

# A Novel Efficient Technique for Traffic Grooming in WDM SONET with Multiple Line Speeds

Huan Liu, Fouad Tobagi  
Department of Electrical Engineering  
Stanford University, Stanford, CA 94305  
Email: huanliu, tobagi@stanford.edu

**Abstract**—SONET rings remain the dominant optical transport architecture in the metropolitan area. To support traffic growth on these rings, WDM technology is used to provide multiple SONET rings on the same fiber, each SONET ring running on a separate wavelength. Traffic grooming refers to intelligently arranging low speed traffic streams onto different SONET rings and selecting the proper line speed for each wavelength so as to minimize certain cost objective, such as the total cost of electronic Add-Drop Multiplexer (ADM) equipments used. The problem can be formulated as an Integer Linear Programming (ILP) problem by generalizing the formulation in [1] [2] to support multiple line speeds. Unfortunately, solving the ILP directly could be very computation intensive. In this paper, we propose a non-linear formulation instead, and then solve it by decomposing it into several smaller ILP subproblems, each can be solved separately by an ILP solver. Decomposition allows us to exploit the symmetry in the problem structure, and cut down the solution space dramatically, therefore, reducing the computation time to solve a problem. Even if we may have to terminate the computation early for large size problems, decomposition allows us to explore a larger portion of the solution space in a given amount of time, therefore, obtain better results.

## I. INTRODUCTION

SONET rings remain the dominant architecture for optical transport, especially in the metropolitan area. To support traffic growth on these rings, WDM technology is used to provide multiple SONET rings in the same fiber, each SONET ring running on a separate wavelength. This solution is attractive because it provides an easy upgrade path, avoiding the need to lay out additional fibers to meet the traffic growth. In the WDM SONET network, each wavelength may run at a different line speed, such as OC-3, OC-12 or OC-48, depending on the total amount of traffic carried on that wavelength.

There are two types of SONET rings: Unidirectional Path Switched Rings (UPSR) and Bidirectional Line Switched Rings (BLSR). In a UPSR, traffic is always routed in one direction (e.g., clockwise). To route a symmetric traffic stream, i.e., the same amount of traffic from node A to B and from node B to A, the same amount of capacity is reserved on every fiber along the ring. Therefore, no spatial capacity reuse is possible. UPSR ring consists of 2 fibers where one of the fibers is dedicated for protection purpose. In BLSR, traffic can be routed either in the clockwise direction or in the counter-clockwise direction. BLSR consists of either two or four fibers. They are commonly referred as BLSR/2 and BLSR/4 respectively. UPSR is commonly used in Metro

access networks, whereas BLSR is commonly used in Metro backbone networks.

Two kinds of equipment are used at a node in WDM SONET networks: Optical Add-Drop Multiplexer (OADM) and electronic Add-Drop Multiplexer (ADM), both of which are very costly. OADM allows a node to add/drop a selected subset of wavelengths into and out of the fiber and let the remaining wavelengths pass through transparently; for a node to be part of a particular SONET ring, the wavelength corresponding to that SONET ring must be in the set of wavelengths that its OADM adds/drops. Electronic ADM can further allow any circuit within a wavelength to be added/dropped at the node. For each wavelength that is added/dropped at a given node, there is a need for an electronic ADM which adds/drops the specific set of circuits carried in the wavelength that is of interest to the node and leaves the rest of circuits intact.

Traffic grooming refers to intelligently arranging low speed traffic streams onto different SONET rings and selecting the proper line speed for each wavelength so as to minimize certain cost objective. A typical objective used in past research work [3] [4] [5] is to reduce the total cost of electronic Add-Drop Multiplexer (ADM) equipments used. Since an OADM is needed at every node, the cost of OADMs is fixed. However, the cost of electronic ADM components varies greatly depending on how traffic is groomed. We do not have to install an ADM for every wavelength at every node if no traffic terminates on that wavelength at that node. By intelligently grooming traffic, we can reduce the total cost of all ADMs needed. The potential saving can be significant and grows with both the network size and internodal demand [6].

Traffic grooming in SONET UPSR rings with one single line speed has been considered in [7], where heuristic algorithms were proposed for specific traffic patterns, and in [1] [2], where an Integer Linear Programming (ILP) formulation was given. Prior work on traffic grooming with multiple line speeds can be found in [5] [8] [9]. Using a simple lower bound, it was shown in [5] that having multiple line speeds can lower the total cost. An algorithm was proposed in [8] [9] for single-hub SONET rings.

In this paper, we first generalize the ILP formulation given in [1] [2] to support multiple line speeds. We then propose a non-linear formulation and solve it by decomposing it into several subproblems where each subproblem is a smaller ILP problem. Decomposition allows us to exploit the symmetry

in the problem structure, and cut down the solution space dramatically, therefore, we are able to solve the problem in a much shorter amount of time. If we have to terminate the computation early for large size problems, the decomposition technique allows us to explore a larger portion of the solution space in a fixed amount of time, therefore, we are able to reach better solutions.

We present the technique in the context of SONET UPSR rings. However, we note that the same technique can be applied to SONET BLSR rings as well.

## II. LINEAR FORMULATION

The traffic grooming problem supporting only a single line speed has been formulated as an Integer Linear Programming (ILP) problem before [1] [2]. We first extend the formulation to the multiple line speeds case. We follow the following notations.

- $N$ : The number of nodes in the ring, i.e., the number of traffic add/drop sites around the ring.
- $W$ : The number of wavelengths available in the WDM system.
- $R$ : The number of line speeds available. For example, if OC-3, OC-12 and OC-48 line speeds are available,  $R = 3$ .
- $K$ : Total number of traffic demands.
- $w$ : When used as a subscript, it denotes a wavelength.  $w \in 1, \dots, W$ .
- $r$ : When used as a subscript, it denotes a particular line speed.  $r \in 1, \dots, R$ .
- $i, j$ : When used as subscripts, they denote nodes on the rings.  $i, j \in 1, \dots, N$ .
- $k$ : When used as a subscript, it denotes a particular traffic demand.  $k \in 1, \dots, K$ .
- $s(k)$ : The source node of the  $k$ th traffic demand.
- $d(k)$ : The destination node of the  $k$ th traffic demand.
- $f(i, j)$ : Total traffic demand from node  $i$  to node  $j$ . We assume all traffic demands are bidirectional, i.e., if there is some amount of traffic from node  $i$  to node  $j$ , there will also be the same amount of traffic from node  $j$  to node  $i$ . We assume all traffic demands are expressed as integer multiples of the lowest traffic granularity, e.g., in multiples of OC-3 circuits. We assume  $f(i, i)$  is always 0.
- $g_r$ : Capacity of the  $r$ th line speed in multiples of the greatest sub-multiple. If the greatest sub-multiple is OC-3, then the capacity of OC-3 is 1, the capacity of OC-12 is 4 and the capacity of OC-48 is 16. We assume that capacity is an integer number throughout this paper.
- $c_r$ : The cost of an ADM running at the  $r$ th line speed. An OC-48 ADM costs more than an OC-12 ADM, which in turn costs more than an OC-3 ADM.
- $M$ : A large constant.

In order to linearize the formulation, we assume there are total  $WR$  SONET rings,  $W$  of them running at each line speed. To meet the wavelength constraint, we then make sure that only one of the  $R$  rings running at a wavelength can carry

traffic. We use the term “ring  $wr$ ” to refer to a ring running at the  $r$ th line speed on wavelength  $w$ .

We use the following variables in the formulation.

- $\delta_{wr}$ : It is a binary variable. It is 1 if ring  $wr$  can carry traffic, otherwise it is 0.
- $x_{kwr}$ : It is the total traffic flow for the  $k$ th traffic demand from node  $s(k)$  to node  $d(k)$  on ring  $wr$ .
- $y_{iwr}$ : It is a binary variable indicating whether traffic will be added or dropped at node  $i$  on ring  $wr$ . If it is 1, an ADM running at the  $r$ th line speed has to be installed at node  $i$  on wavelength  $w$ .

We now present the ILP formulation that supports multiple line speeds as follows. We refer to this formulation as F1.

$$(F1) \quad \min \sum_{i=1}^N \sum_{w=1}^W \sum_{r=1}^R c_r y_{iwr} \quad (1)$$

Subject to

$$\sum_{r=1}^R \sum_{w=1}^W x_{kwr} = f(s(k), d(k)) \quad \forall k \quad (2)$$

$$\sum_{k=1}^K x_{kwr} \leq \delta_{wr} g_r \quad \forall w, r \quad (3)$$

$$M y_{iwr} \geq \sum_{k|i=s(k)} x_{kwr} + \sum_{k|i=d(k)} x_{kwr} \quad \forall i, w, r \quad (4)$$

$$\sum_{r=1}^R \delta_{wr} \leq 1 \quad \forall w \quad (5)$$

$$x_{kwr} \text{ are integers} \quad (6)$$

$$\delta_{wr}, y_{iwr} \in \{0, 1\} \quad (7)$$

The equations are explained in the following.

- Equation (2) ensures that the traffic demand for each source destination pair is met.
- Equation (3) makes sure the total flow on any ring does not exceed the ring capacity. If  $\delta_{wr}$  is 1, ring  $wr$  can carry traffic, and the ring capacity will be  $g_r$ . Otherwise, the ring capacity is 0 which ensures that there is no traffic on ring  $wr$ .
- Equation (4) sets  $y_{iwr}$  to 1 if any traffic is added or dropped at node  $i$  on ring  $wr$ .
- Equation (5) ensures that only one out of the  $R$  rings on wavelength  $w$  can carry traffic.
- The optimization objective (equation 1) is to minimize the total cost of all ADMs that have to be used.

Even though this formulation can be solved directly using an ILP solver, the computation time could be very high [1] [2], especially when the network has a large number of nodes and supports many line speeds. We propose a non-linear formulation in the next section and solve it by decomposition.

## III. NON-LINEAR FORMULATION

In the formulation presented in the last section, we linearize the model by assuming there are  $WR$  rings,  $R$  of them running at each wavelength. This introduces almost  $R$  times more variables and constraints, which increases the complexity

of the problem. In this section, we present a non-linear formulation and show how it can be solved more efficiently than the linear formulation. In the non-linear formulation, we assume there are only  $W$  SONET rings, and the constraints make sure that only one line speed can be selected for each wavelength.

We use the following variables. They are similar to the variables in the linear formulation.

- $\delta_{wr}$ : It is a binary variable. It is 1 if wavelength  $w$  runs at the  $r$ th line speed, otherwise it is 0.
- $x_{kw}$ : It is the total traffic flow for the  $k$ th traffic demand from node  $s(k)$  to node  $d(k)$  on wavelength  $w$ .
- $y_{iw}$ : It is a binary variable indicating whether traffic will be added or dropped at node  $i$  on wavelength  $w$ . If it is 1, an ADM has to be installed at node  $i$  on wavelength  $w$ . The line speed of the ADM is determined by the  $\delta_{wr}$  variables.

We now give the Non-linear Integer Programming (NIP) problem formulation as follows. We refer to this formulation as F2.

$$(F2) \quad \min \sum_{i=1}^N \sum_{w=1}^W \sum_{r=1}^R c_r \delta_{wr} y_{iw} \quad (8)$$

Subject to

$$\sum_{w=1}^W x_{kw} = f(s(k), d(k)) \quad \forall k \quad (9)$$

$$\sum_{k=1}^K x_{kw} \leq \sum_{r=1}^R \delta_{wr} g_r \quad \forall w \quad (10)$$

$$M y_{iw} \geq \sum_{k|i=s(k)} x_{kw} + \sum_{k|i=d(k)} x_{kw} \quad \forall i, w \quad (11)$$

$$\sum_{r=1}^R \delta_{wr} \leq 1 \quad \forall w \quad (12)$$

$$x_{kw} \text{ are integers} \quad (13)$$

$$\delta_{wr}, y_{iw} \in \{0, 1\} \quad (14)$$

The equations in formulation F2 are similar to the ones in formulation F1. They are explained in the following.

- Equation (9) ensures that the traffic demand for each source destination pair is met.
- Equation (10) makes sure the total flow on any ring does not exceed the ring capacity. The  $\delta_{wr}$  variables determine which line speed's capacity should be applied.
- Equation (11) sets  $y_{iw}$  to 1 if any traffic is added or dropped at node  $i$  on ring  $w$ .
- Equation (12) along with the binary constraint on  $\delta_{wr}$  variables ensure that a wavelength can only be set to one of the  $R$  line speeds.
- The optimization objective (equation 8) is to minimize the total cost of all ADMs that have to be used. The  $c_r \delta_{wr}$  factor determines which cost should be applied to an ADM on wavelength  $w$ . Since both  $\delta_{wr}$  and  $y_{iw}$  are variables in this formulation, the cost function is quadratic.

All constraints in formulation F2 are actually linear, only the objective function is quadratic. This formulation can be solved directly using a quadratic programming solver. Unfortunately, using a quadratic programming solver will not necessarily solve the problem faster because the solution space remains the same.

Another way of solving formulation F2 is to decompose the problem into several subproblems, solve each one in turn, then pick the best solution among them. Note that if we fix the  $\delta_{wr}$  variables to be constants, then the objective function (8) is again linear. Therefore, if we decompose formulation F2 into several subproblems: each one corresponds to one particular assignment of the  $\delta_{wr}$  variables, we can solve each subproblem directly using an ILP solver. Unfortunately, using this approach alone will not solve the problem any faster either, again because the solution space remains the same. In fact, decomposition will actually be slower than solving formulation F1 directly. This is because the ILP solver can use the lower bound it computes to potentially prune large solution space which would otherwise have to be searched explicitly when solving each subproblem in turn.

Fortunately, decomposition allows us to exploit the symmetry in the problem structure, therefore, reduce the computation time. Consider a solution to the traffic grooming problem where the  $i$ th wavelength is set to the  $r_i$ th line speed and the  $j$ th wavelength is set to the  $r_j$ th line speed, we can construct an equivalent solution by setting the  $i$ th wavelength to use the  $r_j$ th line speed and setting the  $j$ th wavelength to use the  $r_i$ th line speed, then moving all traffic routed on wavelength  $j$  to wavelength  $i$  and moving all traffic routed on wavelength  $i$  to wavelength  $j$ . The solution thus constructed is clearly identical to the original solution, however, formulation F1 treats them as separate solutions. An ILP solver could not exploit this symmetry in the problem structure to reduce computation time.

We observe that  $\delta_{wr}$  are binary variables, and because of equation (12), at most one out of the  $R$  number of  $\delta_{wr}$  variables for any  $w$  can be 1. Therefore, the total number of possible assignments for  $\delta_{wr}$  variables, as would have been enumerated by an ILP solver, is  $R^W$ . However, out of the  $R^W$  combinations of  $\delta_{wr}$  variables, only a limited subset is unique. The exact line speed assignment for each wavelength is not important, but rather, the number of wavelengths set to any line speed is more important in determining a unique solution.

For example, if two line speeds are available, then  $x$  number of wavelengths can be set to the first line speed and  $W - x$  number of wavelengths can be set to the second line speed, where  $x \in \{0, 1, \dots, W\}$ . Each value of  $x$  corresponds to one unique combination, therefore, there are only  $W + 1$  unique combinations out of the  $2^W$  possibilities.

In the following, we use a tuple  $(l_1, l_2, \dots, l_R)$  to represent a unique combination, where  $l_i$  is the number of wavelengths that are set to the  $i$ th line speed. Naturally,  $l_i \in \{0, 1, \dots, W\}$  and  $\sum l_i = W$ . We assume the first line speed has lower capacity than the second line speed, which in turn has lower capacity than the third line speed, and so on. For example,

if three line speeds are supported (OC-3, OC-12 and OC-48), (1, 3, 4) represents 8 wavelengths in total, 1 at the first line speed (OC-3), 3 at the second line speed (OC-12) and 4 at the third line speed (OC-48).

In general, the number of unique combinations (tuples) when there are many line speeds available is determined by the following formula.

$$\sum_{l_1=0}^W \sum_{l_2=l_1}^W \dots \sum_{l_R=l_{R-1}}^W 1$$

When two line speeds are available, the above formula becomes  $W + 1$ , and when three line speeds are available, the above formula becomes  $(W + 1)(W + 2)/2$ . Consider a network with 10 wavelengths and 3 line speeds, the number of unique tuples is 66 out of the  $3^{10}$  possibilities. If we can examine only these unique tuples, we can greatly reduce the solution space, and at the same time, guarantee that the optimal solution can still be found.

We propose two algorithms to solve formulation F2. They differ in the sequence they examine these unique tuples. The first algorithm is called the Largest Line Speed First (LLSF) algorithm where we first try the tuples with the highest line speeds. The second algorithm is called the Smallest Line Speed First (SLSF) algorithm where we first try the tuples with the lowest line speeds. We say a tuple  $t_a$  has higher line speeds than another tuple  $t_b$  if  $t_a$  has more wavelengths assigned to the highest line speed; if  $t_a$  and  $t_b$  have the same number of wavelengths assigned to the highest line speed, the next highest line speed is used to break ties and so on. For example, (1, 2, 5) has higher line speeds than (1, 3, 4), which in turn has higher line speeds than (2, 2, 4).

The LLSF (SLSF) algorithm proceeds as follows. It first decomposes formulation F2 into several ILP subproblems. Then it only examines the subproblems corresponding to the unique tuples. These unique tuples are sorted. The LLSF (SLSF) algorithm then enumerates the unique tuples in the descending (ascending) order, and solves the subproblem corresponding to the tuple using an ILP solver. The solution from one subproblem is used as an upper bound when solving the next subproblem. This will help to cut down the total solution time because the bound can be used to prune inferior solutions. This process continues until all unique tuples have been examined and the best solution is found.

#### IV. NUMERICAL RESULTS

To evaluate our proposed technique, we consider a SONET network supporting 3 line speeds: OC-3, OC-12 and OC-48. We assume all traffic demands are expressed as multiples of OC-3 circuits.

We use the same cost ratio assumption as in [5], i.e., OC-48 ADMs are 2.5 times more costly than OC-12 ADMs, and OC-12 ADMs are 2.5 times more costly than OC-3 ADMs. We also use the same capacity ratio assumption, i.e., OC-48 has 4 times the capacity of OC-12 and OC-12 has 4 times the capacity of OC-3. We normalize the cost of an OC-3 ADM to

be 1. Therefore, an OC-12 ADM will cost 2.5 and an OC-48 ADM will cost 6.25.

We consider two types of traffic demand patterns. The first is all to all unit uniform traffic, i.e., there is an OC-3 traffic demand between every pair of nodes. We call it the *uniform traffic pattern*. It is the worst case as it entails the most number of commodities. The second traffic pattern we consider is *random traffic pattern*. Both the source and the destination nodes of each traffic demand are randomly generated. In addition, the size of the traffic demand is randomly generated between 1 and a maximum traffic demand size. Random traffic pattern is a good approximation of the real life traffic pattern in Metro networks.

We use CPLEX version 7.5 commercial integer linear programming solver to solve all problem instances, running on a Sun Blade 2000 workstation with two UltraSPARC III+ 900Mhz CPUs. All computation times shown are in seconds in CPU time on the Sun workstations.

We first consider random traffic pattern in an 8 nodes network. We randomly generate 10 sets of traffic demands, each having 7 demands ranging from 1 to 2 OC-3 circuits. The results are shown in Table I. Under the ‘‘F1’’ column, we show the total run time used to solve formulation F1 directly. Under the ‘‘LLSF’’ and ‘‘SLSF’’ columns, we show the total run time taken by the LLSF and the SLSF algorithms respectively. In all cases, both the LLSF and the SLSF algorithms take less time to finish the computation; in some cases, the reduction in run time is very significant. The SLSF algorithm seems to outperform the LLSF algorithm in many cases. This is because in general the lower cost solution tends to use more lower line speeds. Because of economy of scale, using higher line speed tends to lower the total cost only if there are enough traffic demands between some node pairs. If we are able to find the optimal or a near optimal solution early, it can be used to prune large solution space in later iterations, resulting in shorter total run time. In our experiments, we notice that the SLSF algorithm uses shorter amount of time in many cases, but still the LLSF algorithm is able to win in some other cases. For brevity, we only show the SLSF result for the remaining experiments.

Next, we consider uniform traffic demands with 5 wavelengths and 3 line speeds. We only show results for 4, 5 and 6 nodes because networks with more nodes take a lot more time to finish. The results are shown in Table II. The run time is greatly reduced. In the case of 6 nodes, the reduction is up to 97%.

When a problem instance is simple, i.e., the number of nodes and the number of line speeds supported are small, it can be solved exactly within a reasonable amount of time. However, when it is complex, the computation time could be very long. In these cases, we can still apply our proposed technique by terminating the computation early. Using our technique, we can explore a larger portion of the solution space within a given amount of time, thus have a chance to get better results.

We consider uniform traffic in a network supporting 3 line

TABLE I

RUN TIME COMPARISON. RANDOM TRAFFIC, 8 NODES, W=10, 3 LINE SPEEDS.

run #	F1	LLSF	SLSF
0	1963.05	151.78	147.28
1	1860.44	506.94	434.47
2	781.3	346.78	190.29
3	6234.51	196.25	67.9
4	1197.02	324.3	142.27
5	391.77	113.89	54.56
6	7375.06	176.42	207.03
7	71.88	69.66	48.77
8	685.91	636.77	117.38
9	1720.09	250.77	106.65

TABLE II

RUN TIME COMPARISON. UNIFORM TRAFFIC, W=5, 3 LINE SPEEDS.

	N=4	N=5	N=6
ILP	12.82	615.73	25988.31
SLSF	1.71	14.6	690.17

speeds and 10 wavelengths. The results are shown in Table III. We fix the computation time when solving formulation F1 directly to a maximum of 10000 seconds, and record the best result the ILP solver is able to find up to that point. With 3 line speeds and 10 wavelengths, the SLSF algorithm needs to examine 66 unique tuples. To keep the total computation time of the SLSF algorithm less than 10000 seconds, we allocate  $10000/66=151.5$  seconds for each tuple. The total time taken for a tuple may be less than 151.5 seconds because a tuple might be solved quickly, or it might be proven to be infeasible quickly. If this happens, we do not reallocate the saved time to other tuples. We sum up the actual time taken for all tuples in the table too under the “time” column for the SLSF algorithm. Under the “opt” column, we also show the best solution we are able to derive manually up to 12 nodes. We believe they are the optimal solutions, however, we can not prove that they are. Using our proposed technique, we consistently achieve the same or better results than those obtained by solving formulation F1 directly using smaller total amount of run time. In most cases, the result is either the “optimal” or very close to it. We suspect that if we can come up with a better time allocation strategy, so that we use up all of the 10000 seconds allocated, we could improve the results even further.

## V. CONCLUSION

In this paper, we generalized the ILP formulation for traffic grooming in WDM/SONET to support multiple line speeds. The formulation is complex and solving it directly could be very computation intensive. Therefore, we proposed a novel technique to improve the computation efficiency. The technique reformulates the problem as a non-linear problem and

TABLE III

COST AND RUN TIME COMPARISON WHEN COMPUTATION IS TERMINATED EARLY. UNIFORM TRAFFIC, W=10, 3 LINE SPEEDS.

# of nodes	F1		SLSF	
	cost	opt cost	cost	time
4	12	12	12	148
5	20	20	20	1674
6	33.5	33.5	33.5	5386
7	49.5	49.5	49.5	8450
8	68	67	67	8661
9	89.5	87.5	88.5	8032
10	125	111.5	111.5	6972
11	152.5	141.25	141.25	6063
12	218.75	171	174.25	4851
13	252.5		215.75	3791
14	346.25		260	2881
15	312.5		305	1973
16	352.5		346.25	1215

decomposes it into several subproblems. The decomposition not only makes the subproblems linear, but also allows us to exploit the symmetry in the problem structure, thereby, reduce the computation time. If a problem instance is too complex to be solved exactly, we can still apply our technique by terminating the computation early. Our technique can explore a larger solution space within a given amount of time, therefore, it can frequently find better solutions than those obtained by solving the ILP formulation directly.

Even though we have made an attempt to solve the traffic grooming problem more efficiently, the problem remains NP-complete. For large size problems, more efficient algorithms, such as heuristics, should be devised.

## REFERENCES

- [1] J.-Q. Hu, “Traffic grooming in wavelength-division-multiplexing ring networks: a linear programming solution,” *J. of Optical Networking*, vol. 1, no. 11, pp. 397–408, Nov. 2002.
- [2] A. Sutter, F. Vanderbeck, and L. Wolsey, “Optimal placement of add/drop multiplexers: Heuristic and exact algorithms,” *Operations Research*, vol. 46, pp. 719–728, 1998.
- [3] O. Gerstel, R. Ramaswami, and G. H. Sasaki, “Cost-effective traffic grooming in WDM rings,” *IEEE/ACM Trans. Networking*, vol. 8, no. 5, pp. 618–630, Oct. 2000.
- [4] O. O. Gerstel, P. Lin, and G. H. Sasaki, “Wavelength assignment in a WDM ring: To minimize cost of embedded SONET rings,” in *Proc. INFOCOM*, 1998, pp. 94–101.
- [5] O. Gerstel, P. Lin, and G. Sasaki, “Combined WDM and SONET network design,” in *Proc. INFOCOM*, 1999.
- [6] J. M. Simmons, E. L. Goldstein, and A. Saleh, “Quantifying the benefit of wavelength add-drop in WDM rings with distance-independent and dependent traffic,” *J. Lightwave Technol.*, vol. 17, no. 1, pp. 48–57, Jan. 1999.
- [7] A. L. Chiu and E. H. Modiano, “Traffic grooming algorithms for reducing electronic multiplexing costs in WDM ring networks,” *J. Lightwave Technol.*, vol. 18, no. 1, pp. 2–12, Jan. 2000.
- [8] X. Y. Li, L. Liu, P. J. Wan, and O. Frieder, “Practical traffic grooming scheme for single-hub SONET/WDM rings,” in *Proc. 25th IEEE Conf. on Local Computer Networks*, Nov. 2000.
- [9] X. Li, P. Wan, and L. Liu, “Select line speeds for single-hub SONET/WDM ring networks,” in *Proc. ICC*, 2000, pp. 495–499.