Grooming of Arbitrary Traffic in SONET/WDM BLSRs

Peng-Jun Wan, Gruia Călinescu, Liwu Liu, and Ophir Frieder

***Abstract*—SONET add–drop multiplexers (ADMs) are the dominant cost factor in the SONET/WDM rings. They can potentially be reduced by optical bypass via optical add–drop multiplexers (OADMs) and traffic grooming. In this paper we study the grooming of arbitrary traffic in WDM bidirectional line-switched rings (BLSRs) so as to minimize the ADM cost. Two versions of the minimum ADM cost problem are addressed. In the first version, each traffic stream has a predetermined routing. In the second version, the routing of each traffic stream is not given in advance; however, each traffic stream is fully duplex with symmetric demands, which must be routed along the same path but in opposite directions. In both versions, we further consider two variants depending on whether a traffic stream is allowed to be split at intermediate nodes. All the four combinations are NP-hard even for any fixed line-speed. General lower bounds on the minimum ADM cost are provided. Our traffic grooming follows a two-phased approach. The problem targeted at in each phase is NP-hard itself, except the second phase when the line speed is two. Various approximation algorithms are proposed in both phases, and their approximation ratios are analyzed.**

***Index Terms*—Approximation ratio, grooming, optical, SONET, WDM.**

I. Introduction

Coupling wavelength division multiplexing (WDM) technology with synchronous optical network (SONET) rings [11] is a very promising network architecture that has attracted much attention recently [7]–[10], [4], [14], [15]. In this network architecture, each WDM channel carries a high-speed (e.g., OC-48) SONET ring. Each SONET ring can further carry a number of low-speed (e.g., OC-3) traffic streams. The number of the low-speed streams that can be carried in a SONET ring is referred to as the *traffic granularity*, denote by a parameter g. The key terminating equipments are optical add–drop multiplexers (OADMs) and SONET add/drop multiplexers (ADMs). Each node is equipped with one OADM. The OADM can selectively drop wavelengths at a node. Thus, if a wavelength does not carry any traffic from or to a particular node, the OADM allows that wavelength to optically bypass that node rather than being electronically terminated. Consequently, in each SONET ring a SONET ADM is required at a node if and only if it carries some traffic terminating at this node. Therefore, the SONET/WDM ring architecture cannot only greatly increase the capacity, thereby reducing the amount of required fiber and allowing for more graceful upgrades, but also potentially reduce the amount of required SONET ADMs. As SONET ADMs typically cost on the order of hundreds of thousands of dollars, a fundamental bandwidth management problem is how to route and groom the traffic demands to minimize the total SONET ADM cost. This problem is referred to as the *minimum ADM cost problem*.

In this paper, we assume that the physical ring network is a bidirectional line-switched ring (BLSR) with either two fiber rings (i.e., BLSR/2) or four fiber rings (BLSR/4) [10]. All wavelength channels are assumed to have the same line-speed, denoted by g. By allowing parallel traffic streams between the same pair of nodes, each traffic steam is assumed to have a *unital* traffic demand. However, each traffic stream can occur between any *arbitrary* pair of nodes, as opposite to the all-to-all traffic [7], [8], [4], [14], [15], [17] and one-to-all traffic [10], [12], [4], [14]. Two versions of the minimum ADM cost problem will be addressed. In the first version, each traffic stream has its predetermined routing, such as short-path routing. Thus we can deal with the two (working) fiber rings separately. In each fiber ring, a traffic stream can be represented as a (directed) circular arc. Depending on the implementation, the arcs may or may not be allowed to be split at intermediate nodes. If the splitting is not allowed, a valid traffic grooming corresponds to a partition of A into groups of g-colorable arcs. If the splitting is allowed, then a valid traffic grooming consists of a choice of splitting each arc of A, thus obtaining B, and then a partition of B into groups of g-colorable arcs. In either case, each group of g-colorable arcs can be carried in a wavelength and thus form a logical SONET ring. This version is referred to as the *arc-version* the minimum ADM cost problem.

In the second version, the routing of each traffic stream is not given in advance. However, each traffic stream is fully duplex with symmetric (unital) demands, which must be routed along the same path but in opposite directions. Thus we can treat the two (working) fiber rings as one (undirected), and each traffic stream as a (undirected) chord. Similarly, depending on the implementation, the chords may or may not be allowed to be split at intermediate nodes. If the splitting is not allowed, a valid traffic routing and grooming corresponds to a partition of A into groups of g-colorable chords (a set of chords is said to be g-colorable if they can be routed as g-colorable arcs). If the splitting is allowed, then a valid traffic routing and grooming consists of a choice of splitting each chord of A, thus obtaining B, and then a partition of B into groups of g-colorable chords. In either case, each group of g-colorable chords can be carried in a wavelength and thus form a logical SONET ring. This version is referred to as the *chord-version* minimum ADM cost problem.

Manuscript received October 15, 1999; revised May 15, 2000.

The authors are with the Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616 USA (e-mails: {wan, calinescu, liwuliu, ophir}@cs.iit.edu).

Publisher Item Identifier S 0733-8716(00)09026-0.

Both versions of the minimum ADM cost problem are NP-hard when $g = 1$ in [12] and [3], respectively. The reductions made in [12] and [3] can be generalized to show the NP-hardness of both versions for any *fixed constant* $g \geq 1$ (for the arc-version without splits, a proof can be found in [16]). Due to hardness of this problem, we adopt the following two-phased approach as did in [16] and [17].

A. Generation of Primitive Rings

In its arc-version, split the arcs as necessary if the splitting is allowed, and partition the resulting arcs or chords into groups of nonoverlapping arcs of chords (a set of arcs are said to be *nonoverlapping* if their interiors have empty intersection; a set of chords are said to be nonoverlapping if they can be routed as nonoverlapping arcs). In its chord version, split the chords as necessary if the splitting is allowed, and partition the resulting chords into groups of nonoverlapping chords (a set of chords are said to be *nonoverlapping* if they can be routed as nonoverlapping arcs). In both versions, each group of nonoverlapping arcs or chords in the partition can be arranged in a single ring, referred to as a *primitive ring*. The cost of each primitive ring is defined as the number of nodes appearing in this primitive ring, and the cost of a partition is defined as the sum of the costs of all primitive rings within this partition. The objective in this phase is then to find a partition with minimum cost.

B. Grooming of Primitive Rings

Partition those primitive rings that are constructed in the first phase into groups of at most g primitive rings. Each group then forms a logical SONET ring. Once again, the ADM cost of each group or SONET ring is the number of nodes contained in this group, and the total ADM cost of a partition is the sum of ADM costs of all groups in this partition. The objective of this phase is thus to find a partition with the minimum total ADM cost.

Notice that the first phase is essentially the minimum ADM cost problem with $g = 1$, and thus is NP-hard itself. Especially, its arc version can be interpreted as the wavelength assignment to lightpaths to minimize ADMs, which was studied in a series of papers including [9], [12], [2], [3]. The second phase can be solved optimally in polynomial time when $g = 2$. However, for any fixed constraint $g > 2$, [1] proves the NP-hardness of restricted ring grooming, a grooming of primitive rings with the constraint that the number of groups of the primitive rings should be the least. The same reduction used in [1] can be used to show the NP-hardness of the unrestricted ring grooming for any fixed constraint $g > 2$. Thus both phases remain challenging.

The remainder of this paper is organized as follows. In Section II, we present two lower bounds on the minimum ADM cost. In Sections III and IV, we provide a number of heuristics for both versions of generation of primitive rings respectively. In Section V, we present approximation algorithms for optimal grooming of primitive rings. Finally we conclude this paper in Section VI.

II. General Lower Bounds

One straightforward but rather loose lower bound on the minimum ADM cost can be obtained in the following way. We begin with the arc version. Let A be the set of input (clockwise) arcs. For any node i, let $\sigma_A(i)$ denote the total number of arcs in A that originate from node i, and $\tau_A(i)$ denote the total number of arcs in A that terminate at node i. Then the node i must use at least

$$\left\lceil \frac{\max(\sigma_A(i), \tau_A(i))}{g} \right\rceil$$

ADMs. Hence, the total ADM cost is at least

$$\sum_{i=0}^{n-1} \left\lceil \frac{\max(\sigma_A(i), \tau_A(i))}{g} \right\rceil .$$

This lower bound is a generalization of the one in given in [9] for $g = 1$.

Now we consider the chord version. Let A be the set of input chords. For any node i, let $\theta_A(i)$ denote the total number of chords in A that contain node i as one endpoint. As each ADM can terminate at most $2g$ chords, node i must use at least $\lceil(\theta_A(i)/2g)\rceil$ ADMs. Hence the total ADM cost is at least

$$\sum_{i=0}^{n-1} \left\lceil \frac{\theta_A(i)}{2g} \right\rceil .$$

In the remaining of this section, we develop another lower bound by calculating the *maximum ADM efficiency*, as defined later. We notice that similar lower bounding technique was used in [4] for only $g = 4, 16$, but no general lower bound was established in [4]. The lower bound derived below is general and applicable to any values of n and g. Given any two positive integers n and g, let $A(n, g)$ denote the maximal number of arcs with *different* endpoint pairs over a ring of n nodes whose load is g, and let

$$E(n, g) = \frac{A(n, g)}{n}.$$

Then the *maximal ADM efficiency* with line-speed g is defined as

$$E(g) = \max_{n \geq 2} E(n, g).$$

Intuitively, $E(g)$ puts an upper bound on the average number of arcs that can be carried by one ADM. Thus, the total number of ADMs required by a set of arcs (or chords) with different endpoint pairs is at least the total number of arcs (or chords respectively) divided by the $E(g)$.

When g is small, $E(g)$ can be calculated easily. For example, when $g = 1$ or 2,

$$\begin{aligned} A(n,1) &= n \\ E(n,1) &= E(1) = 1 \\ A(n,2) &= \begin{cases} \left\lfloor \dfrac{3n}{2} \right\rfloor & \text{if } n \geq 5 \\ \left\lfloor \dfrac{3n}{2} \right\rfloor - 1 & \text{if } n \leq 4 \end{cases} \\ E(2) &= \frac{3}{2}. \end{aligned}$$

In general, the calculation of $E(g)$ relies on the concept of *canonical* set of arcs. A set of arcs with different endpoint pairs is said to be *canonical* if it satisfies the following property: if an arc of length ℓ is in this set, then all arcs of length less than ℓ are also in this set. It is easy to show that any set of arcs over a ring can be transformed to a canonical set of arcs of the *same* cardinality over the same ring with the same or less load. Thus there is always a canonical set of arcs over a ring of n nodes whose load is g. A canonical set of arcs can be generated in the following greedy manner: we first add all arcs of length one, then we add all arcs of length two, and so on until we cannot add all arcs of some length; in this case, we add as many arcs of such length as possible. The following lemma gives the load of the set of arcs of length no more than k.

Lemma 1: For any $\ell \leq n-1$, the load of all arcs of length no more than ℓ over a ring of size n is $(\ell(\ell+1)/2)$.

Proof: For any $k \leq n-1$, there are n arcs of length k. These n arcs of length k contribute a load of k to each link. Thus the load contributed to each link by those $n\ell$ arcs of length at most ℓ is

$$1+2+\cdots+\ell = \frac{\ell(\ell+1)}{2}.$$

■

Let ℓ be the largest integer satisfying that $(\ell(\ell+1)/2) \leq g$. If $n \leq \ell+1$, then the load of all $n(n-1)$ arcs over an n-node ring is no more than g, and thereby

$$\begin{aligned} A(n,g) &= n(n-1) \\ E(n,g) &= n-1 \leq \ell \leq \frac{g}{\ell+1} + \frac{\ell}{2}. \end{aligned}$$

Now we assume that $n \geq \ell+2$. If $g = (\ell(\ell+1)/2)$, then the load of all $n\ell$ arcs of length at most ℓ over an n-node ring is exactly g, and thereby

$$\begin{aligned} A(n,g) &= n\ell \\ E(n,g) &= \ell = \frac{g}{\ell+1} + \frac{\ell}{2}. \end{aligned}$$

If $g > (\ell(\ell+1)/2)$, then $A(n,g) - n\ell$ is equal to the maximal number of arcs of length $\ell+1$ which contribute a load of no more than $g - (\ell(\ell+1)/2)$ to each link. The cumulative link load contributed by these arcs of length $\ell+1$ is at most $n(g - (\ell(\ell+1)/2))$. As each arc of length $\ell+1$ contributes a unit load to $\ell+1$ links, the total number of such arcs is

$$A(n,g) - n\ell \leq \frac{n\left(g - \dfrac{\ell(\ell+1)}{2}\right)}{\ell+1} = n\left(\frac{g}{\ell+1} - \frac{\ell}{2}\right).$$

On the other hand, it is obvious that

$$A(n,g) - n\ell \geq \left\lfloor \frac{n}{\ell+1} \right\rfloor \left(g - \frac{\ell(\ell+1)}{2}\right).$$

Hence,

$$\begin{aligned} n\ell + \left\lfloor \frac{n}{\ell+1} \right\rfloor \left(g - \frac{\ell(\ell+1)}{2}\right) &\leq A(n,g) \leq n\left(\frac{g}{\ell+1} + \frac{\ell}{2}\right) \\ \ell + \frac{1}{n}\left\lfloor \frac{n}{\ell+1} \right\rfloor \left(g - \frac{\ell(\ell+1)}{2}\right) &\leq E(n,g) \leq \frac{g}{\ell+1} + \frac{\ell}{2}. \end{aligned}$$

TABLE I
MAXIMAL ADM EFFICIENCY WITH VARIOUS LINK CAPACITIES

g	2	4	8	16
ℓ	1	2	3	5
$E(g)$	$\frac{3}{2}$	$\frac{7}{3}$	$\frac{7}{2}$	$\frac{31}{6}$

In particular, when n is a multiple of $\ell+1$,

$$\begin{aligned} A(n,g) &= n\left(\frac{g}{\ell+1} + \frac{\ell}{2}\right) \\ E(n,g) &= \frac{g}{\ell+1} + \frac{\ell}{2}. \end{aligned}$$

Therefore,

$$E(g) = \frac{g}{\ell+1} + \frac{\ell}{2}.$$

The next lemma summarizes the above discussion.

Lemma 2: For any positive integer g, the maximal node efficiency is $(g/\ell+1) + (\ell/2)$ where ℓ is the largest integer satisfying that $(\ell(\ell+1)/2) \leq g$.

Table I lists the values of $E(g)$ when $g = 2$, 4, 8, and 16.

From Lemma 2, we have the following lower bound on the minimal ADMs required.

Lemma 3: Let g be the line-speed and A be any set of arcs with different endpoint pairs in the arc version (or chords with different endpoint pairs in the chord version). If the splitting is not allowed, then the minimum ADM cost of A is at least $\lceil (|A|/(g/\ell+1) + (\ell/2)) \rceil$, where ℓ is the largest integer satisfying that $(\ell(\ell+1)/2) \leq g$.

III. GENERATION OF PRIMITIVE RINGS FROM ARCS

Assume that the ring network consists of n nodes numbered clockwise by $0, 1, \ldots, n-1$. An (clockwise) arc a is represented by $(o(a), t(a))$, where $o(a)$ is the origin of a and $t(a)$ is the termination of a. The *normalized length*, or length in short, of an arc a is defined as the number of links in a divided by n. Let A be any set of arcs. The (normalized) length of A, denoted by $l(A)$, is defined as the sum of the (normalized) lengths of the arcs in A. For any node i, the difference between $\sigma_A(i)$ and $\tau_A(i)$, denoted by $\delta_A(i)$, is referred to as the *surplus* of a node i with respect to A. The *deficiency* of a node i with respect to A is $d_A(i) = 1/2|\delta_A(i)|$. The deficiency of A, denoted by $d(A)$, is defined as the sum of the deficiencies of all nodes with respect to A, i.e., $d(A) = \sum_{i=0}^{n-1} d_A(i)$. In all these notations, the subscripts may be omitted if understood from the context.

A sequence of arcs is called as a *chain* if the termination of each circular arc (except the last one) is the origin of the subsequent circular arc. A chain is said to be *closed* if the termination of the last circular arc is also the origin of the first circular arc, or *open* otherwise. The origin and termination of an open chain C, denoted by $o(C)$ and $t(C)$, respectively, are defined as the origin of the first arc and the termination of the last arc, respectively. An open chain C in a set of arcs A is said to be *tight* with respect to A if $o(C)$ has a negative surplus with respect to A and $t(C)$ has a positive surplus with respect to A. For each chain C, the *cost* of C is the number of nodes in C; the *size* of C, denoted by $|C|$, is the number of arcs in C. A chain of size k is called as a k-chain. A chain is said to be an *odd* (or *even*) chain if its size is odd (or even respectively). If the arcs in a chain do not overlap with each other, then the chain is said to be *valid*; otherwise it is said to be *invalid*.

Any primitive ring generation, with or without splits, induces naturally a set of disjoint valid chains. The cost of a primitive ring generation is simply the sum of the costs of all these valid chains or, equivalently, the sum of the sizes of all valid chains plus the number of open valid chains. In particular, if all arcs are not splittable, the cost of any primitive ring generation equals the total number of arcs plus the number of open valid chains; therefore, an optimal primitive ring generation corresponds to the one having minimal number of open valid chains. Based on this observation, a set of primitive rings can be generated in two stages. In the first stage, called as *valid chain generation*, the arcs are split if splitting is allowed and then grouped into valid chains; in the second stage, called as *valid chain coloring,* these valid chains are grouped into minimum number of primitive rings. The second stage only affects the number of primitive rings, but has no impact on the total cost of these primitive rings. As the second stage is the well-studied circular-arc coloring problem, we focus only on the first stage.

In general, the optimal primitive ring generation, with or without splits, is NP-hard as it is essentially the minimum ADM problem when $g = 1$. So is the optimal valid chain generation. In addition, from the discussion in Section II, a lower bound on the minimum cost of a set of arcs A is

$$\sum_{i=0}^{n-1} \max\{\sigma_A(i), \tau_A(i)\} = |A| + d(A).$$

This lower bound can also follow from the fact that the total sizes of all valid chains is at least $|A|$ and the number of open valid chains is at least $d(A)$. In this section, we will present five approximation algorithms for optimal valid chain generation without splits in Subsection A, and one approximation algorithm for optimal valid chain generation with splits in Section III-B. The performance ratios of all these approximation algorithms are analyzed.

A. Unsplittable Arcs

In this subsection, we assume that all arcs are not allowed to be split. A valid chain generation is said to be *maximal* if no two valid chains can be merged to a larger valid chain, i.e., any pair of valid chains are either disjoint or overlapping. The following lemma provides a *coarse* analysis of the cost of any maximal valid chain generation.

Lemma 4: The cost of any maximal valid chain generationis within 7/4 times of the minimum cost.

Proof: Let OPT be any optimal valid chain generation, and $\mathcal{C}$ be any maximal valid chain generation. Assume that there are k odd open (valid) chains in OPT. Then the optimum cost is $|A| + k$. We call an arc *unmatched* in a valid chain generation if it forms a (valid) chain by itself alone. Then out of any two consecutive arcs in any (valid) chain of $OPT,$ at most one is unmatched in $\mathcal{C}$. Let C be any (valid) chain in OPT. If C is closed, then at most $\lfloor |C|/2 \rfloor$ arcs in C are unmatched in $\mathcal{C}$. If C is open, then at most $\lceil |C|/2 \rceil$ arcs in C are unmatched in $\mathcal{C}$. Suppose that there are i unmatched arcs in $\mathcal{C}$. Then $i \leq (|A| + k/2)$. These i unmatched arcs form i open (valid) 1-chains in $\mathcal{C}$. As the sizes of the other (valid) chains in $\mathcal{C}$ are all at least two, the total number of (valid) chains in $\mathcal{C}$ is at most

$$i + \left\lfloor \frac{|A| - i}{2} \right\rfloor \leq \frac{|A| + i}{2} \leq \frac{|A| + \frac{|A| + k}{2}}{2} = \frac{3}{4}|A| + \frac{k}{4}.$$

Thus, the cost of $\mathcal{C}$ is at most

$$|A| + \left(\frac{3}{4}|A| + \frac{k}{4}\right) = \frac{7}{4}|A| + \frac{k}{4} \leq \frac{7}{4}(|A| + k). \qquad \blacksquare$$

In the following, we will present five maximal valid chain generations.

1) Assign First—Revisited: If all arcs in A do not cross over some link, say the link from $n-1$ to 0, then the arcs form an interval graph and the optimal valid chain generation of such instance can be found in polynomial time. A simple greedy algorithm given in [9] works as follows. The arcs are sorted according to their source endpoints. We then consider each circular arc one by one in this order. For each circular arc, if it can be merged with some existing valid chain to form a larger valid chain, then merge them; otherwise we create a new valid chain consisting of only this circular arc. Obviously, the cost of the resulting (valid) chains is $\sum_{i=0}^{n-1} \max\{\sigma_A(i), \tau_A(i)\}$ and therefore the greedy algorithm is optimal.

The *assign first* heuristic presented in [9] initially puts each circular arc that passes through a carefully selected link into a unique 1-chain. The remaining arcs, which form an interval graph, are then greedily grouped into valid chains as above. This heuristic cannot generate a valid solution in general [13]. In the following, we present a modified *assign-first*. Consider any $0 \leq i < n$. Let A_i denote the set of arcs passing through the link i. We first greedily generate valid chains out of the arcs not in A_i. We then construct a weighted bipartite graph G_i over A_i and the obtained chains as follows: there is an edge between an arc in A_i and an obtained chain if and only if they do not overlap and share at least one endpoint; the number of shared endpoints is set to the weight of the edge. We find a maximum-weighted matching in G_i. For each edge in the matching, we merge the arc and the chain corresponding to the two vertices of the edge into a larger (valid) chain. The arcs in A_i that are not in the matching then each form a unique 1-chain. Let α_i denote the cost of the resulting (valid) chains. We repeat the above procedure over all $0 \leq i < n$ and output

$$\alpha = \min\{\alpha_i : 0 \leq i < n\}.$$

It is obvious that

$$\alpha_i \leq \sum_{i=0}^{n-1} \max\{\sigma_A(i), \tau_A(i)\} + 2|A_i|.$$

Thus,

$$\alpha \leq \sum_{i=0}^{n-1} \max\{\sigma_A(i), \tau_A(i)\} + 2\min\{|A_i| : 0 \leq i < n\}.$$

Therefore, the algorithm uses no more than twice the minimum link load than the optimum. On the other hand, this algorithm is a maximal valid chain generation, and thus its approximation ratio is at most 7/4 from Lemma 4. However, its exact approximation ratio is still unknown.

2) Iterative Merging: Initially each valid chain consists of one circular arc. At each iteration, one of the following three possible operations is performed in decreasing priority.

Operation 1: Merge two open (valid) chains into a closed valid chain.

Operation 2: Split an open (valid) chain into two open (valid) chains and then merge one of them with another open (valid) chain into a closed valid chain.

Operation 3: Merge two open (valid) chains into a larger open valid chain.

Operation 1 decreases the number of open valid chains by two, and Operation 2 and Operation 3 both decrease the number of open valid chains by one. Thus, the total number of iterations before the algorithm terminates is less than the total number of arcs. The algorithm is a maximal valid chain generation. Thus, its approximation ratio is at most 7/4 from Lemma 4. In the next, we will use a bad example to show that the approximation ratio of the algorithm *iterative merging* is at least 3/2.

Example 5: Let $n = 5$ and $A = A_1 \cup A_2$ where

$$A_1 = \{(0,1),(1,3),(3,0)\}$$
$$A_2 = \{(0,2),(2,4),(4,0)\}.$$

Note that the three arcs in A_1 form a closed valid chain, so do the three arcs in A_2. Thus, the minimum cost is $|A| = 6$. On the other hand, the algorithm may output the following three open valid chains:

$$\{(4,0),(0,1)\},\{(0,2),(2,4)\},\{(1,3),(3,4)\}.$$

The cost of these three chains is 9, which is 3/2 times of the minimum cost. This example can be scaled to the rings whose sizes are multiple of five. Thus, the approximation ratio of the algorithm *iterative merging* is at least 3/2.

In summary, we have the following theorem.

Theorem 6: The approximation ratio of the algorithm *iterative merging* is between 3/2 and 7/4.

3) Iterative Matching: Initially each valid chain consists of one circular arc. At any iteration, we construct a weighted graph over the current set of (valid) chains as follows. There is an edge between two (valid) chains if and only they do not overlap but share at least one endpoint. The weight of an edge is the number of endpoints shared by the two (valid) chains incident to this edge. We then find a maximum weighted matching in the graph. The two (valid) chains incident to each edge in the obtained matching are then merged into a larger (valid) chain. This procedure is repeated until no matching can be found any more.

It is obvious that the algorithm has polynomial run-time. The algorithm is a maximal valid chain generation, and thus has an approximation ratio of at most 7/4 from Lemma 4. A more complicated analysis in [2] shows that the approximation ratio is at most 5/3. On the other hand, the same instance in Example 5 leads to a 3/2 lower bound on the approximation ratio of the algorithm *iterative matching*. In fact, the first iteration may create the following matching over A:

$$\{(4,0),(0,1)\},\{(0,2),(2,4)\},\{(1,3),(3,4)\}.$$

It is easy to verify that the chains induced by such matching overlap with each other and thus the algorithm stops. The cost of these three chains is 9, which is 3/2 times of the minimum cost. Therefore, we have the following theorem.

Theorem 7: The approximation ratio of the algorithm *iterative matching* is between 3/2 and 5/3.

4) Minimum-Weighted Cycle Cover: We call a pair of arcs *complementary* if they form a closed valid chain. Suppose there are two complementary arcs in A. Then it is routine to verify that there is an optimal solution in which these two complementary arcs form a closed chain. Thus we can find the maximal pair of complementary arcs in A, and form a closed valid chain from each pair of complementary arcs. From now on, we assume that no pairs of arcs in A are complementary.

We first construct a weighted directed graph $G(A)$ over A as follows. The vertex set is A. For any pair of nonoverlapping arcs a_1 and a_2, add one link from a_1 to a_2 and one link from a_2 to a_1. If a_1 and a_2 do not share any endpoints, the weights of both links are set to two. If the termination endpoint of $t(a_1) = o(a_2)$, the weight of the link from a_1 to a_2 is set to one and the weight of the link from a_2 to a_1 is set to two. If $t(a_2) = o(a_1)$, the weight of the link from a_2 to a_1 is set to one and the weight of the link from a_1 to a_2 is set to two. In addition, there is one loop link with weight two at each arc. We then find a minimum-weighted cycle cover of $G(A)$. Note that any valid chain generation induces naturally a cycle cover whose weight is equal to the cost of the valid chain generation. Thus the weight of the obtained cycle cover is a lower bound of the minimum cost.

From each cycle in the minimum-weighted cycle cover, we remove all links of weight two and obtain a collection of paths. Each path induces a chain of the original arcs. We split those invalid chains into valid chains. An invalid open chain can be split into valid chains by walking along the chain from the origin of the chain and generating a valid chain whenever there is an overlap. If the invalid chain is closed, a splitting is conducted by choosing each node in the chain as the starting point, and then the best one is selected to split the chain.

It is obvious that the above algorithm has polynomial runtime. The algorithm produces a set of valid chains, and thus its approximation ratio is at most 7/4 from Lemma 4. A tighter analysis in [2] shows that the approximation ratio is at most 1.6. Thus we have the following theorem on the performance of the above algorithm.

Theorem 8: The approximation ratio of the algorithm based on minimum weighted cycle cover is at most 1.6.

5) Closed ChainFirst: In this subsection we present yet another greedy algorithm, called *closed chain first* (CCF). The algorithm consists of two phases. In the first phase, the algorithm repeatedly obtains closed valid chains until no valid closed chains are left. In the second phase, a maximum matching algorithm is used iteratively to reduce the number of valid open chains.

The first phase applies the following procedure which outputs a closed valid chain containing a specified arc, if there is any, from a set of arcs S. Let a be any chord in S. We build a directed acyclic graph (dag) that consists of only those arcs in S that do not overlap with a. Obviously there is a path from $t(a)$ to $o(a)$ in the dag if and only if there is a closed valid chain in S that contains a. By using *breadth-first search* in the dag, we can obtain a path, if there is any, from the from $t(a)$ to $o(a)$. Once this path is obtained, we merge it with a to obtain a closed valid chain.

In the first phase, we start with $S = A$ (A is the initial set of arcs). For every arc $c \in S$, the procedure from the previous paragraph is applied to determine if there is a valid closed chain containing a. If a valid closed chain is found, it is output as part of the solution, its arcs are removed from S, and we iterate. If there is no valid closed chain with arcs from S, we proceed to the second phase.

The second phase applies maximum matching algorithm iteratively to group the remaining arcs into open valid chains. Initially each open valid chain consists of one red circular arc. At any iterative step, we construct a graph over the current set of open (valid) chains as follows. There is an edge between two open (valid) chains if and only there do not overlap and share one endpoint (note that they can share at most one endpoint). We then find the maximum matching in the resulting graph. The two open (valid) chains incident to each edge in the obtained maximum matching are then merged into a larger open (valid) chain. This procedure is repeated until no matching can be found any more. Then each remaining circular arc forms an open chain by itself.

The algorithm also has polynomial run-time. The algorithm produces a set of maximal valid chains, and thus has an approximation ratio of at most 7/4 from Lemma 4. A sophisticated analysis in [2] shows that the approximation ratio is at most 3/2. The following example shows that the approximation ratio of the proposed algorithm is at least 4/3.

Example 9: Let $n = 6$, and $A = A_1 \cup A_2 \cup A_3$ where

$$\begin{aligned} A_1 &= \{(0,2),(2,5),(5,0)\} \\ A_2 &= \{(0,3),(3,4),(4,0)\} \\ A_3 &= \{(1,2),(2,4),(4,1)\}. \end{aligned}$$

Note that for any $1 \le i \le 3$, the three arcs in A_i form a closed valid chain. Thus, the minimum cost is $opt = |A| = 9$. The algorithm we proposed, if unlucky, chooses the closed valid chain

$$\{(0,2),(2,4),(4,0)\}.$$

The remaining 6 arcs do not contain a closed valid chain. The iterative matching generates three open valid chains

$$\{(2,5),(5,0)\},\{(0,3),(3,4)\},\{(1,2),(4,1)\}.$$

Thus, the total ADM cost of all these valid chains is $12 = (4/3) \cdot$ *opt*. This example can be scaled to the rings whose sizes are multiples of six.

In summary, we have the following theorem.

Theorem 10: The approximation ratio of the algorithm *closed chain first* is between 4/3 and 3/2.

B. Splittable Arcs

In this subsection, we assume that each arc is allowed to be split. We call each input circular arc an *original* arc, and an arc resulting from splitting an original arc a *fragment*. A chain is said to be *unsplit* if all arcs in it are original. Obviously, the optimum with splits is at most the optimum without splits. Actually, the optimum with splits can be 25% lower, as shown by the following example.

Example 11: Let $n = 5$ and $A = \{a_1, a_2, a_3, a_4, a_5\}$, with $a_1 = (0,3)$, $a_2 = (3,1)$, $a_3 = (1,4)$, $a_4 = (4,2)$, and $a_5 = (2,0)$. The optimum without splits is 8, while with splits a solution of cost 6 can be obtained by splitting a_3 into a_3' and a_3'' at node 0 with $t(a_3') = o(a_3'') = 0$, and then a_1, a_2, a_3' form one valid chain and a_3'', a_4, a_5 form another valid chain.

Not only can splits improve the optimum, but with splits we can approximate the optimum better. Indeed, we will present a polynomial algorithm with splits that has approximation ratio of at most 5/4. We start by describing a procedure called *Eulerian rounding*. Let S be a set of arcs. The Eulerian rounding first adds a set of $d(S)$ fake arcs F such that $d(S \cup F) = 0$. This can be easily done by adding one-by-one fake arcs with the origin being a node of positive surplus and the termination being a node of negative surplus, thus each fake arc decreasing the deficiency by one. Now the directed graph with edges $S \cup F$ is Eulerian. Choosing any Eulerian tour and then removing all fake arcs results in $d(S)$ open chains. Every invalid open chain C is then split at its origin into $\lfloor l(C) \rfloor - 1$ closed valid chains and one open valid chain. Every invalid closed chain C is then split into $l(C)$ valid closed chains.

In general, Eulerian rounding cannot guarantee a good performance ratio if the arcs have large lengths. To make the Eulerian rounding effective, we preprocess the input arcs by forming valid chains out of arcs as long as possible. Let A be the set of input arcs. Note that if two arcs are complementary, then it is easy to verify that there is an optimal solution in which they form a closed valid chain. Therefore, we assume that A does not contain any pair of complementary arcs. We propose the following algorithm with splits.

- The input is a set of arcs A. Initialize $i \leftarrow 1$.
- Phase 1: While A contains a closed valid 3-chain C, unsplit the arcs in C, then set $C_i \leftarrow C$, $A \leftarrow A - C$, and $i \leftarrow i + 1$.
- Phase 2: While A contains *tight* valid 1-chain C of length at least 1/2, find the longest one. Unsplit the arc in C, then set $C_i \leftarrow C$, $A \leftarrow A - C$, and $i \leftarrow i + 1$.
- Phase 3: While A contains *tight* valid 2-chain C, find the longest one. Unsplit the arcs in C, then set $C_i \leftarrow C$, $A \leftarrow A - C$, and $i \leftarrow i + 1$.
- Phase 4: Do the Eulerian rounding of A.

It is shown in [2] that the approximation ratio of the above algorithm is at most 5/4. The same instance in Example 9 shows that, even after the practical improvements above, the approximation ratio of the algorithm is at least 10/9. Indeed, the optimum has cost 9. If our algorithm, if unlucky, chooses the closed chain

$$\{(0,2),(2,4),(4,0)\}$$

in phase two, it produces a solution of cost 10.

Theorem 12: The performance ratio of algorithm above is between 10/9 and 5/4.

IV. Generation of Primitive Rings from Chords

Assume that the WDM self-healing ring consists of n nodes numbered clockwise by $0, 1, \ldots, n-1$. For any chord c, let c^- be the circular arc between the two endpoints of c that passes through the link between node $n-1$ and node 0, and let c^+ be the arc complementary to c^-. We call c^- and c^+ the counterclockwise and clockwise orientations of c, respectively.

As the optimal primitive ring generation from chords is NP-hard, we will develop two 1.5-approximation algorithms with or without splits, respectively.

A. Unsplittable Chords

In this section, we assume that all chords are unsplittable. The algorithm we will propose is a modification of the arc-version *closed chain first* (CCF). The chord-version CCF algorithm is also a greedy two-phased algorithm. In the first phase, the algorithm repeatedly obtains closed valid chains until no valid closed chains are left. In the second phase, a maximum matching algorithm is used iteratively to reduce the number of valid open chains.

The first phase applies the following procedure which outputs a closed valid chain containing a specified chord, if there is any, from a set of chords S. Let c be any chord in S. Let S_c^+ (S_c^-, respectively) be the set of chords in $S-\{c\}$ whose two endpoints are both in c^+ (c^-, respectively). Let G_c^+ (G_c^-, respectively) be the directed graph with the nodes in c^+ (c^-, respectively) as its vertices and directed edges obtained from orienting the chords in S_c^+ clockwise (orienting the chords in S_c^- counterclockwise, respectively). There is a closed valid chain in S that contains c^- if and only if there is a path between the two endpoints of c in G_c^+. Similarly, there is a closed valid chain in S that contains c^+ if and only if there is a path between the two endpoints of c in G_c^-. After constructing G_c^+ and G_c^-, such a path, if there is any, can be found by breadth-first search. Once this path is obtained, we add c to it to obtain a closed valid chain.

In the first phase, we start with $S=A$ (A is the initial set of chords). For every arc $c \in S$, the procedure from the previous paragraph is applied to determine if there is a valid closed chain containing c. If a valid closed chain is found, it is output as part of the solution, its chords are removed from S, and we iterate. If there is no valid closed chain with chords from S, we proceed to the second phase.

The second phase applies maximum matching algorithm iteratively to group the remaining chords into open valid chains. Initially the set of open valid chains is *empty*. At each iteration, we construct a graph over the obtained valid open chains and the remaining chords as follows.

- There is an edge between two open chains if and only if they do not overlap with each other and share one endpoint.
- There is an edge between two chords if and only if they share one endpoint.
- There is an edge between an open chain and a chord if and only if one endpoint of the chord is the origin or termination of the chain and the other endpoint is outside the chain.

We then find the maximum matching in the graph. For each edge in the obtained maximum matching, we do the following processing.

- If the edge is between two open chains, merge these two chains to obtain a larger valid open chain.
- If the edge is between two chords, orient these two chords in the unique way to form a valid open chain.
- If the edge is between an open chain and a chord, orient the chord in the unique way and merge the resulting arc with the open chain to form a larger valid open chain.

This iteration is repeated until no matching can be found any more. Then each remaining chord forms an open chain by itself.

By using the similar argument to that to Theorem 10, we can prove the following performance of the above algorithm.

Theorem 13: The approximation ratio of the chord-version *closed chain first* is between 4/3 and 3/2.

B. Splittable Chords

Let A be a set of input chords. The *degree* of a node i is the number of chords in A that contain node i as one endpoint. Then the number of nodes with odd degree is even. Our algorithm is the chord-version Eulerian rounding, which is described as follows.

- *Step 1*: Divide the set of nodes with odd degree into disjoint pairs. We then add one *fake* chord between the two nodes in each pair. Let F be the set of fake chords. Then the undirected graph with edges $A \cup F$ is Eulerian.
- *Step 2*: Choose any Eulerian tour in this undirected graph.
- *Step 3*: This Eulerian tour can be oriented in two opposite directions, and we choose the one which has shorter total length of nonfake arcs and break the ties arbitrarily. If there is no fake arc, go to Step 6.
- *Step 4*: Remove all fake arcs from the oriented Eulerian tour to get (open) chains.
- *Step 5*: For every invalid (open) chain C output by Step 4, split it into valid chains as follows: for each circular arc a in C that passes through $o(C)$, the origin of P, split it into two arcs
$$a'=(o(a),o(C)), a''=(o(C),t(a)).$$
After these splittings, the invalid chain P is then decomposed into valid chains by walking along C from $o(C)$, and output a valid chain whenever reaching $o(C)$. Stop the algorithm.
- *Step 6*: Find a node i through which the smallest number of arcs in the oriented Eulerian tour pass. Break the ties arbitrarily.
- *Step 7*: For any circular arc a in the oriented Eulerian tour that passes through i, split it into two arcs
$$a'=(o(a),i), a''=(i,t(a)).$$
After these splittings, the oriented Eulerian tour is then decomposed into valid (closed) chains by walking along the oriented Eulerian tour from node i and output a valid (closed) chain whenever reaching node i. Stop the algorithm.

It is obvious that the algorithm has polynomial run-time and generates a set of valid chains. It is shown in [3] that the approximation ratio of the above algorithm is at most 3/2. The following example shows that the approximation ratio of the algorithm presented in this section is at least 3/2.

Example 14: Let n be odd. Consider the following set of chords:

$$C = \{c_0, c_1, \ldots, c_{n-3}, c', c''\}$$

where for $0 \leq i \leq n-3$, the endpoints of c_i are the nodes i and $i+2$, the endpoints of c' are 0 and 1, and the endpoints of c'' are $n-2$ and $n-1$.

The Eulerian tour selected by the algorithm is

$$c_{n-3}, c_{n-5}, \ldots, c_0, c', c_1, c_3, \ldots, c_{n-4}, c''.$$

The chords $c', c_1, c_3, \ldots, c_{n-4}, c''$ are oriented clockwise, and the chords $c_{n-3}, c_{n-5}, \ldots, c_0$ are oriented counterclockwise. In Step 6, node 0 is selected. Then all the arcs $c_{n-3}, c_{n-5}, \ldots, c_2$ are going to be split in Step 7, obtaining a solution of cost

$$n + \frac{n-3}{2} = \frac{3(n-1)}{2}$$

since there are n chords and $n-(3/2)$ of them are split.

However, an optimum of cost $n+2$ exists: orient all the chords clockwise and produce two open chains: $c_0, c_2, \ldots, c_{n-3}$ and $c', c_1, c_3, \ldots, c_{n-4}, c''$. Therefore, the cost ratio of the output of the algorithm to the optimum is $(3/2) \cdot (n - 1/n + 2)$. If we let n large enough, this ratio gets arbitrarily close to 3/2.

In conclusion, we have the following theorem.

Theorem 15: The approximation ratio of the chord-version Eulerian rounding is exactly 3/2.

V. Grooming of Primitive Rings

An instance of the ring grooming is the grooming granularity g, and a collection of primitive rings represented by a collection of sets $A_1, A_2, \ldots, A_m$ from the universe $\{0, 1, \ldots, n-1\}$. A solution is a partition of the collection of primitive rings (or sets) into groups of size at most g. A group of size k is referred to as a k-group. If all groups in a partition are k-groups, the partition is referred to as a k-grouping. For each group P, the ADM cost of P is defined by $|\bigcup_{A\in P} A|$, and the ADM savings of P is defined by $\sum_{A\in P}|A| - |\bigcup_{A\in P} A|$. The total ADM cost (savings) of a grooming is thus the sum of the costs (savings) of the all groups. The objective is to find a grooming with minimal total ADM costs. Note that a grooming with minimal total ADM costs must have maximal total ADM savings, and vice versa.

While a ring grooming should have as low ADM cost as possible, it is also desirable for a ring grooming to partition the primitve rings to as few groups as possible so as to minimize the wavelength requirement. A ring grooming is said to be *maximal* if no two groups can be merged into a larger group. One can always convert a ring grooming into a maximal ring grooming of the same or less cost by repeatedly merging any two groups whose total size is at most g into a larger group. In addition, in any maximal ring grooming, at most one group contains $\lfloor g/2 \rfloor$ or less primitive rings, and thus the number of groups is at most $1 + \lfloor (m/\lceil g/2 \rceil) \rfloor \leq 2\lceil m/g \rceil$. This implies that there always exists an optimal ring grooming in which the number of groups is at most twice the least ($\lceil m/g \rceil$).

Obviously, any ring grooming must contain at least $\lceil m/g \rceil$ groups. A ring grooming is said to be *restricted* if it contains exactly $\lceil m/g \rceil$ groups. Any ring grooming can be converted to a restricted ring grooming of at most twice the cost as follows. Consider a ring grooming consisting of ℓ groups of sizes $k_1, k_2, \ldots, k_\ell$, respectively. Reorder the indices $1, 2, \ldots, m$ such that for each $1 \leq i \leq \ell$, the ith group consists of k_i primitive rings

$$\left\{A_{\sum_{j=1}^{i-1} k_j + 1}, A_{\sum_{j=1}^{i-1} k_j + 2}, \ldots, A_{\sum_{j=1}^{i} k_j + 1}\right\}.$$

Now construct a restricted grooming in which the ith group consists of g primitive rings

$$\{A_{g(i-1)+1}, A_{g(i-1)+2}, \ldots, A_{gi}\}$$

for $1 \leq i \leq \lfloor m/g \rfloor$, and the remaining primitive rings, if there is any, form another group. Thus, each group of the original ring grooming is either entirely contained in a group of the restricted ring grooming, or split into two subgroups which are contained in two different groups of the restricted ring grooming. Therefore, the cost of the restricted ring grooming is at most twice the cost of the original ring grooming.

In general, the minimum ADM cost may not be achieved by any restricted ring grooming. However, when $g = 2$, such minimum ADM cost can be achieved by a restricted ring grooming. Furthermore, such restricted ring grooming can be found in polynomial time by a reduction to maximum-weighted perfect matching. The reduction relies on the concept of *intersection graph* of a grouping Π. For any collection of sets Π, its *intersection graph*, denoted by $G(\Pi)$, is a weighted graph constructed as follows: the vertex set is Π; an edge exists between two groups P and Q if and only if $|P| + |Q| \leq g$; the weight of each edge (P, Q) is equal to $|(\bigcup_{A\in P} A) \cap (\bigcup_{A\in Q} A)|$. For simplicity, a (perfect) matching of $G(\Pi)$ is also simply called as a (perfect) matching of Π. Let Π_0 be the 1-grouping of the input the input primitive rings. Then an optimal solution for $g = 2$ is to find a maximum-weighted (perfect) matching of Π_0 and then groom every matched pair of 1-groups into a 2-group. This solution achieves not only the minimal total ADM cost, but also the least number of groups ($\lceil m/2 \rceil$).

When $g > 2$, both ring grooming and restricted ring grooming are NP-hard. In the following, we assume that $m \bmod g = 0$, by adding dummy empty sets if necessary. When g is a power of two, we propose the following algorithm called *iterative matching* for optimal restricted ring grooming. It consists of $\log g$ iterations. Let Π_0 be the original sets. The ith iteration starts with Π_{i-1}, a 2^{i-1}-grouping of Π_0, and finds a maximum-weighted *perfect* matching of Π_i. Then for each edge in the obtained matching, the two sets incident to the edge are merged. Thus the ith iteration outputs a 2^i-grouping of Π_0, denoted by Π_i.

The above algorithm has polynomial run-time. A trivial upper bound on its approximation ratio is $g/2$, which can be proved as follows. For any $1 \leq i \leq \log g$, let S_i denote the weight of the maximum-weighted perfect matching of Π_i. Then it's easy to verify that the total savings of the 2^i-grouping Π_i is $\sum_{j=1}^{i} S_j$ for any $1 \leq i \leq \log g$. In particular, the total savings of the g-grooming $\Pi_{\log g}$ output by the iterative matching is $\sum_{j=1}^{\log g} S_j$. Thus we also refer S_i to as the ADM savings at the ith iteration. We fix an optimal restricted ring grooming in which the kth group is $\{A_{k_i} | 1 \leq i \leq g\}$ for any $1 \leq k \leq$

(m/g). Let opt_g denote the minimum cost. Then the cost of the iterative matching is

$$opt_2 - \sum_{j=2}^{\log g} S_j \leq opt_2 \leq \sum_{k=1}^{m/g} \sum_{j=1}^{g/2} |A_{k_{2j-1}} \cup A_{k_{2j-1}}|$$
$$= \frac{g}{2} \cdot \sum_{k=1}^{m/g} \left| \bigcup_{i=1}^{g} A_{k_i} \right| = \frac{g}{2} \cdot opt_g.$$

When $g = 4$ and 8, a very sophisticated analysis in [1] showed tighter bounds on the approximation ratios of the *iterative matching*.

Theorem 16: The approximation ratio of the iterative matching for restricted ring grooming is exactly 1.5 when $g = 4$ and at most 2.5 when $g = 8$.

When g is greater than 8, [1] made the following conjecture on the approximation ratio of the *iterative matching*.

Conjecture 17: For any $g = 2^k$ with $k \geq 4$, the approximation ratio of the iterative matching for restricted ring grooming is at most $(g/4) + (1/2)$.

The *iterative matching* can be extended for the case that g is not a power of two. The general *iterative matching* algorithm maintains the group size for each group. The initial grouping is the 1-grouping of the input primitive rings. At each iteration, find a maximum-weighted matching in the intersection graph of the current grouping, merge the two groups incident to each edge with *nonzero* weight in the obtained matching, and update the group sizes accordingly. Such an iteration is repeated until the intersection graphs of the current grouping have no edges. At this moment, we merge the groups by applying any approximation algorithm [6] for the bin-packing problem to reduce the number of groups used. Note that even when g is a power of two the general *iterative matching* can perform potentially better than the restricted *iterative matching*. For example, if two groups have empty intersection in an iteration, they are left alone so that they can be potentially matched with some other groups in the future to save some ADMs.

We suggest another algorithm, running in time polynomial in $(m^g/g!)$. We can model the unrestricted ring grooming problem as a set cover problem as follows: the elements are $1, 2, \ldots, m$, the sets are the $\theta((mg)^g/g!)$ subsets of $\{1, 2, \ldots, m\}$ of size at most g, and each set B has cost $|\bigcup_{j\in B} A_j|$. A solution to the ring grooming corresponds to a solution to the set cover problem we defined, while from a set cover solution we can construct easily a ring grooming solution (which is a partition, not just a collection of sets) without increasing the cost. Thus we can use Chvatal's algorithm [5] to approximate within a factor of $O(\log g)$ this set cover problem. In conclusion, there is an algorithm for ring grooming with running time polynomial in $m^g/g!$ and performance ratio $O(\log g)$.

VI. Conclusion

This paper addresses the traffic grooming of arbitrary traffic in SONET/WDM rings. We first prove the NP-hardness of this problem. We then present two general lower bounds on the minimum ADM cost. After that we decompose the minimum ADM cost problem into two subproblems. Both subproblems remain NP-hard. Various approximation algorithms are proposed to each subproblem, and their performances are analyzed.

There are a number of open issues that are very challenging. First of all, the exact approximation ratios of most algorithms presented in this paper remain unknown despite that certain bound obtained in this paper. Second, the overall performance of the two-phased approach is still open although the performance of algorithms for each phase has been analyzed. Third, for all the problems studied in this paper, it is unknown whether they have polynomial-time approximation scheme. We would like to address these issues in the future.

References

[1] G. Călinescu, P. W. Liu, and P.-J. Wan, "Algorithms for ring grooming in SONET/WDM networks," paper, 2000, submitted for publication.

[2] ——, "Traffic partition in SONET/WDM rings to minimize SONET ADMs," paper, 2000, submitted for publication.

[3] ——, "Stability traffic partition in SONET/WDM rings to minimize SONET ADMs," paper, 2000, submitted for publication.

[4] A. Chiu and E. Modiano, "Traffic grooming algorithms for reducing electronic multiplexing costs in WDM ring networks," *J. Lightwave Technol.*, vol. 18, pp. 2–12, Jan. 2000.

[5] V. Chvátal, "A greedy heuristic for the set-covering problem," *Math. Operations Res.*, vol. 4, no. 3, pp. 233–235, 1979.

[6] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson, "Approximation algorithms for bin-packing—A survey," in *Approximation Aglorithms for NP-Hard Problems*. Boston: PWS Publishing, 1997, pp. 46–93.

[7] C. Colbourn and P.-J. Wan, "Minimizing drop cost for SONET/WDM networks with $\frac{1}{8}$ wavelength requirements," paper, 1999, submitted for publication.

[8] O. Gerstel, R. Ramaswami, and G. Sasaki, "Cost effective traffic grooming in WDM rings," in *Proc. IEEE INFOCOM'98*, vol. 1, San Francisco, CA, pp. 69–77.

[9] O. Gerstel, P. Lin, and G. Sasaki, "Wavelength assignment in a WDM ring to minimize cost of embedded SONET rings," in *Proc. IEEE INFOCOM'98*, vol. 1, San Francisco, CA, pp. 94–101.

[10] ——, "Combined WDM and SONET network design," in *Proc. IEEE INFOCOM'99*, vol. 2, New York, pp. 734–743.

[11] I. Haque, W. Kremer, and K. Raychauduri, "Self-healing rings in a synchronous environment," in *SONET/SDH: A Sourcebook of Synchronous Networking*, C. A. Siller and M. Shafi, Eds. New York: IEEE Press, 1996, pp. 131–139.

[12] X.-Y. Li and P.-J. Wan, "Select line speeds for single-hub SONET/WDM ring networks," accepted by ICC, 2000.

[13] L. W. Liu, X.-Y. Li, P.-J. Wan, and O. Frieder, "Wavelength assignment in WDM rings to minimize SONET ADMs," in *Proc. INFOCOM 2000*, vol. 2, Tel-Aviv, Israel, pp. 1020–1025.

[14] J. Simmons, E. Goldstein, and A. Saleh, "Quantifying the benefit of wavelength add–drop in WDM rings with distance-independent and dependent traffic," *J. Lightwave Techol.*, vol. 17, pp. 48–57, Jan. 1999.

[15] P.-J. Wan, "Combinatorial design of bidirectional SONET/WDM rings with $\frac{1}{2}$ and $\frac{1}{4}$ wavelength uniform traffic," paper, 2000, submitted for publication.

[16] P.-J. Wan, L.-W. Liu, and O. Frieder, "Grooming of arbitrary trafffic in SONET/WDM rings," in *Proc. IEEE GLOBECOM'99*, vol. 1B, pp. 1012–1016.

[17] X. Zhang and C. Qiao, "Effective and comprehensive solution to traffic grooming and wavelength assignment in SONET/WDM rings," in *SPIE Proc. All-Optical Networking: Architecture, Control, Management Issues*, vol. 3531, Boston, MA, Nov. 1998, pp. 221–232.

Peng-Jun Wan photograph and biography not available at the time of publication.

Gruia Călinescu photograph and biography not available at the time of publication.

Liwu Liu photograph and biography not available at the time of publication.

Ophir Frieder photograph and biography not available at the time of publication.