Efficient Algorithms for Traffic Grooming in SONET/WDM Networks

Yong Wang and Qian-Ping Gu
School of Computing Science, Simon Fraser University
Burnaby B.C. Canada V5A 1S6
{ywangb,qgu}@cs.sfu.ca

Abstract

In SONET/WDM optical networks, a wavelength channel is shared by multiple low-rate traffic demands. The multiplexing is known as traffic grooming *and carried out by SONET add-drop multiplexers (SADM). A key optimization problem in traffic grooming is to minimize the number of SADMs. This optimization problem is challenging and NP-hard even for unidirectional SONET/WDM rings (UPSR) with symmetric unitary traffic demands. In this paper, we give a linear time heuristic algorithm for this NP-hard problem. Empirical results show that the algorithm outperforms previous algorithms. The algorithm uses the minimum number of wavelengths, which are also precious resources in optical networks. An important subclass of the symmetric unitary traffic pattern is the regular traffic pattern, where each network node appears in exactly r symmetric demands. The regular traffic pattern is a generalization of the well known all-to-all traffic pattern, in which $r = n-1$ for a network of n nodes. We prove that the optimization problem remains NP-hard for the regular traffic pattern on the UPSR. We also propose an algorithm for this problem with a better upper bound on the number of used SADMs than previous algorithms. This algorithm always uses the minimum number of wavelengths as well.*

Keywords — *Traffic grooming, SONET/WDM networks, unidirectional rings, regular graph, graph decomposition, NP-complete.*

1 Introduction

In SONET/WDM networks, an optical fiber is shared by multiple wavelength channels, each of which has a bandwidth up to a few gigabit per second. Since the bandwidth required by a traffic demand in practice is usually much smaller than that of a wavelength channel, low-rate traffic demands are multiplexed to share a wavelength channel. The multiplexing is known as *traffic grooming*. The maximum number of low-rate traffic demands that can be multiplexed onto a wavelength channel is called *grooming factor*. For example, sixteen OC-3 traffic demands multiplexed onto one OC-48 wavelength channel gives a grooming factor of 16. Traffic grooming is realized by SONET add-drop multiplexers (SADM). When a wavelength channel carries low-rate traffic demands from/to a network node, one SADM is used to multiplex/demultiplex low-rate traffic demands onto/from the wavelength channel at the node. When a wavelength channel does not carry any traffic demand from/to a network node, the channel can optically bypass the node and no SADM is needed for the wavelength channel at the node. Since SADMs dominate the cost of SONET/WDM networks, it is critical to minimize the number of SADMs in the SONET/WDM network design. To reduce the number of SADMs, low-rate traffic demands should be groomed in a way to yield as many optical bypasses as possible. A major goal of traffic grooming in SONET/WDM networks is to find a grooming scheme for a given set of traffic demands such that the total number of used SADMs is minimized. This problem is surveyed in [5, 14, 15, 22]. Another important optimization goal in traffic grooming is to minimize the number of used wavelengths for a given set of traffic demands. It has been shown that the two optimization goals can not be achieved simultaneously [1, 7, 13] for many cases. Much work has been done on finding grooming schemes for one optimization goal subject to a specific condition of the other goal. Especially, grooming schemes for SADM optimization subject to using the minimum number of wavelengths have been explored in [1, 11, 13, 19, 21].

A main architecture for SONET/WDM networks is the unidirectional path-switched ring (UPSR) [8]. The optical fibers in a UPSR constitute two unidirectional rings with one in the clockwise direction and the other in the counter-clockwise direction. One ring (e.g., the clockwise ring) is used as a working ring and the other as a protecting ring. A low-rate traffic demand from node x to node y, denoted as (x, y), is carried by a wavelength channel on the unique path from x to y in the working ring. The wavelength of each channel is the same for all optical links used by the channel. A set R of traffic demands is *symmetric* if $(x, y) \in R$ implies $(y, x) \in R$. Symmetric traffic demands are common in many applications, for example, a TCP connection. A set of traffic demands are *unitary* if each demand requires one unit of bandwidth. We use $\{x, y\}$ to denote the pair of unitary traffic demands (x, y) and (y, x). In the UPSR, a pair $\{x, y\}$ of demands is carried by two wavelength channels, one is from x to y and the other is from y to x on the working ring. It is easy to prove that assigning a same wavelength to the two channels for $\{x, y\}$ uses no more SADMs than assigning two distinct wavelengths [18]. By this, for a set R of symmetric demand pairs, one can concentrate

on partitioning R into subsets such that each subset of pairs can be carried by one wavelength to solve the traffic grooming problem. The number of SADMs used for the wavelength assigned to each subset is the number of distinct nodes involved in the symmetric pairs in the subset. Minimizing the total number of used SADMs is equivalent to minimizing the sum of the number of distinct nodes appearing in each subset. In this paper, we consider the traffic grooming problem in the UPSR network for symmetric unitary traffic demands. Other variants of the traffic grooming problem in the UPSR include those for non-unitary and/or non-uniform traffic demands [4, 8, 17, 21].

For the problem we study in this paper, a graph partition approach has been used to develop various algorithms [1, 3, 9, 19]. In this approach, a set R of symmetric unitary traffic demand pairs is viewed as an undirected simple graph $G(V, E)$ called *traffic graph*, where $V(G)$ is the set of nodes in the UPSR and there is an edge in $E(G)$ between nodes x and y if and only if $\{x, y\} \in R$ The traffic grooming problem is formulated as the following k-edge-partitioning problem on the traffic graph: for a positive integer $k \leq |E(G)|$, partition the edge set $E(G)$ into a collection of subsets $\mathcal{E} = \{E_1, E_2, \ldots, E_W\}$ (where $\bigcup_{i=1}^{W} E_i = E(G)$ and $E_p \cap E_q = \emptyset$ for $p \neq q$), such that $|E_i| \leq k$ for each $E_i \in \mathcal{E}$ and $\sum_{E_i \in \mathcal{E}} |V_i|$ is minimized, where V_i is the set of nodes in the sub-graph induced by edge set E_i. It is obvious that integer k corresponds to the grooming factor, W corresponds to the number of used wavelengths, and $\sum_{E_i \in \mathcal{E}} |V_i|$ corresponds to the number of used SADMs in the traffic grooming problem.

The k-edge-partitioning problem on arbitrary traffic graphs has been proved NP-hard by Goldschmidt *et al.* [9]. Several algorithms [3, 9, 19] have been proposed, and those algorithms fall into two categories: spanning-tree-partition based algorithms [9, 19] and Euler-path-partition based algorithms [3]. In this paper, we propose a linear time algorithm combining the spanning-tree-partition based skeleton cover approach in [19] and the Euler path approach in [3]. Our algorithm uses the minimum number $\lceil \frac{|E(G)|}{k} \rceil$ of wavelengths. Experimental results show that our algorithm has better performance on the number of used SADMs than algorithms of [3, 9, 19] in most cases.

An important subclass of the symmetric traffic pattern is the regular traffic pattern, in which each network node appears in r symmetric traffic demand pairs, where $1 \leq r \leq n-1$ for a network of n nodes. The regular traffic pattern is a generalization of the all-to-all traffic pattern, in which $r = n - 1$. The all-to-all traffic pattern has been well studied for the traffic grooming problem [1, 11, 13, 21]. Little work has been known for the general regular traffic pattern, although it models the communications on the networks in which each node has limited capacity on sourcing/ending traffic demands simultaneously due to the hardware constraints. For the regular traffic pattern, the traffic graph is a r-regular graph. We prove that the traffic grooming problem remains NP-hard for the regular traffic pattern by showing the NP-hardness for the k-edge-partitioning problem on regular graphs. We also propose an algorithm for the regular traffic pattern. The algorithm uses at most $\lceil |E(G)|(1 + \frac{1}{k}) \rceil$ SADMs for even value r, and $\lceil |E(G)|(1 + \frac{1}{k}) \rceil + (\frac{3n}{2(r+1)} - 1)$ SADMs for odd value r, which are better than previous algorithms of [3, 9, 19] applying on regular graphs in most cases. The algorithm also uses the minimum number $\lceil \frac{|E(G)|}{k} \rceil$ of wavelengths.

The rest of the paper is organized as follows. Section 2 gives the preliminaries of the paper. The algorithm for the k-edge-partitioning problem on arbitrary traffic graphs is given in Section 3. In Section 4, we study the k-edge-partitioning problem on regular traffic graphs. Section 5 gives the experimental performances of our algorithms. The final section concludes the paper.

2 Preliminaries

Readers are referred to a textbook on graph theory (e.g., the one by West [20]) for basic definitions and terminology on graphs. Let $G(V, E)$ be a simple undirected graph with node set $V(G)$ and edge set $E(G)$. A *path* of G is a sequence of consecutive edges in G, where no repeated edge is allowed in the path. The two nodes at which the path starts and terminates are called *end-points* of the path, and all other nodes on the path are called *mid-points*. A *simple path* is a path with no repeated node. An *Euler path* is a path which uses each edge of G exactly once, and a connected graph has an Euler path if and only if it has at most two odd-degree nodes. A *matching* M is a set of edges of G such that no two of them share a node in common. We say a node is *saturated* by matching M if the node is an end-point of some edge in M. A *tree* T is a connected graph with $|V(T)| - 1$ edges, and a *spanning tree* of G is a tree that contains all nodes and $|V(G)| - 1$ edges of G. For a graph G and a spanning tree T of G, graph $G \backslash T$ is defined as a graph (which might be disconnected) with vertex set $V(G)$ and edge set $E(G) \backslash E(T)$. A *connected component* of G is a maximal connected sub-graph of G. The *edge connectivity* of G, denoted as $\lambda(G)$, is defined to be the minimum number of edges whose deletion from G disconnects graph G. An *edge coloring* of G is a coloring of the edges of G such that adjacent edges receive different colors. For each node v in graph G, we use $\delta(v)$ to denote the degree of v in G. If $\delta(v) = r$ for every $v \in V(G)$, we say G is a *r-regular graph*. We use $\Delta(G)$ to denote the maximum degree over all nodes in G, (i.e., $\Delta(G) = \max_{v \in V(G)}\{\delta(v)\}$), and we simply use Δ instead of $\Delta(G)$ when it is clear in the context.

We define a *skeleton* S of G to be a connected sub-graph of G that consists of a *backbone* and a set of *branches*, where the backbone is a path of G, and a branch is an edge of G such that at least one end-point of the edge is in the backbone. We say a branch $\{u, v\}$ is *attached* to a backbone if u or v is a node in the backbone. The size of a skeleton S, denoted as $s(S)$, is the number of edges in the skeleton. A *skeleton cover* $\mathcal{S}$ of graph G is a collection of skeletons $\{S_1, S_2, \ldots, S_j\}$ which form an edge partition of G (i.e., $\bigcup_{i=1}^{j} E(S_i) = E(G)$ and $E(S_p) \cap$

$E(S_q) = \emptyset$ for $p \neq q$), where j is the size of the skeleton cover. It was proved in [19] that the following proposition holds for any skeleton.

Proposition 1 *[19] For any skeleton S and integer t with $0 \leq t \leq s(S)$, S can be partitioned into two skeletons S_1 and S_2, such that $s(S_1) = t$ and $s(S_2) = s(S) - t$.*

By Proposition 1, it is easy to transform a skeleton cover to a k-edge partition of G with exactly k edges in each sub-graph except the last one (note that some sub-graphs might be disconnected). Especially we have the following proposition for any skeleton cover.

Proposition 2 *Any skeleton cover $\mathcal{S} = \{S_1, S_2, \ldots, S_j\}$ of graph G can be transformed into a k-edge partition $\mathcal{E} = \{E_1, ..., E_W\}$ of G with $W = \lceil \frac{|E(G)|}{k} \rceil$, $|E_i| = k$ for $1 \leq i < W$, and $\sum_{E_i \in \mathcal{E}} |V_i| \leq \lceil |E(G)|(1 + \frac{1}{k}) \rceil + (j-1)$.*

Proof: Let s_i and t_i be the end-points of the backbone of each skeleton $S_i \in \mathcal{S}$. We can connect $S_1, ..., S_j$ into one skeleton S^* by adding $(j-1)$ virtual edges $\{t_i, s_{i+1}\}$ for $1 \leq i \leq j-1$. According to Proposition 1, S^* can be cut into W skeletons $\{S_1^*, S_2^*, \ldots, S_W^*\}$, where $W = \lceil \frac{|E(G)|}{k} \rceil$, each skeleton S_i^* $(1 \leq i \leq W-1)$ contains exactly k edges of G and j_i virtual edges, and S_W^* contains at most k edges of G and j_W virtual edges. We have $\sum_{i=1}^{W} j_i = j - 1$ since the total number of added virtual edges is $j-1$. For a skeleton of $k + j_i$ edges, the maximum number of nodes in the skeleton is $k + j_i + 1$. Therefore the set of skeletons $\{S_1^*, S_2^*, \ldots, S_W^*\}$ becomes a k-edge-partition $\mathcal{E} = \{E_1, ..., E_W\}$ of G after deleting virtual edges, and

$$\sum_{E_i \in \mathcal{E}} |V_i| \leq |E(G)| + \sum_{i=1}^{W} j_i + W = \lceil |E(G)|(1 + \frac{1}{k}) \rceil + (j-1).$$

□

3 Grooming with arbitrary traffic graph

Intuitively, to achieve good solutions for the k-edge-partitioning problem, we need partition the traffic graph G into sub-graphs with at most k edges such that each sub-graph contains as few nodes as possible. One key observation is that given a fixed number of edges, a sub-graph with fewer connected components more likely contains fewer nodes. This is the basic idea behind the algorithms in [3, 9, 19]. In this section, we propose Algorithm Span_Euler which tries to minimize the total number of connected components over all sub-graphs. The algorithm in [3] constructs an Euler path of traffic graph G by adding virtual edges between odd-degree nodes. Then the Euler path is cut into segments and virtual edges are deleted to obtain a k-edge partition of G. However in the case that G contains a large number of odd-degree nodes, there will be a large number of virtual edges, whose deletion further implies a large number of connected components over sub-graphs in the k-edge partition. Another algorithm is proposed in [19] to construct a skeleton cover of G, and then transform the skeleton cover into a k-edge partition. Each skeleton is constructed based on a spanning-tree-partition approach, which likely produces skeletons with small size, and thus the size of the skeleton cover is usually large. According to Proposition 2, a skeleton cover with large size yields a solution of the k-edge-partitioning problem with large value. Our algorithm SpanT_Euler combines the techniques of constructing Euler path and skeleton cover. We intend to generate a skeleton cover of small size with the help of Euler path construction.

On constructing a skeleton cover, the following lemma holds.

Lemma 3 *If there exist l edge-disjoint paths which span every node in graph G, then G has a skeleton cover with size l.*

Proof: Each of the l paths can be considered as the backbone of a skeleton. Since the backbones span every node in graph G, other edges which do not appear in the backbones can be attached as branches. Thus the size of the constructed skeleton cover is exactly l. □

It is proved by Jaeger [12] that if $\lambda(G) \geq 4$, then there exists a path that spans every node in G, which implies that there exists a skeleton cover with size one for $\lambda(G) \geq 4$ according to Lemma 3. Following a similar argument, we generalize the result of [12] as the next Lemma.

Lemma 4 *Let T be a spanning tree of G, and c be the the number of connected components in graph $G \backslash T$, then G has a skeleton cover with size at most c.*

Proof: Let $V_{odd} \subseteq V(G)$ be the set of nodes having odd degree in graph $G \backslash T$. $|V_{odd}|$ must be even since the number of odd-degree nodes in any graph is even. Pair the nodes of V_{odd} arbitrarily and let $\mathcal{P} = \{p_1, p_2, \ldots, p_{\frac{|V_{odd}|}{2}}\}$ be the set of simple paths in T between each pair of nodes. For every edge $e \in E(T)$, let $\alpha(e)$ denote the number of paths (in $\mathcal{P}$) that contain edge e. Define

$$E_{odd} = \{e | e \in E(T) \text{ and } \alpha(e) \text{ is odd}\}, \text{ and}$$

$$E_{even} = \{e | e \in E(T), \alpha(e) \text{ is even, and } \alpha(e) \neq 0\}.$$

Let G_1 denote the graph with vertex set $V(G)$ and edge set E_{odd}. We shall prove in the following that V_{odd} is also the set of nodes that have odd degree in graph G_1.

For every node $v \in V_{odd}$, let $E_v \subseteq E_{odd} \cup E_{even}$ be the set of edges adjacent to node v. We notice that v is an end-point for exactly one path in $\mathcal{P}$ (i.e., only one edge in the path is adjacent to v), and for any other path in $\mathcal{P}$, either v is a mid-point of the path (i.e., two edges in the path are adjacent to v) or v is not in the path (i.e., no edge in the path is adjacent to v). Therefore $\sum_{e \in E_v} \alpha(e)$ must be odd. We also know that

$$\sum_{e \in E_v} \alpha(e) = \sum_{e \in (E_v \cap E_{odd})} \alpha(e) + \sum_{e \in (E_v \cap E_{even})} \alpha(e),$$

Algorithm SpanT_Euler
Input: An undirected graph G and integer k.
Output: A k-edge partition of G.
begin
Compute a spanning tree T of G;
Let c be the number of connected components in $G\backslash T$;
Let V_{odd} be the set of odd degree nodes in $G\backslash T$;
Pair nodes of V_{odd}, and let $\mathcal{P} = \{p_1, p_2, \ldots, p_{|V_{odd}|/2}\}$ be simple paths in T between each pair of nodes;
Let $E_{odd} \subseteq E(T)$ be the set of edges appearing in odd number paths of $\mathcal{P}$;
Construct an Euler path for each of the c connected components of G_2;
Attach each edge in $E(T)\backslash E_{odd}$ to Euler paths to obtain a skeleton cover $\mathcal{S} = \{S_1, ..., S_c\}$;
Transform skeleton cover to a k-edge partition of G;
end.

Figure 1. Pseudo code of SpanT_Euler.

so $|E_v \cap E_{odd}|$ must be odd, that is, every node $v \in V_{odd}$ has odd degree in graph G_1. Similarly for every node $u \in V(G)\backslash V_{odd}$, since either u is a mid-point of some paths or u is not in any path, we have that $\sum_{e \in E_u} \alpha(e)$ must be even. That is,

$$\sum_{e \in E_u} \alpha(e) = \sum_{e \in (E_u \cap E_{odd})} \alpha(e) + \sum_{e \in (E_u \cap E_{even})} \alpha(e)$$

must be even, and thus $|E_u \cap E_{odd}|$ must be even. Therefore every node $u \in V(G)\backslash V_{odd}$ has even degree in graph G_1.

Let G_2 denote the graph with vertex set $V(G)$ and edge set $E_{odd} \cup (E(G)\backslash E(T))$. Since V_{odd} is defined to be the set of nodes having odd degree in graph $G\backslash T$, every node in graph G_2 has even degree. We know that $G\backslash T$ contains at most c connected components, so there are at most c connected components in graph G_2 as well. For each of the c connected component in G_2, we can construct an Euler path since all nodes have even degree (for the component consisting of a single node, the Euler path is a special one consisting of the single node). It is clear that the Euler paths span every node in G. Thus we can construct a skeleton cover of G with size at most c according to Lemma 3. □

We need point out that the above lemma generalizes the result by Jaeger [12], which showed that if $\lambda(G) \geq 4$ then G has two edge-disjoint spanning trees T_1, T_2. This implies $c = 1$ if we remove T_1 (or T_2) from G.

The pseudo code of Algorithm SpanT_Euler is given in Figure 1.

Theorem 5 *Algorithm SpanT_Euler finds a k-edge partition $\mathcal{E} = \{E_1, ..., E_W\}$ of G with $W = \lceil \frac{|E(G)|}{k} \rceil$, $|E_i| = k$ for $1 \leq i < W$, and $\sum_{E_i \in \mathcal{E}} |V_i| \leq \lceil |E(G)|(1 + \frac{1}{k}) \rceil + (c - 1)$.*

Proof: The theorem can be easily proved based on Proposition 2 and Lemma 4. □

We do not have a precise upper bound on the number c of connected components in graph $G\backslash T$, however some algorithms on constructing spanning tree T with specific properties might be used to achieve such bounds (e.g., the algorithm by F$\ddot{u}rer$ and Raghavachari [6] approximates the optimal solution on generating a spanning tree T with maximum degree $\Delta(T)$ minimized). We develop experimental simulation in Section 5, and Algorithm SpanT_Euler outperforms the algorithms in [3, 9, 19] according to our simulation results.

Since the spanning tree generation and Euler path construction can be done in $O(|E(G)|)$ time, it is clear that each step of Algorithm SpanT_Euler takes $O(|E(G)|)$ time. So Algorithm SpanT_Euler runs in $O(|E(G)|)$ time, which is linear in the size of the input graph.

4 Grooming with regular traffic graph

Due to the hardware constraints in some networks, each node can communicate with at most r other nodes simultaneously at any specific time. The communication on such networks can be abstracted as a r-regular traffic graph. We first prove that the k-edge-partitioning problem remains NP-hard on r-regular graph for arbitrary r, and then give an algorithm with guaranteed performance.

We consider the decision version of the k-Edge-Partitioning of Regular Graph (KEPRG) problem, and prove it is NP-complete. The KEPRG problem is stated as follows:

k-Edge-Partitioning of Regular Graph (KEPRG) Problem
Instance: An undirected regular graph $G(V, E)$, and integers k and L.
Question: Is there a partition of $E(G)$ into a collection of subsets $\mathcal{E} = \{E_1, E_2, \ldots, E_W\}$ (where $\bigcup_{i=1}^{W} E_i = E(G)$ and $E_p \cap E_q = \emptyset$ for $p \neq q$), such that $|E_i| \leq k$ for each $E_i \in \mathcal{E}$ and $\sum_{E_i \in \mathcal{E}} |V_i| \leq L$, where V_i is the set of nodes in the sub-graph induced by edge set E_i?

We shall give a two-step reduction from the Edge-Partition into Triangles (EPT) problem to the KEPRG problem. The EPT problem is known to be NP-complete [10] and stated as follows:

Edge-Partition into Triangles (EPT) Problem
Instance: An undirected graph $G(V, E)$ with $|E(G)| = m$.
Question: Is there a partition of $E(G)$ into sets $\{E_1, E_2, ..., E_{m/3}\}$ such that each E_i induces a triangle?

In the first step we show the EPT problem on regular graphs is NP-complete by the reduction from the EPT problem.

Lemma 6 *The EPT problem is NP-complete even if the input graph G is regular.*

Proof: It is easy to see that if an odd-degree node exists in a graph, the graph can not be partitioned into triangles. There-

fore, the EPT problem is NP-complete even if the input graph contains no odd-degree nodes. Otherwise, the EPT problem itself is not NP-complete. In what follows, we assume the input graph has no odd-degree nodes when we refer to an instance of the EPT problem.

The regular graph version of EPT is clearly in NP. Given an undirected graph G with maximum degree Δ (where Δ is even) , we construct a Δ-regular graph G^* as follows (for simplicity, we use (u, v, w) to denote a triangle with $\{u, v, w\}$ as node set):

1. For every node v in G with degree $\delta(v) < \Delta$, add triangles $\{(v, u_{v,i}, u_{v,i+1})\}$ for $i = 1, 3, 5, \ldots, \Delta - \delta(v) - 1$ to get graph G', where the $\Delta - \delta(v)$ nodes $u_{v,i}$'s and $u_{v,i+1}$'s are the new added nodes.

2. Make two extra copies of G', and assume the number of new added nodes in G' is q.

3. Re-label all the new added nodes as u_1, u_2, u_3, ..., u_{3q}.

4. If $3q < \frac{\Delta-2}{2}$

   (a) add $3p$ new nodes $u_{3q+1}, u_{3q+2}, u_{3q+3}, \ldots, u_{3q+3p}$, where p is the smallest integer satisfying $3q + 3p \geq \frac{\Delta-2}{2}$;

   (b) add p triangles $\{(u_{3q+i}, u_{3q+i+1}, u_{3q+i+2})\}$ for $i = 1, 4, 7, \ldots, 3p - 2$;

   (c) set $q = q + p$.

5. Add $3q$ new nodes $w_1, w_2, w_3, \ldots, w_{3q}$ and q triangles $\{(w_i, w_{i+1}, w_{i+2})\}$ for $i = 1, 4, 7, \ldots, 3q - 2$. Add another $3q$ new nodes $y_1, y_2, y_3, \ldots, y_{3q}$ and q triangles $\{(y_i, y_{i+1}, y_{i+2})\}$ for $i = 1, 4, 7, \ldots, 3q - 2$.

6. Repeat the following for $i = 1, 2, 3, \ldots, \frac{\Delta-2}{2}$: add triangles $\{(u_j, w_{j\oplus i}, y_{j\oplus 2i})\}$ for $j = 1, 2, 3, \ldots, 3q$, where $\oplus$ is defined as mod $3q$ sum.

Figure 2 shows how to construct the regular graph G^* from G by an example (We make two copies for nodes y_1, y_2 and w_1 to avoid messy drawing). Now we prove that G can be partitioned into triangles if and only if G^* can be partitioned into triangles. If G can be partitioned into triangles, according to the construction of G^*, G^* can be partitioned into triangles as well. If G^* can be partitioned into triangles, it is noticed that any triangle containing an edge from a copy of G can not contain any edge outside of that copy. So each copy of G by itself must be able to be partitioned into triangles. □

In the second step we show that the KEPRG problem is NP-complete by the reduction from the regular graph version of the EPT problem.

Theorem 7 *The KEPRG problem is NP-complete.*

Proof: The problem is clearly in NP. Given an instance of the regular graph version of EPT problem, where the input graph G is regular, we construct an instance of the KEPRG problem on the same graph with $L = m$ and $k = 3$, where $m = |E(G)|$.

If G can be partitioned into triangles, the triangle partition gives a solution for the KEPRG problem where $|E_i| = 3$ for each $E_i \in \mathcal{E}$ and $\sum_{E_i \in \mathcal{E}} |V_i| = m$.

We now prove that if G has a solution for the KEPRG problem with $k = 3$ and $L = m$, then it can be partitioned into triangles. It is noticed that $\sum_{E_i \in \mathcal{E}} |V_i| \geq m$ for $k = 3$, and the equation holds if and only if G can be partitioned into complete graphs with 3 edges (i.e., triangles). Since we have $\sum_{E_i \in \mathcal{E}} |V_i| \leq L = m$, it must be the case that $\sum_{E_i \in \mathcal{E}} |V_i| = m$, which implies that G can be partitioned into triangles. □

We shall give Algorithm Regular_Euler with guaranteed performance for the KEPRG problem. First, we prove the following lemma about the lower bound on the size of the maximum matching of regular graphs. This lemma also gives a solution for an open problem proposed by Biedl *et al.* [2].

Lemma 8 *A r-regular graph G with $|V(G)| = n$ has a maximum matching containing at least $\frac{n}{2} \cdot \frac{r}{r+1}$ edges.*

Proof: It is proved by Vizing [16] that any simple graph G has an edge coloring with Δ or $\Delta + 1$ colors, where Δ is the maximum degree of G. The proof immediately yields an approximation algorithm to color any simple graph G with at most $\Delta + 1$ colors. Therefore, a r-regular graph G can be colored by $r + 1$ colors. Considering such an edge coloring, we notice that each set of edges with the same color is a matching of G. Since the total number of edges in G is $\frac{nr}{2}$, there must exist at least $\frac{nr/2}{r+1} = \frac{n}{2} \cdot \frac{r}{r+1}$ edges having the same color in the edge coloring. Therefore, $\frac{n}{2} \cdot \frac{r}{r+1}$ is a lower bound on the size of the maximum matching of r-regular graph. □

We prove the following lemma for any r-regular graph, where r is nontrivial (i.e., $r \neq 0$ or 1).

Lemma 9 *For r-regular graph G with $|V(G)| = n$, there exists a skeleton cover of size 1 if r is even, and there exists a skeleton cover of size at most $\frac{3n}{2(r+1)}$ if r is odd.*

Proof: For even value r, we can construct an Euler path of G. Such an Euler path is a skeleton of G without branches, therefore we obtain a skeleton cover of size 1.

For odd value r, we first compute a maximum matching M of G. Then after deleting the edges in M from graph G, all the nodes saturated by M have degree $r - 1$, all the unsaturated nodes have degree r, and there might be more than one connected components in graph $G(V(G), E(G) \backslash M)$. We call a connected component in $G(V(G), E(G) \backslash M)$ an *even component* if every node in the component has degree $r - 1$, and an *odd component* if there exists a node with degree r. Assume there are s even components and t odd components. Let

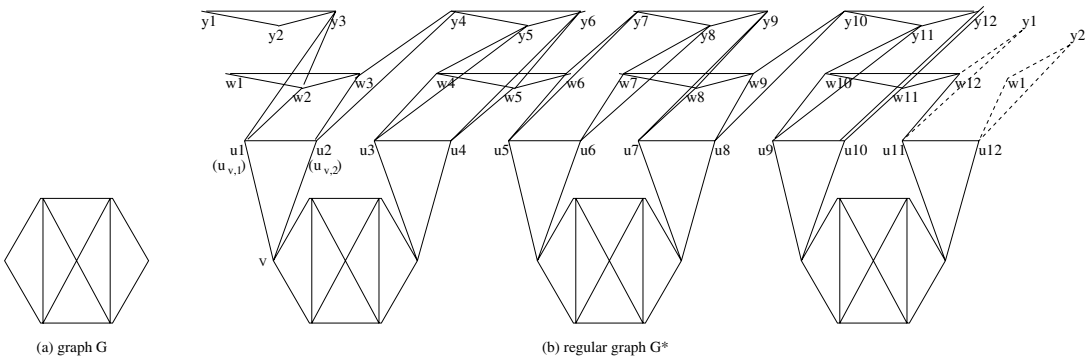

Figure 2. An example: graph G with $\Delta = 4$ and the corresponding 4-regular graph G^*.

$\mathcal{C}_{even} = \{C_1', C_2', \ldots, C_s'\}$ be the set of even components, and $\mathcal{C}_{odd} = \{C_1, C_2, \ldots, C_t\}$ be the set of odd components. It is noticed that any odd component C_i contains at least two nodes $v_{i,1}, v_{i,2}$ with degree r, since the number of odd-degree nodes in any graph must be even. Thus all the odd components can form one connected graph G_{odd} by adding $t-1$ virtual edges $(v_{1,1}, v_{2,1}), (v_{2,2}, v_{3,1}), \ldots, (v_{t-2,2}, v_{t-1,1}), (v_{t-1,2}, v_{t,1})$. Then for other nodes with degree r in graph G_{odd}, we arbitrarily pair them and add one virtual edge between each pair. So the degree of each node in G_{odd} is either $r+1$ or $r-1$, except that nodes $v_{t,2}$ and $v_{1,2}$ have degree r. Therefore we can construct an Euler path of G_{odd}. Since each node has degree $r-1$ in an even component, we can construct an Euler path for each even component as well. Thus we obtain $s+1$ edge-disjoint Euler paths in total, and those Euler paths span every node of G. According to Lemma 3, we can construct a skeleton cover with size $s+1$. After deleting the virtual edges, we obtain a skeleton cover of G with size $(s+1) + (\frac{n-2|M|}{2} - 1) = s + \frac{n-2|M|}{2}$ since there are $\frac{n-2|M|}{2} - 1$ virtual edges in total.

Since each node in an even component has degree $r-1$, each even component must contain at least r nodes. In graph $G(V(G), E(G) \backslash M)$, the total number of nodes with degree $r-1$ is $2|M|$, so the number of even components in graph $G(V(G), E(G) \backslash M)$ is at most $\frac{2|M|}{r}$, that is, $s \leq \frac{2|M|}{r}$.

We have $|M| \geq \frac{n}{2} \cdot \frac{r}{r+1}$ according to Lemma 8, so the size of the skeleton cover for odd value r is

$$s + \frac{n-2|M|}{2} \leq \frac{2|M|}{r} + \frac{n-2|M|}{2} \leq \frac{3n}{2(r+1)}.$$

□

The pseudo code of Algorithm Regular_Euler is given in Figure 3.

Theorem 10 *Algorithm Regular_Euler finds a k-edge partition $\mathcal{E} = \{E_1, ..., E_W\}$ of r-regular graph G with $W = \lceil \frac{|E(G)|}{k} \rceil$, $|E_i| = k$ for $1 \leq i < W$, $\sum_{E_i \in \mathcal{E}} |V_i| \leq \lceil |E(G)|(1 + \frac{1}{k}) \rceil$ for even value r, and $\sum_{E_i \in \mathcal{E}} |V_i| \leq \lceil |E(G)|(1 + \frac{1}{k}) \rceil + (\frac{3n}{2(r+1)} - 1)$ for odd value r.*

Algorithm Regular_Euler
Input: An undirected r-regular graph G and integer k.
Output: A k-edge partition of G.
begin
  If r is even **then begin**
    Construct an Euler path of G;
    Cut Euler path into $W = \lceil |E(G)|/k \rceil$ segments s.t.
      each of first $W-1$ segments has k edges of G;
  end
  else begin
    Compute a maximum matching M of G;
    Connect odd components in $G(V(G), E(G) \backslash M)$
      by virtual edges to form connected graph G_{odd};
    Construct an Euler path for G_{odd};
    Construct an Euler path for each even component
      in graph $G(V(G), E(G) \backslash M)$;
    Attach edges in M to Euler paths and delete virtual
      edges to obtain skeleton cover $\mathcal{S} = \{S_1, ..., S_{\frac{3n}{2(r+1)}}\}$;
    Transform skeleton cover to a k-edge partition of G;
  end;
end.

Figure 3. Pseudo code of Regular_Euler.

Proof: The theorem can be easily proved based on Proposition 2 and Lemma 9. □

If we apply the algorithms in [3] on r-regular graph, the upper bound on the number of SADMs is $\lceil |E(G)|(1 + \frac{1}{k}) \rceil$ for even value r, and $\lceil |E(G)|(1 + \frac{1}{k}) \rceil + \frac{n}{2}$ for odd value r. Therefore our algorithm Regular_Euler always gives a better upper bound. The algorithm in [9] applying on r-regular graph gives an upper bound of $\lceil |E(G)|(1 + \frac{2}{k}) \rceil$. So for even value r, Algorithm Regular_Euler is always better. And for odd value r, Algorithm Regular_Euler is better as long as the graph is relatively dense. The algorithm in [19] applying on r-regular graph gives an upper bound of $\lceil |E(G)|(1 + \frac{1}{k}) \rceil + \lfloor \frac{n}{4} \rfloor$, therefore Algorithm Regular_Euler is always better except the only case of $r = 3$. In summary, Algorithm Regular_Euler almost always achieves a better upper bound than previous al-

gorithms.

A maximum matching can be computed in $O(|V|^{\frac{1}{2}}|E(G)|)$ time, the Euler path construction can be done in $O(|E(G)|)$, and each of other steps in Algorithm Regular_Euler can be done in $O(|E(G)|)$ as well, therefore Algorithm Regular_Euler runs in $O(|V|^{\frac{1}{2}}|E(G)|)$ time.

5 Empirical results

We implemented the algorithms in [3, 9, 19], and Algorithm SpanT_Euler to compare their performances on randomly generated traffic graphs. In generating graphs, we specify the number n of nodes and dense ratio d that is used to calculate the number $m = n^{1+d}$ of edges. A graph of m edges and n nodes is generated by randomly connecting m pairs of nodes among all $n(n-1)/2$ possible pairs. Figure 4 shows the empirical results for graphs of 36 nodes with different settings of dense ratio d and grooming factor k. We refer to the algorithm in [9] [3] and [19] as Algo. 1, Algo. 2 and Algo. 3 respectively. For these previous algorithms, we observe that the spanning-tree-partition based algorithms Algo. 1 and Algo. 3 have better performances if dense ratio d is smaller, and the Euler-path-partition based algorithm Algo. 2 has better performance if dense ratio d is larger. As we mentioned in Section 3, Algorithm SpanT_Euler combines the techniques of constructing Euler path and skeleton cover. Therefore, Algorithm SpanT_Euler can be considered as a hybrid of the spanning-tree-partition based algorithm in [19] and the Euler-path-partition based algorithm in [3]. The empirical results in Figure 4 verify that Algorithm SpanT_Euler does take advantages of both approaches, and has better performance than all of the previous algorithms. The performance is especially good for grooming factor k being relatively small values (e.g., $k \leq 16$), which include some important values of practical interest. We ran simulations on graphs with different parameter settings as well, and the results present similar characteristics. To save space, we report some representative results in Figure 4.

We have shown in Section 4 that Algorithm Regular_Euler achieves better worst case performance guarantee than previous algorithms. We implemented Algorithm Regular_Euler as well to compare its performance to previous algorithms. We use the regular graph generator from [23] to randomly generate traffic graphs. The results show that Algorithm Regular_Euler outperforms previous algorithms in most cases. We ran the simulation on graphs with different parameter settings as well, and the results present similar characteristics. To save space, we report some representative results in Figure 5.

6 Concluding remarks

In this paper, we studied the traffic grooming problem on the SONET/WDM UPSR network with symmetric unitary traffic demands. For arbitrary traffic graphs, we gave an algorithm which uses the minimum number of wavelengths and achieves better practical performance than previous algorithms according to our simulation results. For the case that the traffic graph is regular, we proved the traffic grooming problem remains NP-hard. We also gave an algorithm with guaranteed performance for this subclass of symmetric traffic pattern, and the algorithm achieves a better upper bound in most cases than previous algorithms and uses the minimum number of wavelengths as well. Further improvements for both algorithms include heuristics on constructing denser sub-graphs in the k-edge partition, for example, partitioning the traffic graph into sub-graphs which are cliques or close to cliques. In addition, for Algorithm SpanT_Euler, it is worth developing techniques to bound the number of connected components after deleting spanning tree T from graph G, such that the guaranteed performance can be achieved.

Acknowledgment

This work was partially supported by the NSERC research grant (611352).

References

[1] J.-C. Bermond and D. Coudert. Traffic grooming in unidirectional WDM ring networks using design theory. In *Proceedings of IEEE ICC'03*, pages 1995–2003, 2003.

[2] T. Biedl, E.D. Demaine, C.A. Duncan, R. Fleischer, and S.G. Kobourov. Tight bounds on maximal and maximum matchings. *Discrete Mathematics*, 285:7–15, 2004.

[3] N. Brauner, Y. Crama, G. Finke, P. Lemaire, and C. Wynants. Approximation algorithms for the design of SDH/SONET networks. *RAIRO Operations Research*, 37:235–247, 2003.

[4] W. Cho, J. Wang, and B. Mukherjee. Improved approaches for cost-effective traffic grooming in WDM ring networks: uniform-traffic case. *Photonic Network Communications*, 3(3):245–254, 2001.

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

[6] M. Fürer and B. Raghavachari. Approximating the minimum degree spanning tree to within one from the optimal degree. In *Proceedings of ACM-SIAM SODA'92*, pages 317–324, 1992.

[7] O. Gerstel, P. Lin, and G. Sasaki. Wavelength assignment in a WDM ring to minimize cost of embedded SONET rings. In *Proceedings of IEEE INFOCOM'98*, pages 94–101, 1998.

[8] O. Gerstel, P. Lin, and G. Sasaki. Combined WDM and SONET network design. In *Proceedings of IEEE INFOCOM'99*, pages 734–743, 1999.

[9] O. Goldschmidt, D.S. Hochbaum, A. Levin, and E.V. Olinick. The SONET edge-partition problem. *Network*, 41(1):13–23, 2003.

[10] I. Holyer. The NP-completeness of some edge-partition problems. *SIAM Journal on Computing*, 10:713–717, 1981.

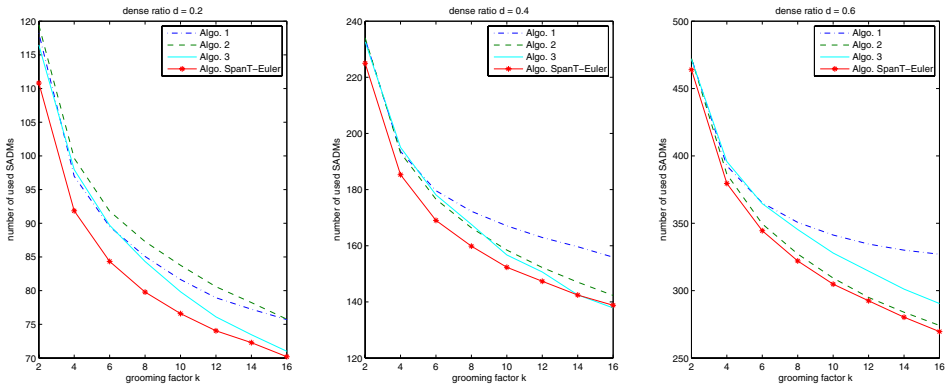

Figure 4. Algorithm SpanT_Euler: empirical results on graphs with $n = 36$ nodes.

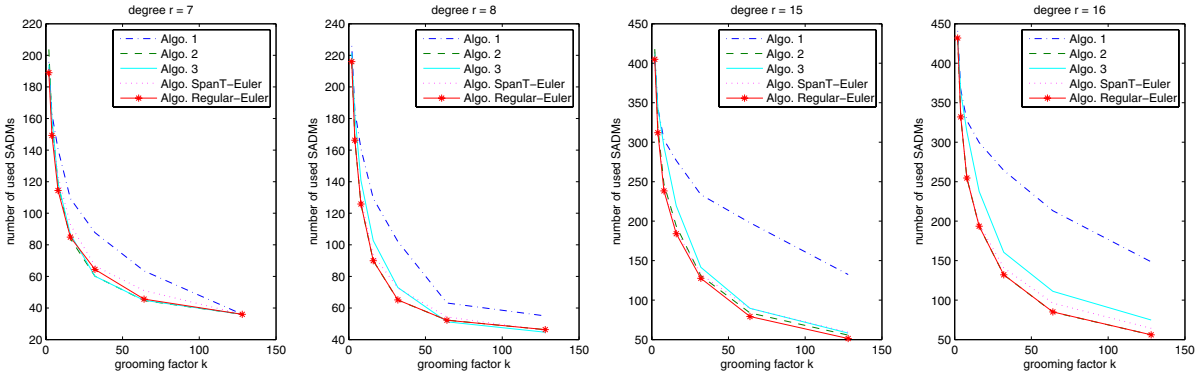

Figure 5. Algorithm Regular_Euler: empirical results on graphs with $n = 36$ nodes.

[11] J.Q. Hu. Optimal traffic grooming for wavelength-division-multiplexing rings with all-to-all uniform traffic. *Journal of Optical Networking*, 1(1):32–42, 2002.

[12] F. Jaeger. A note on sub-Eulerian graphs. *Journal of Graph Theory*, 3:91–93, 1979.

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

[14] E. Modiano and P. Lin. Traffic grooming in WDM networks. *IEEE communications Magazine*, 39(7):124–129, 2001.

[15] A.K. Somani. Surviable traffic grooming in WDM networks. In *Proceedings of the International Conference on Broad-Band Optical Fibre Communication Technology (BBOFCT'01)*, pages 17–45, 2001.

[16] V.G. Vizing. On an estimate of the chromatic class of a p-graph (in Russian). *Metody Discret Analiz.*, 3:25–30, 1964.

[17] J. Wang, W. Cho, V. Vemuri, and B. Mukherjee. Improved approaches for cost-effective traffic grooming in WDM ring networks: ILP formulations and single-hop and multihop connections. *Journal of Lightwave Technology*, 19(11):1645–1653, 2001.

[18] Y. Wang and Q.-P. Gu. Grooming of symmetric traffic in unidirectional SONET/WDM rings. *Technical Report 2005-15, School of Computing Science, Simon Fraser University*, 2005.

[19] Y. Wang and Q.-P. Gu. Grooming of symmetric traffic in unidirectional SONET/WDM rings. To appear in *Proceedings of IEEE ICC'06*.

[20] D.B. West. *Introduction to graph theory*. Prentice Hall Inc., Upper Saddle River, NJ, 1996.

[21] X. Zhang and C. Qiao. An effective and comprehensive approach for traffic grooming and wavelength assignment in SONET/WDM rings. *IEEE/ACM Transactions on Networking*, 8(5):608–617, 2000.

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

[23] M. Meringer. Genreg: A very fast structure generator for regular graphs. *http://www.mathe2.uni-bayreuth.de/markus/reggraphs.html*.