refer also to Figure 2) that we were able to obtain after running the second method for 24 hours. Note that in Figure 4, we used $(U_l, U_h) = (0.5, 0.6)$ as the pair of thresholds for the algorithm; however, other pairs of thresholds may be applied to further reduce the running time.

Based on these results, we conclude that, for small networks, e.g., of size between 8-10 nodes, the *VTTR* problem can be directly solved by using the MILP formulation with or without imposing a 2% optimality gap. However, for larger networks, the iterative algorithm we presented in Section III is more efficient and effective.

V. Conclusions

We have presented an efficient iterative algorithm based on partial LP relaxation to solve the virtual topology and traffic routing (*VTTR*) problem, that arises in a sequential decomposition of the traffic grooming problem. The threshold parameters of the algorithm may be tuned to achieve a desired tradeoff between running time and quality of the final solution for *VTTR*. By combining this algorithm with scalable RWA algorithms [10], [15], this decomposition approach scales well to both ring and mesh network topologies and enables operators to carry out extensive "what-if"analysis in optimizing their network.

References

[1] B. Chen, G. N. Rouskas, and R. Dutta. On hierarchical traffic grooming in WDM networks. *IEEE/ACM Transactions on Networking*, 16(5):1226–1238, October 2008.

[2] A.L. Chiu and E.H. Modiano. Traffic grooming algorithms for reducing electronic multiplexing costs in WDM ring networks. *IEEE/OSA Journal of Lightwave Technology*, 18(1):2 –12, Jan 2000.

[3] M. Dawande, R. Gupta, S. Naranpanawe, and C. Sriskandarajah. A traffic-grooming algorithm in wavelength-routed optical networks. *INFORMS Journal on Computing*, 19(4):565 – 574, 2007.

[4] R. Dutta, S. Huang, and G. N. Rouskas. Traffic grooming in path, star, and tree networks: Complexity bounds and algorithms, April 2006.

[5] R. Dutta and G.N. Rouskas. On optimal traffic grooming in WDM rings. *IEEE J. Selected Areas Comm.*, 20(1):110 –121, January 2002.

[6] R. Dutta and G.N. Rouskas. Traffic grooming in WDM networks: past and future. *IEEE Network*, 16(6):46 – 56, nov/dec 2002.

[7] J.Q. Hu and B. Leida. Traffic grooming, routing, and wavelength assignment in optical WDM mesh networks. In *Proceedings of IEEE INFOCOM 2004*, March 2004.

[8] V. R. Konda and T. Y. Chow. Algorithm for traffic grooming in optical networks to minimize the number of transceivers. *IEEE Workshop on High Performance Switching and Routing*, pages 218 – 221, 2001.

[9] J.-F.P. Labourdette and A.S. Acampora. Logically rearrangeable multihop lightwave networks. *IEEE Trans. Comm.*, 39(8):1223-1230, 1991.

[10] Z. Liu and G. N. Rouskas. A fast path-based ILP formulation for offline RWA in mesh optical networks. *Proc. of IEEE GLOBECOM*, Dec. 2012.

[11] X. Munoz and I. Sau. Traffic grooming in unidirectional WDM rings with bounded degree request graph. *Graph-Theoretic Concepts in Computer Science*, LNCS 5344, pages 300–311, 2008.

[12] B. Vignac, B. Jaumard, and F. Vanderbeck. A hierarchical optimization approach to optical network design where traffic grooming and routing is solved by column generation. In *ICON*, 2009.

[13] H. Wang, Z. Liu, and G. N. Rouskas. Scalable traffic grooming in optical networks. In *Proc. of ACP 2012*, November 2012.

[14] W. Yao, G. Sahin, M. Li, and B. Ramamurthy. Analysis of multi-hop traffic grooming in WDM mesh networks. In *Proceedings of BroadNets 2005*, pages 165 – 174, October 2005.

[15] E. Yetginer, Z. Liu, and G. N. Rouskas. Fast exact ILP decompositions for ring RWA. *J. Optical Comm. and Netw.*, 3(7):577–586, July 2011.

[16] E. Yetginer and G. N. Rouskas. Power efficient traffic grooming in optical WDM networks. *Proc. of IEEE GLOBECOM 2009*, Dec. 2009.

[17] K. Zhu and B. Mukherjee. Traffic grooming in an optical WDM mesh network. *IEEE J. Selected Areas Comm.*, 20(1):122–133, 2002.

[18] K. Zhu and B. Mukherjee. A review of traffic grooming in WDM optical networks: Architectures and challenges. *Optical Networks Magazine*, 4(2):55–64, March/April 2003.