Approximation Algorithms for Survivable Optical Networks (Extended Abstract)

T. Eilam S. Moran S. Zaks
Department of Computer Science
The Technion,
Haifa 32000, Israel
email: {eilam,moran,zaks}@cs.technion.ac.il

We are motivated by the developments in all-optical networks – a new technology that supports high bandwidth demands. These networks provide a set of lightpaths which can be seen as high-bandwidth pipes on which communication is performed. Since the capacity enabled by this technology substantially exceeds the one provided by conventional networks, its ability to recover from failures within the optical layer is important. In this paper we study the design of a survivable optical layer. We assume that an initial set of lightpaths (designed according to the expected communication pattern) is given, and we are targeted at augmenting this initial set with additional lightpaths such that the result will guarantee survivability. For this purpose, we define and motivate a *ring partition survivability condition* that the solution must satisfy. Generally speaking, this condition states that lightpaths must be arranged in rings. The cost of the solution found is the number of lightpaths in it. This cost function reflects the *switching cost* of the entire network. We present some negative results regarding the tractability and approximability of this problem, and an approximation algorithm for it. We analyze the performance of the algorithm for the general case (arbitrary topology) as well as for some special cases.

1 Introduction

1.1 Background

Optical networks play a key role in providing high bandwidth and connectivity in today's communication world, and are currently the prefered medium for the transmission of data. While first generation optical networks simply served as a transmission medium, second generation optical networks perform some switching and routing functions in the optical domain. In these networks (also termed, *all-optical*) routing is performed by using *lightpaths*. A lightpath is an end-to-end connection established across the optical network. Every lightpath corresponds to a certain route in the network, and it uses a wavelength in each link in its route. (Two lightpaths which use a same link are assigned different wavelengths.) Routing of messages is performed on top of the set of lightpaths where the route of every message is a sequence of complete lightpaths. At least in the near term the optical layer provides a static (fixed) set of lightpaths which is set up at the time the network is deployed.

Since the capacity enabled by this technology substantially exceeds the one provided by conventional networks, it is important to incorporate the ability to recover from failures into the optical layer. *Survivability* is the ability of the network to recover from failures of hardware components. In this paper we study the design of a survivable optical layer. Our goal is the construction of a low-cost survivable set of lightpaths in a given topology. We assume that an initial set of lightpaths (designed according to the expected communication pattern) is given, and we are targeted at augmenting this initial set with additional lightpaths such that the resulting set will guarantee survivability. For this purpose, we define a *survivability condition* that the solution must satisfy and a *cost function* according to which we evaluate the cost of the solution found.

We focus on the *ring partition survivability condition*. Informally, this condition states that lightpaths are partitioned to rings, and that all lightpaths in a ring traverse disjoint routes in the underlying topology. The motivation for the ring partition survivability condition is two folded. First, it supports a simple and fast protection mechanism. In the case of a failure, the data is simply re-routed around the impaired lightpath, on the alternate path of lightpaths in its ring. The demand that all lightpaths in one ring traverse disjoint routes guarantees that this protection mechanism is always applicable in the case of one failure. Second, a partition of the lightpaths to rings is necessary in order to support a higher layer in the form of SONET/SDH self healing rings which is anticipated to be the most common architecture at least in the near term future ([GLS98]).

Another issue is determining the cost of the design. We assume that a *uniform cost* is charged for every lightpath, namely, the cost of the design is the number of lightpaths in it. This cost measure is justified for two reasons. First, in regional area networks it is reasonable to assume that the same cost will be charged for all the lightpaths ([RS98]). Second, every lightpath is terminated by a pair of line terminals (LTs, in short). The *switching cost* of the entire network is dominated by the number of LTs which is proportional to the number of lightpaths ([GLS98]).

We assume that the network topology is given in the form of a simple graph. A lightpath is modeled as a pair (ID, P) where ID is a unique identifier and P is a simple path in the graph. A *design* D for a set of lightpaths C is a set of lightpaths which subsumes C (i.e., $C \subseteq D$). A design is termed *ring partition* if it satisfies the ring partition condition. The *cost* of a design is the number of lightpath in it (namely, $cost(D) = |D|$). We end up with the following optimization problem which we term the *minimum cost ring partition design* (MCRPD in short) problem. The input is a graph G and an initial set C of lightpaths in G. The goal is to find a ring partition design D for C with minimum cost.

1.2 Results

We prove that the MCRPD problem is NP-hard for every family of topologies that contains cycles with unbounded length, e.g., rings (see formal definition in the sequel). Moreover, we prove that there is no polynomial time approximation algorithm A that constructs a design D which satisfies $Cost(D) \leq OPT + n^{\alpha}$, for any constant $\alpha < 1$, where n is the number of lightpaths in the initial set, and OPT is the cost of an optimal solution for this instance (unless $P = NP$). For $\alpha = 1$, a trivial approximation algorithm constructs a solution within this bound.

We present a *ring partition algorithm* (RPA, in short) which finds in polynomial time a ring partition design for every given instance of MCRPD (if it exists). We analyze the performance of RPA and show that for the general case (arbitrary topology) RPA guarantees
$Cost(D) \leq OPT + \frac{3}{5} \cdot n$, where n and OPT are as defined above. We analyze the performance of RPA also for some interesting special cases in which better results are achieved.

We first present the model (Section 2), followed by a description of the MCRPD problem (Section 3). We then discuss the results (Section 4), followed by a summary and future research directions (Section 5). Some of the proofs in this extended abstract are only briefly sketched or omitted.

1.3 Related Works

The paper [GLS98] studies ring partition designs for the special case where the physical topology is a ring. In fact, the MCRPD problem is a generalization of this problem for arbitrary topologies. This paper also motivates the focus on the number of lightpaths rather than the total number of wavelengths in the design. Some heuristics to construct ring partition designs in rings are given and some lower and upper bounds on the cost (as a function of the load) are proved. The paper also considers *lightpath splitting* – a lightpath might be partitioned to two or more lightpath. It is shown that better results can be achieved by splitting lightpaths.

Other works in this field refer to different models than what we considered. [GRS97] presents methods for recovering from channel, link and node failures in first generation WDM ring networks with limited wavelength conversion.

Other works refer to second generation optical networks, where traffic is carried on a set of lightpaths. The paper [RS97] assumes that lightpaths are dynamic and focuses on management protocols for setting them up and taking them down.

When the set of lightpaths is static, the survivability is achieved by providing disjoint routes to be used in the case of a failure. [HNS94] and [AA98] studies this problem but the objective is the minimization of the total number of wavelengths and not the number of lightpaths.

The paper [ACB97] offers some heuristics and empirical results for the following problem. Given the physical topology and a set of connections requests (i.e., requests for lightpaths in the form of pairs of nodes), find routes for the requests so as to minimize the number of pairs (l, e) consisting of a routed request (i.e., a lightpath) l and a physical link e, for which there is no alternative path of lightpaths between the endpoints of l in the case that e fails. Note that this survivability condition is less restrictive than the ring partition condition that we consider in this paper.

2 Model and Definitions

For our purposes, lightpaths are modeled as *connections*, where every connection c has a unique identifier $ID(c)$ and is associated with a simple path $\mathcal{R}(c)$ in the network. $\mathcal{R}$ is termed the *routing function.* Note that two different connections might have the same route. We assume that routes of connections are always simple (i.e., they do not contain loops). We say that two connections are *disjoint* if their routes are disjoint, namely, they do not share any edge and any node which is not an end node of both connections. We use the terms connections and lightpaths interchangeably.

A *virtual path* P is a sequence $\langle v_1, c_1, v_2, c_2, \cdots, c_k, v_{k+1}\rangle$, where c_i is a connection with endpoints v_i and v_{i+1} (for $i = 1, \cdots, k$). P is termed a *virtual cycle* if $v_1 = v_{k+1}$. We denote by $S(P)$ the set $\{c_1, c_2, \cdots, c_k\}$ of connections in P. The routing function $\mathcal{R}$ is naturally generalized to apply to virtual paths (and cycles) by concatenating the corresponding paths of connections. A virtual path (or cycle) P is termed *plain* if $\mathcal{R}(P)$ is a simple path (or cycle) in the network.

A design D for a set of connections C in a network G is a set of connections which subsumes C (i.e., $C \subseteq D$). A *ring partition design* D for a set of connections C satisfies $D = \cup_{t\in T} S(P_t)$, where every P_t, $t \in T$, is a plain virtual cycle, and $S(P_{t_1}) \cap S(P_{t_2}) = \emptyset$, for every $t_1, t_2 \in T$. The partition $\{P_t\}_{t\in T}$ is termed *the ring partition of the design D.* For a design D, $cost(D) = |D|$, i.e., the number of lightpaths in the design.

The *minimum cost ring partition design* (MCRPD, in short) problem is formally defined as follows. The input is a graph G and a set of connections C in G. The goal is to find a ring partition design D for C that minimizes $cost(D)$. The corresponding decision problem is to decide for a set of connections C in G and a positive integer s whether there is a ring partition design D for C such that $cost(D) \leq s$.

MCRPD$_{\mathcal{G}}$ denotes the version of the problem in which the input is restricted to a family $\mathcal{G}$ of networks (e.g., the family $\mathcal{R}$ of rings).

Figure 1 is an example of the MCRPD problem, where $\boldsymbol{(a)}$ shows an instance with an initial set of size 4, and $\boldsymbol{(b)}$ shows a solution which consists of 2 rings and 3 new connections. The cost of the solution is thus 7.

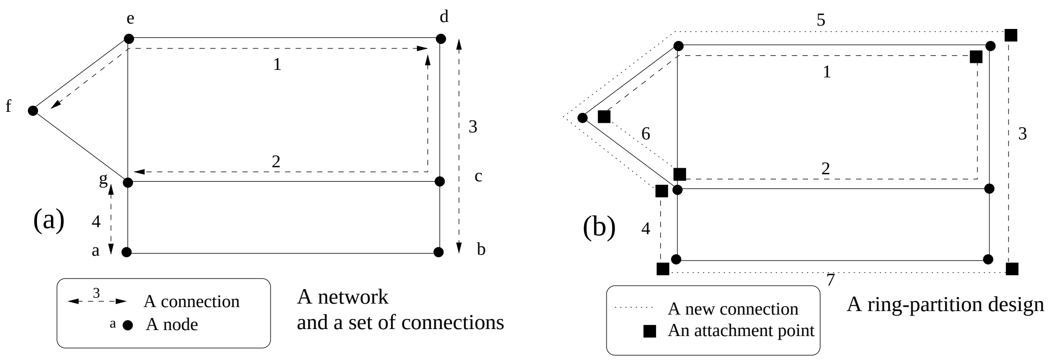

Figure 1: The MCRPD problem.

3 The MCRPD Problem

In this section we start our study of the MCRPD problem by providing some negative results regarding the tractability and approximability of the problem.

We say that a family of topologies $\mathcal{G} = G_1, G_2, \cdots$ has the *unbounded cycle* (UBC) property if there exists a constant k, such that for every n, there exists a graph $G_{i_n} \in \mathcal{G}$, with size $O(n^k)$, that contains a cycle of length n. Examples for families of topologies having the UBC property are the family $\mathcal{R}$ of ring topologies, and the family of complete graphs.

Theorem 1 *The $MCRPD_{\mathcal{G}}$ problem is NP-hard for every family of topologies $\mathcal{G}$ having the UBC property.*

Proof: In Appendix **A**. ∎

We continue by studying approximation algorithms for the MCRPD problem. A trivial approximation algorithm is achieved by adding for every connection c, a new disjoint connection between c's endpoints. Note that if there is no such route then there is no ring partition design for this instance. The resulting ring partition design will include virtual cycles, each with two connections, one of which belongs to the initial set C. For an algorithm A, we denote by $A(I)$ the value of a solution found by A for an instance I, and by $OPT(I)$ the value of an optimal solution. Clearly, $TRIV(I) = 2n \leq OPT(I) + n$, for every instance $I = (G, C)$ of MCRPD, where $|C| = n$. A question which arises naturally is whether there exists an approximation algorithm A for the MCRPD problem that guarantees, $A(I) \leq OPT(I) + n^{\alpha}$, for some constant $\alpha < 1$. We give a negative answer for this questions (for every constant $\alpha < 1$).

Theorem 2 *Let $\mathcal{G}$ be any family of topologies having the UBC property. Then for any constant $\alpha < 1$, $MCRPD_{\mathcal{G}}$ has no polynomial-time approximation algorithm A that guarantees $A(I) \leq OPT(I) + n^{\alpha}$ (unless $P = NP$).*

Proof: In Appendix **B**. ∎

The next question is whether there is an approximation algorithm A for MCRPD which guarantees $A(I) \leq OPT(I) + k \cdot n$, where $k < 1$ is a constant (clearly, the trivial algorithm $TRIV$ satisfies this bound for $k = 1$). In the sequel we answer this question positively for $k = \frac{3}{5}$.

4 A Ring Partition Approximation Algorithm

In this section we provide an approximation algorithm, termed *ring partition algorithm* (RPA, in short), for the MCRPD problem. We analyze RPA and show that it guarantees $RPA(I) \leq OPT(I) + \frac{3}{5} \cdot n$ for every instance I (where n is the number of connections in the initial set). We also study some special cases in which better results are achieved.

Unless stated otherwise we assume an arbitrary network topology $G = (V, E)$, where $V = \{v_1, \cdots, v_m\}$, and an initial set of connections C in G, where $|C| = n$. We assume that the route $\mathcal{R}(c)$ of every connection c in C is a sub-path in some simple cycle in G (observe that this assumption can be verified in polynomial time, and without it there is no ring partition design D for C).

4.1 Preliminary Constructions

We define some preliminary constructions that are used later for the definition of RPA. Recall that a virtual path P is a sequence $\langle v_1, c_1, v_2, c_2, \cdots, c_k, v_{k+1}\rangle$, where c_i is a connection with endpoints v_i and v_{i+1} (for $i = 1, \cdots, k$). P is termed a *virtual cycle* if $v_1 = v_{k+1}$. The pair of connections c_i and c_{i+1} are termed *attached at node* v_{i+1} in P (or simply, *attached in* P). If P is a virtual cycle then the pair c_1 and c_k are also considered *attached (at node* v_{k+1}*)* in P.

Let C be a set of connections in G, and let v be a node in G. We denote by $C(v) \subseteq C$ the set of connections for which v is an endpoint. Let Q be the symmetric binary relation over the set C of connections that is defined as follows. $(c_1, c_2) \in Q$ iff c_1 and c_2 are disjoint and there exists a simple cycle in G which contains both routes $\mathcal{R}(c_1)$ and $\mathcal{R}(c_2)$. Then Q defines an *end-node graph* $NG_v = (NV_v, NE_v)$ for every node v, where the set of nodes NV_v is $C(v)$, and NE_v is the set of edges, as follows. For every pair of connections $c_i, c_j \in C(v)$, $\{c_i, c_j\} \in NE_v$ iff $(c_i, c_j) \in Q$. A *matching* for a graph $G = (V, E)$ is a set $E' \subseteq E$ such that no two edges in E' share a common endpoint. A *maximum matching* is a matching of maximum size. We denote by *match*(G) the size of a maximum matching for G. A matching in an end-node graph NG_v, for a node v describes a set of attachments of pairs of connections (which satisfy Q) in v.

Consider a graph $G = (V, E)$, where $V = \{v_1, v_2, \cdots, v_m\}$, and a set of connections C in G. A *matching-set* for G and C is a set of matchings $\mathcal{E} = \{NE'_{v_1}, NE'_{v_2}, \cdots, NE'_{v_m}\}$, where $NE'_{v_i} \subseteq NE_{v_i}$ is a matching in the end-node graph NG_{v_i} (see Figure 2 as an example).

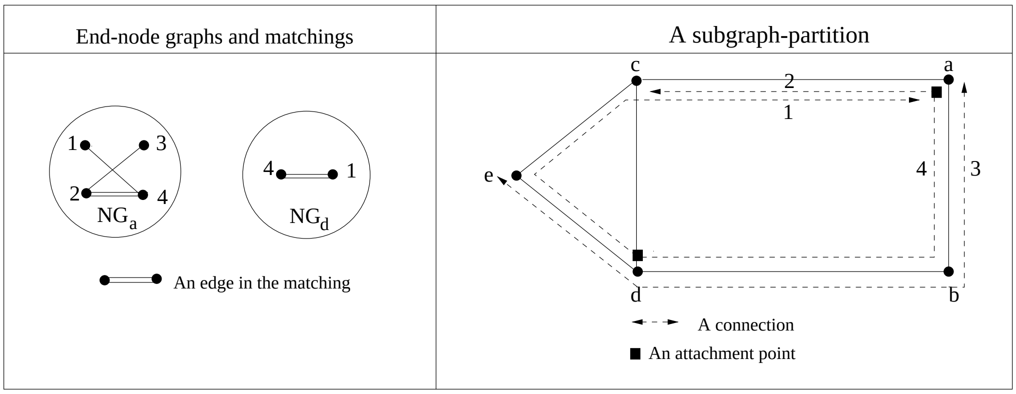

Figure 2: A graph, a set of connections, a matching-set (where only matchings in non-trivial end-node graphs are shown), and the equivalent subgraph-partition.

A *subgraph-partition* $\mathcal{G} = \mathcal{G}_p \cup \mathcal{G}_c$, for a set of connections C, is a partition of the connections in C into virtual paths and cycles (which are also termed *subgraphs*) as follows. Recall that $S(g)$ is the set of connections that are included in a virtual path (or cycle) g. $\mathcal{G}_p$ is a set of virtual paths, $\mathcal{G}_c$ is a set of virtual cycles, $C = \cup_{g \in \mathcal{G}} S(g)$, and $S(g_1) \cap S(g_2) = \emptyset$ for every $g_1, g_2 \in \mathcal{G}$. Note that the ring partition $\{P_t\}_{t \in T}$ of a ring partition design $D = \cup_{t \in T} S(P_t)$ is actually a subgraph-partition for D (where, $\mathcal{G} = \mathcal{G}_c$, $\mathcal{G}_p = \emptyset$). In general the virtual paths and cycles in a subgraph-partition might not be plain.

Note that there is a one-to-one correspondence between matching-sets and subgraph-partitions, as follows. Consider a matching-set $\mathcal{E} = \{NE'_{v_1}, NE'_{v_2}, \cdots, NE'_{v_m}\}$ and a subgraph-partition $\mathcal{G} = \mathcal{G}_p \cup \mathcal{G}_c$ for a set of connections C in G. $\mathcal{E}$ and $\mathcal{G}$ are termed *equivalent* if the following condition is satisfied. For every pair of connections $c_1, c_2 \in C$, there exists a subgraph g, $g \in \mathcal{G}$, such that c_1 and c_2 are attached at node v_i in g, iff $\{c_1, c_2\} \in NE'_{v_i}$.

For a matching-set $\mathcal{E}$ we denote by $\mathcal{G}_\mathcal{E}$ the (unique) equivalent subgraph-partition. Similarly, $\mathcal{E}_\mathcal{G}$ is the (unique) equivalent matching-set for a given subgraph-partition $\mathcal{G}$. Clearly, for a matching-set $\mathcal{E}$, $\mathcal{E}_{\mathcal{G}_\mathcal{E}} = \mathcal{E}$. As an example see Figure 2.

4.2 Ring Partition Algorithm (RPA)

We present a ring partition algorithm, called RPA, which finds a ring partition design for a set of connections C in G in four main stages. First, the end-node graph NG_{v_i} is constructed and a maximum matching in it is found for every node v_i, $i = 1, \cdots, m$. This defines a maximum matching-set $\mathcal{E}$. Then, the equivalent subgraph-partition $\mathcal{G} = \mathcal{G}_\mathcal{E}$ is constructed. Next, we partition every non-plain virtual path or virtual cycle in $\mathcal{G}$ to plain virtual paths. In addition, we make sure that for every virtual path $P \in \mathcal{G}$, there is a simple cycle in G in which $\mathcal{R}(P)$ is a sub-path. Last, the subgraph-partition is completed to a ring partition, by adding for every virtual path $P \in \mathcal{G}$, a connection which completes it to a plain virtual cycle. Following is the description of RPA followed by an informal description of the operations taken by its main functions.

$\boldsymbol{RPA(G, C)}$
$(\mathcal{G}_p, \mathcal{G}_c) := ConstructPartition(G, C)$
$(\mathcal{G}_p, \mathcal{G}_c) := AdjustPartition(\mathcal{G}_p, \mathcal{G}_c, G)$
$D := C \cup CompletePartition(\mathcal{G}_p, \mathcal{G}_c, G)$
return D

$\boldsymbol{ConstructPartition(G, C)}$
for every $i \in 1, \cdots, m$
    construct $NG_{v_i} = (NV_{v_i}, NE_{v_i})$
    find maximum matching $NE'_{v_i} \subseteq NE_{v_i}$
$\mathcal{E} := \{NE'_{v_1}, NE'_{v_2}, \cdots, NE'_{v_m}\}$
construct the equivalent subgraph-partition $\mathcal{G}_\mathcal{E} = (\mathcal{G}_p, \mathcal{G}_c)$
return $(\mathcal{G}_p, \mathcal{G}_c)$

$\boldsymbol{AdjustPartition(\mathcal{G}_p, \mathcal{G}_c, G)}$
for every $P \in \mathcal{G}_p \cup \mathcal{G}_c$
    $\mathcal{G}_c := \mathcal{G}_c \setminus \{P\}$ /* in case P is a cycle */
    $C_p := Partition(P)$
    $\mathcal{G}_p := (\mathcal{G}_p \setminus \{P\}) \cup C_P$
for every $P \in \mathcal{G}_P$
    if $(cycle(P))$ then
        $\mathcal{G}_P := \mathcal{G}_P \setminus \{P\}$
        $\mathcal{G}_C := \mathcal{G}_C \cup \{P\}$
return $(\mathcal{G}_p, \mathcal{G}_c)$

$\boldsymbol{CompletePartition(\mathcal{G}_p, \mathcal{G}_c, G)}$
$D' := \emptyset$
for every $P \in \mathcal{G}_P$
    $P^c := findDisjoint(P)$
    $D' := D' \cup \{P^c\}$
return D'

$\boldsymbol{Partition(P)}$
Assume that $P := \langle v_1, c_1, v_2, c_2, \cdots, v_l, c_l, v_{l+1} \rangle$
$C_P := \emptyset$; $first := 1$
for $i := 1$ to l

$P' := \langle v_{first}, c_{first}, \cdots, c_i, v_{i+1} \rangle$
   if ($\neg(plain(P') \wedge cycleExists(P'))$) then
      $C_P := C_P \cup \{\langle v_{first}, c_{first}, \cdots, c_{i-1}, v_i \rangle\}$
      $first := i$
return $C_P \cup \{\langle v_{first}, c_{first}, \cdots, c_l, v_{l+1} \rangle\}$

The function *ConstructPartition* first constructs the end-node graphs. The algorithm to construct the end-node graphs is straightforward and is not elaborated. It consists of determining, for every pair of connections with a common endpoint, whether they are disjoint, and whether the path that is formed by concatenating them can be completed to a simple cycle in G. This could be done using standard BFS techniques (see, e.g., [Eve79]). *ConstructPartition* then finds maximum-matchings in the end-node graphs. Efficient algorithms for finding maximum matchings in graphs can be found in, e.g., [MV80] (for a survey see [vL90], pages 580–588)). Last, the construction of the equivalent subgraph-partition is straightforward.

The function *AdjustPartition* partitions every virtual path and virtual cycle in the subgraph-partition using the function *Partition*. After the partition, every virtual path is plain and can be completed to a simple cycle in G. Every virtual path is then checked and if it is actually a cycle (i.e., its endpoints are equal) then it is inserted into $\mathcal{G}_c$.

The task of *Partition* is to partition a virtual path (or cycle) to a set $\{P_1, \cdots, P_l\}$ of plain virtual paths, s.t. for every P_i, $\mathcal{R}(P_i)$ is a sub-path in some simple cycle in G. The function *cycleExists*(P) returns *true* if there is a disjoint path in G between P's endpoints. The function *cycle*(P) returns *true* if the endpoints of a given virtual path are equal.

Last, the function *CompletePartition* completes every virtual path in $\mathcal{G}_p$ to a virtual cycle by adding a new disjoint connection P^c between P's endpoints.

4.3 Correctness and Analysis

We first present four observations that are used for the proof of the main theorem (Theorem 3). Observation 1 shows a connection between the sizes of matching-sets and the equivalent subgraph-partitions.

Observation 1 *Let $\mathcal{E} = \{NE'_{v_1}, NE'_{v_2}, \cdots, NE'_{v_m}\}$ be a matching-set for a set of connections C in $G = (V, E)$, where $|C| = n$, and $V = \{v_1, \cdots, v_m\}$. Let $\mathcal{G}_{\mathcal{E}} = \mathcal{G}_p \cup \mathcal{G}_c$ be the equivalent subgraph-partition. Then $|\mathcal{G}_p| = n - \sum_{i=1}^{m} |NE'_{v_i}|$.*

Proof: Let an *attachment point* in $\mathcal{G}_{\mathcal{E}}$ be an ordered pair $(\{c_1, c_2\}, v)$, where the connections c_1 and c_2 are attached at node v in some subgraph $g \in \mathcal{G}_{\mathcal{E}}$. Clearly the number of unique attachment points in a virtual path $P_x \in \mathcal{G}_p$ is one less than the number of connections in P_x, i.e., $|S(P_x)| - 1$. The number of unique attachment points is equal to $|S(P_x)|$ if $P_x \in \mathcal{G}_c$ is a virtual cycle. It follows that the number of unique attachment points is equal to $(\sum_{g \in \mathcal{G}_{\mathcal{E}}} |S(g)|) - |\mathcal{G}_p| = n - |\mathcal{G}_p|$. Now by the definitions there is a one-to-one correspondence between attachment points and edges in the matchings. It follows that the number of attachment points is equal to the number of edges in the matching set, i.e, $n - |\mathcal{G}_p| = \sum_{i=1}^{m} |NE'_{v_i}|$. ■

Let $\mathcal{G}(D)$ be a subgraph-partition for a set of connections D. The *projection* $\mathcal{G}(D)|_C$ of $\mathcal{G}(D)$ on a set of connections $C \subseteq D$ is a subgraph-partition for C which is obtained from $\mathcal{G}(D)$ by deleting all the connections that are not in C (i.e., all the connections in $D \setminus C$). Note that a virtual path (or cycle) in $\mathcal{G}(D)$ might be cut by this process into few virtual paths. Similarly, let $\mathcal{E}(D)$ be a matching-set for D. Then the *projection* $\mathcal{E}(D)|_C$ of $\mathcal{E}(D)$ on a set of connections $C \subseteq D$, is a matching-set for C which is obtained from $\mathcal{E}(D)$ by deleting from the end-node graphs (and the matchings) nodes which correspond to connections in $D \setminus C$ and the edges that meet them. Clearly, if $\mathcal{G}(D)$ and $\mathcal{E}(D)$ are equivalent then so are $\mathcal{G}(D)|_C$ and $\mathcal{E}(D)|_C$.

Consider a ring partition design $D = \cup_{t \in T} S(P_t)$ for a set of connections C. We denote by $\mathcal{G}(D)$ the ring partition $\{P_t\}_{t \in T}$ of D, and by $\mathcal{E}(D)$ the equivalent matching-set for D (i.e., $\mathcal{E}(D) = \mathcal{E}_{\mathcal{G}(D)}$). The subgraph-partition $\mathcal{G}(D)|_C$ and the matching-set $\mathcal{E}(D)|_C$ for the initial set of connections C are termed *the induced subgraph-partition* and *the induced matching-set*, respectively (note that they are equivalent). Observation 2 associates the cost of ring partition designs, with the sizes of the induced matching-sets and subgraph-partitions.

Observation 2 *Let $D = \cup_{t\in T} S(P_t)$ be a ring partition design for a set of connections C in a physical topology $G = (V, E)$, where $|C| = n$, and $|V| = m$. Let $\mathcal{E}(D)|_C = \{NE'_{v_1}, NE'_{v_2}, \cdots, NE'_{v_m}\}$ and $\mathcal{G}(D)|_C = \mathcal{G}_p \cup \mathcal{G}_c$ be the induced matching-set and subgraph-partition for C. Then $cost(D) \geq n + |\mathcal{G}_p| = 2n - \sum_{i=1}^{m} |NE'_{v_i}|$.*

Proof: By the definitions, $Cost(D) = \sum_{t\in T} S(P_t)$. Let $new(P_t)$ be the number of new connections in the virtual cycle P_t, i.e., $new(P_t) = S(P_t) \cap (D \setminus C)$. Clearly, $Cost(D) = n + \sum_{t\in T} new(P_t)$. Consider now the induced subgraph partition $\mathcal{G}(D)|_C = \mathcal{G}_p \cup \mathcal{G}_c$. Recall that it is obtained from D by deleting all the new connections. In this process a virtual cycle in the ring-partition might be cut into few virtual paths. Clearly the number of such virtual paths for each virtual cycle, is at most the number of new connections in it. It follows that $|\mathcal{G}_p| \leq \sum_{t\in T} new(P_t)$, thus $Cost(D) \geq n + |\mathcal{G}_P|$. by Observation 1, $n + |\mathcal{G}_p| = 2n - \sum_{i=1}^{m} |NE'_{v_i}|$. Note that strict inequality occurs when two new connections are attached in one of the virtual cycles. ■

A *maximum-matching-set*, is a matching set $\mathcal{E} = \{NE'_{v_1}, \cdots, NE'_{v_m}\}$ for a set of connections C, s.t. the matching NE'_{v_i} is a maximum matching for the end-node graph NG_{v_i}, for every $i = 1, \cdots, m$. Recall that $match(G)$ is the size of a maximum matching for G. Observation 3 is a lower bound on the value of an optimal solution.

Observation 3 *Every ring partition design D for C satisfies $cost(D) \geq 2n - \sum_{i=1}^{m} match(NG_{v_i})$ (where, n and m are defined as above).*

Proof: Let $D = \cup_{t\in T} S(P_t)$ be a ring-partition design for C. Note that every two connections that are attached in a virtual cycle P_t, $t \in T$, in the design satisfy the relation Q, i.e., they are disjoint and there is a simple cycle that contains both routes. Clearly, the same holds also for the induced subgraph-partition $\mathcal{G}(D)|_C$ and matching-set (since we only delete connections). Consider the equivalent matching set $\mathcal{E}(D)|_C = \{NE'_{v_1}, NE'_{v_2}, \cdots, NE'_{v_m}\}$. In follows that NE'_{v_i} is actually a matching in the end-node graph NG_{v_i}, for $i = 1, \cdots, m$, and thus $|NE'_{v_i}| \geq match(NE_{v_i})$. It follows, by Observation 2, that $Cost(D) \geq 2n - \sum_{i=1}^{m} |NE'_{v_i}| \geq 2n - \sum_{i=1}^{m} match(NG_{v_i})$. ■

Consider a ring partition design $D = \cup_{t\in T} S(P_t)$ for a set of connections C in G. Let $new(P_t)$ be the number of new connections in $S(P_t)$ (i.e., connections in $S(P_t) \cap (D \setminus C)$). A *canonical* ring partition design satisfies that $new(P_t) \leq 1$ for every $t \in T$. Note that it is always possible to construct from a given ring partition design D, a canonical ring partition design D' such that $Cost(D') \leq Cost(D)$ as follows. Let $\mathcal{G}(D)|_C = \mathcal{G}_p \cup \mathcal{G}_c$ be the induced subgraph partition of D. To construct a canonical ring partition design D' with at most the same cost we complete every virtual path in $\mathcal{G}_p$ to a plain virtual cycle by adding one new connection. (This is always doable since every virtual path in $\mathcal{G}_p$ is plain and is included in some simple cycle in G). By the discussion above, $Cost(D') = n + |\mathcal{G}_p| \leq Cost(D)$. Observation 4 follows.

Observation 4 *If there is a ring partition design for a set of connections C in G then there is a canonical ring partition design with minimum cost.*

It can be proved that Observation 2 holds for canonical ring partition designs D' with equality i.e., $cost(D') = n + |\mathcal{G}_p|$. It is therefore sometimes convenient to consider for simplicity only canonical ring partition designs.

We are now ready to prove the main theorem.

Theorem 3 *$RPA(I) \leq OPT(I) + \frac{3}{5} \cdot n$, for every $I = (G, C)$, where $|C| = n$.*

Sketch of Proof: For the analysis we denote by $\mathcal{G}_p^1$ and $\mathcal{G}_c^1$ the sets $\mathcal{G}_p$ and $\mathcal{G}_c$ right after the execution of *ConstructPartition*, and by $\mathcal{G}_p^2$ and $\mathcal{G}_c^2$ the corresponding sets right after the execution of *AdjustPartition*.

We now examine the partition procedure *Partition*. Recall that the end-node graphs are constructed w.r.t. the relation Q which is *true* for a pair of connections c_1 and c_2 iff their routes $\mathcal{R}(c_1)$ and $\mathcal{R}(c_2)$ are disjoint and there is a simple cycle which contains both routes (as sub-paths). Consider a virtual path $P = \langle v_1, c_1, v_2, c_2, \cdots, v_{l-1}, c_l, v_l \rangle \in \mathcal{G}_p^1$. Since P is a virtual path in the equivalent subgraph-partition $\mathcal{G}_{\mathcal{E}}$, it holds that $(c_i, c_{i+1}) \in Q$, for every $i = 1, \cdots, l-1$. Let C_P be the set of virtual paths which is the output of $Partition(P)$. By the above discussion, and by the definition of *Partition*, at most one virtual path in C_P contains less than two connections. Such a virtual path can be only the last one, which contains the connection c_l. Let $n_P = |S(P)|$ (i.e., the number of connections in the virtual path P). Let $m_P = |C_P|$ (i.e., the number of plain virtual paths that are the result of applying the partition procedure on P). It follows that $m_P \leq \lfloor \frac{n_P+1}{2} \rfloor$.

Now consider a non-plain virtual cycle $P \in \mathcal{G}_c^1$. Then, by the same considerations, $m_P \leq \lfloor \frac{n_P+1}{2} \rfloor$, where n_P and m_P are defined similarly.

Let $\mathcal{G}_c' \subseteq \mathcal{G}_c^1$ and $\mathcal{G}_c'' \subseteq \mathcal{G}_c^1$ be the sets of non-plain virtual cycles with, respectively, odd and even number of connections, after *ConstructPartition*. Note that *CompletePartition* adds one new connection for every virtual path $P \in \mathcal{G}_p^2$. We get,

$$\begin{aligned} RPA(I) &= |\mathcal{G}_p^2| + n \\ &\leq \textstyle\sum_{P \in \mathcal{G}_c'} (\frac{n_P}{2} + \frac{1}{2}) + \sum_{P \in \mathcal{G}_c''} (\frac{n_P}{2}) + \sum_{P \in \mathcal{G}_p^1} (\frac{n_P}{2} + \frac{1}{2}) + n \\ &\leq \tfrac{3n}{2} + \tfrac{1}{2}|\mathcal{G}_c'| + \tfrac{1}{2}|\mathcal{G}_p^1| \end{aligned}$$

Observe that a non-plain virtual cycle in $\mathcal{G}_P^1$ contains at least 4 connections, since otherwise clearly there are two consecutive connections that are not disjoint in the cycle, which is not possible by the definition of the algorithm. It follows that $|\mathcal{G}_c'| \leq \frac{n}{5}$. We get, $RPA(I) \leq n + \frac{3}{5} \cdot n + \frac{1}{2}|\mathcal{G}_p^1|$. Now, by Observation 3, we can show that $OPT(I) \geq n + |\mathcal{G}_p^1|$ (since in the first step RPA finds maximum matchings in the end-node graphs). Thus, $RPA(I) \leq OPT(I) + \frac{3}{5} \cdot n$. ■

Note that since $OPT(I) \geq n$, this is actually better than a $\frac{8}{5}$-approximation.

Time complexity. The time complexity depends on the exact format of the input for the algorithm and the data structures which are used in order to represent the physical topology, the set of connections and the auxiliary combinatorial constructions (i.e., the end-node graphs, and the subgraph partition). It is clear however that this time is polynomial in the size of C and G. It is well-known that it takes $O(\sqrt{|V|} \cdot |E|)$ time to find a maximum matching in a graph $G = (V, E)$ ([MV80]) and that it takes $O(|E|)$ time to find whether two paths are disjoint, or whether there exists a disjoint path between a given path's endpoints. For special topologies these tasks can be significantly simpler. For instance, clearly in the ring physical topology case, every plain virtual path can be completed to a plain virtual cycle, thus the relation Q can be simplified to $Q(c_1, c_2) = disjoint(c_1, c_2)$. The end-node graphs are bipartite, and finding maximum matchings in bipartite graphs is considerably easier ([vL90]). Also, to find a disjoint path between the endpoints of a given simple path is trivial. In any case, for the applications of RPA for the design of optical networks time-efficiency is not crucial since the algorithm is applied only in the design stage of the network and it is reasonable to invest some preprocessing time once in order to achieve better network designs.

4.4 Special Cases

4.5 Optimal Cases

Since the MCRPD problem is NP-hard (Theorem 1) it is natural to try and find restricted families of topologies for which it can be solved in polynomial time. Unfortunately, we actually proved in Theorem 1 that the MCRPD problem is NP-hard for every family of topologies that contains cycles with unbounded length (e.g., rings). Since trees do not support ring partition designs, this implies that the problem is NP-hard for every family of topologies which is of interest in this setting. This observation motivates the question of finding polynomially solvable classes of instances of the problem when taking into account not only the topology of the network but also the initial set of connections.

The *induced graph* $IG_C = (IV_C, IE_C)$ for a set of connections C in G is the subgraph of G which includes all the edges and nodes of G that are used by at least one connection in C.

A natural question is whether applying restrictions on the induced graph suffices to guarantee efficient optimal solution to the problem. We answer this question negatively by showing that the problem remains NP-hard even for the most simple case where the induced graph is a chain.

Theorem 4 *The MCRPD problem is NP-hard even if the induced graph for the set of connections C in G is a chain (or a set of chains).*

Next we show that if, in addition to an induced graph with no cycles, the network topology satisfies a certain condition (w.r.t. the initial set of connections), then RPA finds a minimum cost ring partition design.

Theorem 5 *$RPA(I) = OPT(I)$ for every instance $I = (G, C)$ which satisfies the following two properties.*

- ***No Cycles.*** *The induced graph $IG_C = (IV_C, IE_C)$ is a forest.*
- ***Completion.*** *For every plain virtual path P over C, there is a simple cycle in G that contains the route of P, $\mathcal{R}(P)$, as a sub-path.*

We discuss below some cases in which the conditions in Theorem 5 are satisfied. A perfectly-connected graph (PC, in short) satisfies that every simple path in it is included in a simple cycle. Clearly, if a graph is perfectly connected than the completion property is satisfied for every initial set of connections. This property also guarantees that there is a ring partition design D for *every* initial set of connections C. A natural question is to characterize perfectly connected graphs. We give a full characterization of perfectly connected graphs by proving that a graph is PC iff it is randomly Hamiltonian. Randomly Hamiltonian graphs are defined and characterized in [CK68].

Theorem 6 *A graph G is perfectly connected iff it is one of the following: a ring, a complete graph, or a complete bipartite graph with equal number of nodes in both sets.*

We note that RPA does not have to be modified in order to give an optimal result for instances which satisfy the conditions in Theorem 5. However, we can benefit from recognizing in advance such instances since in these cases the procedure *AdjustPartition* can be skipped. The Recognition can be done easily for specific topologies (e.g., rings), and in polynomial time in the general case.

4.6 Bounded Length Connections in Rings

We analyze the performance of RPA in the case of a ring physical topology, when there is a bound on the length of connections in the initial set.

Theorem 7 *$RPA(I) \leq OPT(I) + \frac{3k}{2m} \cdot n$, for every instance $I = (R_m, C)$ of $MCRPD_{\mathcal{R}}$, if for every connection $c \in C$, $length(\mathcal{R}(c)) \leq k$, for any constant k, $1 \leq k \leq m-1$.*

Note that RPA does not guarantee that the same bound on the length holds also for connections in the ring partition design which is constructed. Indeed, the case where the length of connections in the solution must be bounded is inherently different, and the main results in this paper do not hold for it.

4.7 Approximations Based on the Load

Let the *load* l_e of an edge $e \in E$ be the number of connections in C which use e, and $l_I = \max_{e \in E} l_e$. Recall the definition of an induced graph $IG_C = (IV_C, IE_C)$ for a set of connections C in G (Section 4.5). We add to this definition a weight function $w : IE_C \rightarrow N$ that assigns a weight for every edge that is equal to its load. Although in the worst case the load of an instance is equal to the number of connections $|C|$, usually it is substantially smaller. Therefore, it is interesting to bound the cost of a design as a function of the load.

For this purpose, we assume that the route of every virtual path is a sub-path is some simple cycle in G (i.e., the completion property). Let $W = \sum_{e \in E} l_e$. Now consider the weighted induced graph $IG_C = (IV_C, IE_C, W_C)$ for C. Let T_{max} be a maximum-weight spanning tree in IG_C, $W_{T_{max}} = \sum_{e \in T_{max}} l_e$, and $W_{G-T_{max}} = W - W_{T_{max}}$. Following is a description of a modified version of RPA, termed RPA$_l$. We temporarily remove all connections that use edges that are not in T_{max}. Next, we find a ring partition design for the remaining set of connections (using RPA). Last, we insert back the removed connections and complete each one of them to a virtual cycle by adding a new connection. We prove that the cost of the resulting ring partition design is larger by at most $2W_{G-T_{max}}$ than the optimal one. (Note that an improved heuristics might be to repeat the same process with the remaining set of connections.)

Theorem 8 *$RPA_l(I) \leq OPT(I) + 2W_{G-T_{max}}$, for every instance $I = (G, C)$ which satisfies the completion property.*

For the case of a ring physical topology, it holds $RPA_l(I) \leq OPT(I) + \min_{e \in E} l_e$. A slightly better bound is given for this case in [GLS98].

Note that there might be a set of connections C'_{min} with size smaller than $W_{G-T_{max}}$ such that the induced graph for the remaining set $C \setminus C'_{min}$ is a forest. However, we prove in Proposition 1 that finding a minimum set of connections whose removal leaves us with an induced graph with no cycles is NP-hard.

Proposition 1 *Finding a minimum set of connections $C' \subseteq C$ in a graph G such that the induced graph for the remaining set $C \setminus C'$ does not contain cycles is NP-hard.*

5 Summary and Future Research

In this paper we studied the MCRPD problem for which the input is an initial set of lightpaths in a network and the goal is to augment this set by adding lightpaths such that the result is a ring partition design with minimum cost. We have shown an approximation algorithm for this problem that guarantees $Cost(D) \leq OPT + k \cdot n$, where $k = \frac{3}{5}$, n is the number of lightpaths in the initial set, and OPT is the cost of an optimal solution. Moreover, we have shown that, unless $P = NP$, there is no approximation algorithm A for this problem that guarantees $Cost(D) \leq OPT + n^{\alpha}$, for every constant $\alpha < 1$. The main open question here is whether the constant k can be improved.

Ring partition designs are necessary for the near term future of optical networks since they support a SONET higher layer network which is configured in the form of rings. However it is claimed that the core network architecture will have to change and that SONET will give way to a smart optical layer. In this case other less restrictive survivability conditions might be considered. For example, it might be enough that every lightpath will have an alternative disjoint virtual path between its endpoints. While less restrictive survivability conditions might be less expensive to implement, the price to pay is of a more complex protection mechanism that is executed for every failure. The challenge here is two folded. First, to study the gain in the cost of the network when less restrictive survivability conditions are considered. Second, to study the algorithmic and technological issues of implementing protection mechanisms in the optical domain based on these conditions.

Acknowledgment We would like to thank Ornan (Ori) Gerstel for introducing us to this problem and for very helpful discussions.

References

[AA98] M. Alanyali and E. Ayanoglu. Provisioning algorithms for WDM optical networks. In *Proc. IEEE INFOCOM '98*, pages 910–918, 1998.

[ACB97] J. Armitage, O. Crochat, and J. Y. Le Boudec. Design of survivable WDM photonic network. In *Proc. IEEE INFOCOM '97*, pages 244–252, 1997.

[CK68] G. Chartrand and H. V. Kronk. Randomly traceable graphs. *SIAM J. Appl. Math.*, 16:696–700, 1968.

[Eve79] S. Even. *Graph Algorithms.* Computer Science Press, Woodland Hills, CA, 1979.

[GJMP80] M. Garey, D. Johnson, G. Miller, and C. Papadimitriou. The complexity of coloring circular arc graphs and chords. *SIAM Journal of Discrete Math*, 1(2):216–227, 1980.

[GLS98] O. Gerstel, P. Lin, and G. Sasaki. Wavelength assignment in WDM ring to minimize cost of embedded SONET rings. In *Proc. IEEE INFOCOM '98*, pages 94–101, 1998.

[GRS97] O. Gerstel, R. Ramaswami, and G.H. Sasaki. Fault tolerant multiwavelength optical rings with limited wavelength conversion. In *Proc. IEEE INFOCOM '97*, pages 507–515, 1997.

[HNS94] Y. Hamazumi, N. Nagatsu, and K. Sato. Number of wavelengths required for optical networks with failure restoration. In *Optical Fiber Communication*, pages 67–68, Febuary 1994.

[MV80] S. Micali and V.V. Vazirani. An $O(\sqrt{V} \cdot E)$ algorithm for finding maximum matching in general graphs. In *Proc. 21st Ann. Symp. Foundations of Computer Science*, pages 17–27, 1980.

[RS97] R. Ramaswami and A. Segall. Distributed network control for optical networks. *IEEE/ACM Transactions on Networking*, 5(6):936–943, 1997.

[RS98] Rajiv Ramaswami and Kumar N. Sivarajan. *Optical Networks: A Practical Perspective.* Academic Press/ Morgan Kaufmann, 1998.

[vL90] J. van Leeuwen, editor. *Handbook of Theoretical Computer Science*, volume A, chapter 10. The MIT Press, 1990.

Appendix A: Proof of Theorem 1

We prove Theorem 1 in two steps. In Step 1 we prove that MCRPD$_{\mathcal{R}}$ is NP-hard and in Step 2 we extend the result to every family of topologies with the unbounded cycle (UBC) property.

Step 1: We prove that the MCRPD$_{\mathcal{R}}$ problem is NP-hard by a polynomial transformation from the *circular arc coloring problem* which is known to be NP-hard ([GJMP80]).

A graph G is termed a *circular arc graph* if its nodes can be placed in a one-to-one correspondence with a set $\mathcal{F}$ of routes (paths) in a ring in such a way that two nodes of G are joined by an edge iff the corresponding two routes intersect.

The circular arc coloring problem is formalized as follows (definitions are adapted from definitions in this paper). A *set* $\mathcal{F}$ of circular arcs in a ring R_m is a set $\{A_0, A_1, \cdots, A_{n-1}\}$, where each A_i is an ordered pair (a_i, b_i) of positive integers, with $a_i \neq b_i$. For our needs, there is no difference between circular arcs and connections. Note that we can assume also in this case that $m \leq 2n$. The circular arc coloring problem is formally defined as follows. Given a pair $I = (R_m, \mathcal{F})$, where $\mathcal{F}$ is a set of circular arcs in the ring R_m, and a positive integer K, can $\mathcal{F}$ be partitioned into K classes so that no two arcs in the same class intersect?

Recall that load l_e of an edge e is the number of connections (or circular arcs) in the set C which use e, and let l_I be the maximum load of an edge (for a given instance $I = (G, C)$). We say that an instance I is *full* if it satisfies $l_e = l_I$ for every edge. Given an instance $I = (R_m, C)$, for every $t \geq l_I$, $full_t(I) = (R_m, C \cup C')$ is a full instance with load t, which is constructed from I by adding connections of length one. Formally, $C' = \cup_{x=0}^{m-1} C_x$, where $C_x = \cup_{i=1}^{t-l_{\{x,x+1\}}} \{(x, x+1)\}$, for every $x = 0, \cdots m-1 \pmod{m}$.

For an instance $I = (R_m, \mathcal{F})$ of the circular arc coloring problem and a positive integer K, we construct the instance $full_K(I) = (R_m, C)$ of the MCRPD$_\mathcal{R}$ problem, where $C = \mathcal{F} \cup C'$ (and the set C' contains the appropriate connections of length 1). Note that we can assume that K satisfies $l_C \leq K \leq |\mathcal{F}|$ since otherwise the answer for this instance of the circular arc coloring problem can be determined immediately. Note that $|C| \leq K \cdot m$, thus the construction is polynomial (since $K \leq |\mathcal{F}|$ and $m \leq 2 \cdot |\mathcal{F}|$). See Figure 3 for an example of the transformation.

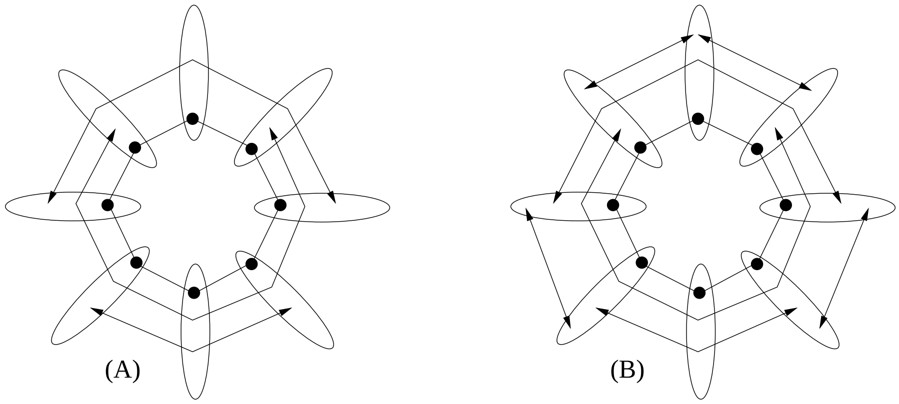

Figure 3: (A) An instance I of the circular arc coloring problem. (B) The instance $full_2(I)$ of the MCRPD$_\mathcal{R}$ problem.

Proposition 2 *The set $\mathcal{F}$ can be partitioned into K classes of pair-wise non-intersecting arcs iff there is a ring partition design D for C with $cost(D) = |C|$.*

Proof: In one direction, assume that there is a ring partition design $D = \cup_T S(C_t)$ for C such that $cost(D) = |C|$. Clearly, this situation where the cost of a design is equal to the size of the set of connections C (i.e., no new connections are added) is possible only if the instance is full (as is the case with $full_K(I)$) and only if the number of rings in D is equal to the load of the instance, i.e., $|T| = K$, in our case. Such a solution D induces a partition of $\mathcal{F}$ to K classes of pairwise non-intersecting arcs in the obvious way, since routes of connections in the same virtual cycle C_t do not intersect.

In the other direction, consider a partition of $\mathcal{F}$ to K classes $\{C_i\}_{i=1}^k$ such that in every class all the arcs do not intersect. Since $full_K(I)$ is a full instance with load K, and since C' contains only paths of length one, it can be easily seen that from such a partition of $\mathcal{F}$ into K classes one can construct a ring partition of C simply by completing every class to a cycle by using paths (of length 1) from C'. We get a ring partition design D for C with $cost(D) = |C|$. ■

Step 2: Let $\mathcal{G}$ be a family of topologies with the UBC property. We prove that the MCRPD$_\mathcal{G}$ problem is NP-hard by a polynomial transformation from the MCRPD$_\mathcal{R}$(which is NP-hard by Step 1).

Let $I = (R_m, C)$ be an instance of the MCRPD$_\mathcal{R}$ problem. Let $G_{i_m} \in \mathcal{G}$ be the graph in $\mathcal{G}$ that contains a cycle of length m (by the definition of the UBC property, such graph G_{i_m} exists in $\mathcal{G}$).

We construct an instance $I' = (G_{i_m}, C')$ of the MCRPD$_{\mathcal{G}}$ problem in the obvious way such that there is a one-to-one corresponds (which preserves the intersection pattern) between paths from C in R_m and paths in the ring R'_m (of size m) which is a sub-graph of $\mathcal{G}_{i_m}$.

Clearly, for every ring partition design D for C there is a ring partition design D' for C' with the same cost. In the other direction, let $D' = \cup_{t\in T} S(P'_t)$ be a ring partition design for C'. Note that, by the construction, all connection in the set C' have routes which are paths in the ring R'_m, but new connections might have any route in G_{i_m}. It is clear that we can construct form D' a ring partition design D such that *all* connections in D have routes on the ring and $Cost(D) \leq Cost(D')$ as follows. We can take the induced subgraph partition of D' (see Section 4.3) and complete every virtual path in it (whose route must be a path on the ring) to a virtual cycle by adding one new connection with the complement route on the ring. By the observations in Chapter 4.3, $Cost(D) \leq Cost(D')$.

We note that the definition of the UBC property can be relaxed so as to include families of topologies in which for every $n > 0$ there is a graph of size $\Theta(n^{k_1})$ that includes a cycle of length $\Theta(n^{k_2})$, for some constants K_1, K_2. The same proof holds for the relaxed definition with slight technical extensions. ∎

Appendix B: Proof of Theorem 2

We prove Theorem 2 for the case of a rings. The proof is generalized to every family of topologies having the UBC property by using the same transformation as in Step 2 in the proof of Theorem 1. Assume to the contrary that there is a constant $\alpha < 1$ and a polynomial approximation algorithm A such that for every instance I of the MCRPD$_{\mathcal{R}}$ problem $A(I) \leq OPT(I) + n^{\alpha}$. We show a polynomial algorithm A' that optimally solves the MCRPD$_{\mathcal{R}}$ problem, in contradiction with Theorem 1.

Given an instance $I = (R_m, C)$, (where $|C| = n$) of the MCRPD$_{\mathcal{R}}$ problem we construct a new instance $I' = (R_{m'}, C')$, where $|C'| = n \cdot n^{\beta}$, $m' = m \cdot n^{\beta}$, and β is a constant whose value will be determined later, as follows. The ring $R_{m'}$ is divided into n^{β} sections $J_0, \cdots, J_{n^{\beta}-1}$ each of size m. The section J_t contains the interval of nodes $m \cdot t, \cdots, (m+1) \cdot t - 1$, for every $t = 0, \cdots, n^{\beta}$. The set of connections C' is the union of n^{β} sets of connections, where all the sets are isomorphic to the original set C but shifted in steps of m to different sections on the ring. Formally, $C' = \cup_{j=0}^{n^{\beta}-1} C^j$, where $C^j = \{(a_i^j, b_i^j) | i = 0, \cdots, n-1\}$, $a_i^j = a_i + j \cdot m$, and $b_i^j = b_i + j \cdot m$, for every $j \in 0, \cdots, n^{\beta} - 1$. See Figure 4 for an example of the construction.

Recall that a canonical ring partition design satisfies that the number of new connections in every virtual ring is at most 1. By the discussion in Section 4.3, we can consider w.l.o.g. only canonical ring partition designs. Consider a canonical ring partition design $D = \cup_T S(P_t)$ for C'. Note that in canonical ring partition designs every attachment point of two connections in a virtual ring is an end point of a connection in the original set. Since every connection in I' has its two endpoints in one of the (disjoint) sections of size m on the ring, all the connections in every virtual cycle P_t, $t \in T$, have their endpoints in the same section. Formally,

Observation 5 *Consider a canonical ring partition design $D = \cup_T S(C_t)$ for the instance $I' = (R_{m'}, C')$. For every virtual cycle C_t, $t \in T$, there exists j, $j \in 0, \cdots, n^{\beta} - 1$, such that every connection $c = (x, y) \in S(C_t)$ satisfies $j \cdot m \leq x, y \leq j \cdot m + (m - 1)$.*

Proof: By the above discussion. ∎

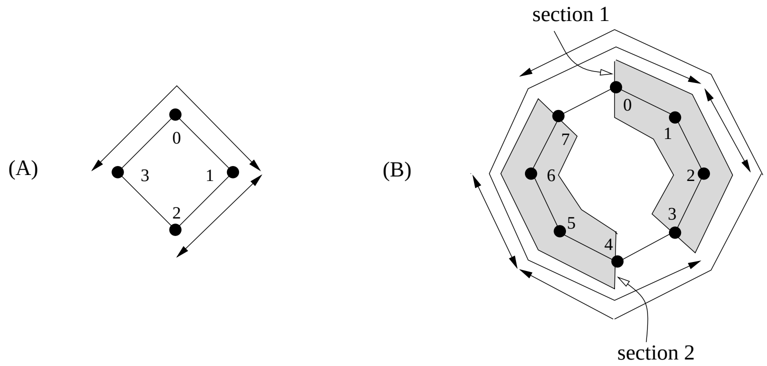

Figure 4: (a) The input instance I, where $m = 4$, $n = 2$, and $C = \{(1,2),(3,1)\}$. (b) The new instance I', where $\beta = 1$, i.e., a multiplication by 2.

It follows that D can be partitioned into n^β sets of virtual cycles $\mathcal{C}_j$, $j = 0, \cdots, n^\beta$, according to the endpoints of their connections; the family $\mathcal{C}_j$ includes all virtual cycles with endpoints in the jth section on the ring. Every family $\mathcal{C}_j$ naturally induces a canonical ring partition design D_j to the original set C as follows. $D_j = \{(a,b) \mid (m \cdot j + a, m \cdot j + b) \in S(P_t), P_t \in \mathcal{C}_j\}$.

Claim 1 $OPT(I') = OPT(I) \cdot n^\beta$.

Proof: On one direction one can easily see that from a canonical ring partition design D to the original set C we can construct a (canonical) ring partition design D' for C' such that $Cost(D') = Cost(D) \cdot n^\beta$ simply by multiplying it by n^β in the same manner as we constructed the instance I' from I. Thus, $OPT(I') \leq OPT(I) \cdot n^\beta$.

On the other direction, by Observation 4, there is an optimal canonical ring partition design $D' = \cup_T S(P_t)$ to C'. Such design can be partitioned, as explained above, to n^β subsets of connections. We have to show that $D_j = \{(a,b) \mid (m \cdot j + a, m \cdot j + b) \in S(P_t), P_t \in \mathcal{C}_j\}$ is indeed a ring partition design for I. But this is clear from the construction, since for every connection $c = (a,b) \in C$, there is a connection $c' = (m \cdot j + a, m \cdot j + b) \in C'$. Thus, in D' there is a virtual cycle P_x, s.t. $c' \in S(P_x)$ and by the discussion above, since D' is canonical, $P_x \in \mathcal{C}_j$. It follows that D_j is a ring partition design for the original set C. Now, since $\sum_{i=0}^{n^\beta} Cost(D_j) = Cost(D')$, we get $OPT(I) \cdot n^\beta \leq OPT(I')$. ■

The idea of the algorithm A' that we construct from A should be clear by now. Given an instance $I = (R_m, C)$ of the MCRPD$_\mathcal{R}$ problem, the algorithm A' first construct the instance $I' = (R_{m'}, C')$. It then execute A on the new instance. Let D' be the ring partition design found by A and let us assume that D' is canonical (otherwise A' can construct in polynomial time a canonical ring partition design from D'). Then D' induces n^β ring partition designs $D_0, \cdots D_{n^\beta - 1}$ for C such that $\sum_{j=0}^{n^\beta - 1} Cost(D_j) = Cost(D')$. The output of A' is a ring partition design D_t for C, $t \in 0, \cdots, n^\beta - 1$, s.t. $Cost(D_t) = Min_{j=0}^{n^\beta - 1} Cost(D_j)$. Clearly, for a constant β, A' runs in time polynomial in the size of I.

Claim 2 $A'(I) \cdot n^\beta \leq A(I')$.

Proof: Immediate from the discussion, since $A'(I) = Cost(D_t) = Min_{j=0}^{n^\beta - 1} Cost(D_j)$, and $A(I') = Cost(D') = \sum_{j=0}^{n^\beta - 1} Cost(D_j)$. ■

Now,

$$\begin{aligned} A'(I)\cdot n^{\beta} &\leq A(I') \\ &\leq OPT(I') + (n\cdot n^{\beta})^{\alpha} \\ &= OPT(I)\cdot n^{\beta} + (n\cdot n^{\beta})^{\alpha} \end{aligned}$$

We get,

$$A'(I) \leq OPT(I) + n^{(\beta+1)\alpha-\beta}$$

Fixing β such that $\beta > \frac{\alpha}{1-\alpha}$, we get $n^{(\beta+1)\alpha-\beta} < 1$, thus A' actually finds an optimal solution for every instance I, in contradiction with Theorem 1. We conclude that there is no constant $\alpha < 1$ for which there is an approximation algorithm that guarantees $A(I) \leq OPT(I) + n^{\alpha}$ (unless $P = NP$). ■