OPTIMAL DESIGN PROBLEMS FOR OPTICAL NETWORKS

MORDECHAI SHALOM
OPTIMAL DESIGN PROBLEMS FOR OPTICAL NETWORKS

RESEARCH THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

MORDECHAI SHALOM

SUBMITTED TO THE SENATE OF
THE TECHNION - ISRAEL INSTITUTE OF TECHNOLOGY
Adar, 5766 Haifa March 2006
THE RESEARCH THESIS WAS DONE UNDER THE SUPERVISION OF
PROF. SHMUEL ZAKS IN THE DEPARTMENT OF COMPUTER
SCIENCE

Acknowledgment

First and foremost I deeply thank Shmuel Zaks, for the support he gave me at all stages and all levels of my work. There is no doubt that this work was impossible without his help, support, and especially his tolerance. He acted as a friend (or better a brother), more than he acted as the supervisor of this work. I thank him for everything he did.

I began this work after many years spent in fields fairly distant from scientific research. At the beginning it seemed an impossible mission even to myself. To make scientific research after such a long break and a certain age, seemed to be feasible after a conversation held with Mr. Vitali Saporta on June 1999 and after a talk with Prof. Shmuel Zaks in the same month. I deeply thank them for their encouragement.

I deeply thank Tamar Eilam who gave me the first lecture on Optical Networks and introduced me to the field.

I deeply thank Ori Gerstel who, from the other end of the world, answered all the technical questions I asked him, although it was not his duty.

I deeply thank Michele Flammini for the fruitful discussions during, between and after my visits to the University of L’aquila. I must admit that it was a challenge to understand his ideas at the speed he produced them. I want to thank him and his wife Laurella for their hospitality. I thank his students Luca Moscardelli and Alfredo Navarra for their help and hospitality.

I deeply thank Jean-Claude Bermond, David Coudert and Stephane Perennes for their help and hospitality during my visit to INRIA, Sophia-Antipolis, for introducing me to the latest results on the ADM minimization problem, and for the fruitful discussions.

I thank to Reuven Cohen and Doron Tsur for introducing me to the problem of minimizing ADMs under uniform traffic in multi-hop SONET rings discussed in Chapter 4, and for the fruitful discussions.

I thank to Oded Shmueli, Roy Friedman and Seffi Naor for the pleasure it was to be a Teaching Assistant in their courses and the numerous things I learned from them.
I thank Zaks family, and especially Irit Zaks for the hospitality they showed during the countless visits I made to their home, for making me feel like a member of the family and making the work with Shmuel Zaks even more enjoyable.

THE GENEROUS FINANCIAL HELP OF THE TECHNION IS GRATEFULLY ACKNOWLEDGED.
# Contents

1 Introduction ................................................. 4  
   1.1 Background ......................................... 4  
   1.2 All-optical Networks ................................ 6  
   1.3 Approximation Algorithms and Approximation Ratio .... 10  
   1.4 Cost of Optical-Electronic (O-E-O) conversion ............ 10  
      1.4.1 ADM Minimization Problem ......................... 10  
      1.4.2 Traffic Grooming Problem ......................... 12  
   1.5 Formal Description ..................................... 12  
   1.6 Summary of Results and Published Work .................. 16  

2 Minimizing the number of ADMs in SONET Rings ............... 18  
   2.1 Introduction ........................................... 18  
      2.1.1 Background ....................................... 18  
      2.1.2 Previous Work ..................................... 19  
      2.1.3 Our Contribution .................................... 19  
   2.2 Problem Definition and Preliminary Results ............... 20  
      2.2.1 Problem Definition ................................ 20  
      2.2.2 Notation and Definitions ............................ 21  
      2.2.3 Restatement of The Problem ......................... 22  
   2.3 Algorithm PAF ........................................... 23  
      2.3.1 Algorithm AF ...................................... 23  
      2.3.2 Algorithm PAF ...................................... 24  
      2.3.3 Correctness and Complexity ......................... 26  
      2.3.4 Approximation Ratio ................................ 27  
      2.3.5 A Lower Bound ..................................... 33  
   2.4 Algorithms with Improved Preprocessing - IPA F and IEMZ ... 34  
      2.4.1 The Motivation .................................... 34  
      2.4.2 Algorithm $IPA_F$ .................................. 35
2.4.3 Analysis of $IPAF_k$ ........................... 35
2.4.4 Algorithm $IEMZ_k$ ........................... 36
2.4.5 Analysis of $IEMZ_k$ ........................... 37
2.5 Simulation Results, Conclusion and Possible Improvements  ... 38

3 Minimizing the number of ADMs in General Networks 40
3.1 Introduction ........................................ 40
  3.1.1 Background ..................................... 40
  3.1.2 Previous Work .................................. 41
  3.1.3 Our Contribution ............................... 41
3.2 Problem Definition and Preliminary Results .................. 42
  3.2.1 Problem Definition ............................... 42
  3.2.2 Preliminary Results ............................. 45
3.3 Algorithm MM ........................................ 48
  3.3.1 Correctness ..................................... 48
  3.3.2 Analysis ......................................... 49
  3.3.3 A lower bound .................................. 58
3.4 Algorithm $PMM(l)$ .................................. 60
  3.4.1 Correctness ..................................... 60
  3.4.2 Analysis ......................................... 60
3.5 Conclusion and Possible Improvements ....................... 81

4 Uniform All-to-all traffic in SONET Rings 83
4.1 Introduction ......................................... 83
  4.1.1 Background and Previous Work .................. 83
  4.1.2 Our Contribution ................................ 84
4.2 Problem Definition and Preliminary Results ............... 84
  4.2.1 Problem Definition ............................... 84
  4.2.2 Preliminary Results ............................. 88
4.3 The Demand Function ................................ 89
4.4 Nested Polygons ..................................... 91
  4.4.1 Definitions ....................................... 91
  4.4.2 Properties of Nested Polygons ................. 92
  4.4.3 Optimum Solution for $W=2$ ..................... 95
  4.4.4 A solution for any $W$ using Nested Polygons ... 96
  4.4.5 An improved upper bound ....................... 97
4.5 Generalization and Implications to Traffic Grooming .... 98
  4.5.1 The Traffic Grooming Problem .................... 99
5 Traffic Grooming in Ring Networks
   5.1 Introduction ............................................. 103
      5.1.1 Background ........................................... 103
      5.1.2 Previous Work ......................................... 103
      5.1.3 Our Contribution ....................................... 104
   5.2 Problem Definition ....................................... 104
   5.3 Algorithm GROOMBYSC(k) .................................. 106
   5.4 Analysis .................................................. 108
      5.4.1 Correctness ........................................... 108
      5.4.2 Running Time .......................................... 108
      5.4.3 Approximation Ratio ................................... 109
   5.5 Conclusion and Possible Improvements ...................... 112

6 Further Research .............................................. 114
   6.1 The Chord Version of the minimum ADM Problem in rings .. 114
   6.2 Splitable Requests - Multi-Hop Communication ............. 115
   6.3 Tradeoffs ............................................... 116
   6.4 Online Algorithms and Competitive Ratio .................. 116
   6.5 Randomized Algorithms ................................... 118

A ................................................................. 119
   A.1 Linear Program Used in Theorem 2.3.1 ...................... 120
   A.2 Linear Program Used in Theorem 2.4.1 ...................... 121
   A.3 Linear Program Used in Theorem 2.4.2 ...................... 122
List of Figures

2.1 Worst Case instance for Assign First ........................................ 25
2.2 Failure and Unmatch events ...................................................... 26
2.3 Proof of inequality 2.5 ............................................................... 31

3.1 Nodes are not repeated in a connected component ......................... 47
3.2 Case D is impossible ................................................................. 52
3.3 $d(r) = 1$, with different indices ................................................ 53
3.4 $d(r) = 2$ with different indices .................................................. 54
3.5 $d(r) = 2$ with even indices ....................................................... 55
3.6 $i$ cannot be even ................................................................. 55
3.7 A Cycle and a Spoon ............................................................... 56
3.8 A cycle of length 3 ............................................................... 57
3.9 A worst case input ............................................................... 58
3.10 A worst case $G_a$ and conflict graph ......................................... 59
3.11 A worst case solution ........................................................... 59
3.12 Blocking and blocked edges ....................................................... 64
3.13 Edges of $G_S$ with respect to cycles of $G_S$ ............................ 65
3.14 The nodes between two nodes of the solution ............................ 69
3.15 Matching by MODNDC-C of even cycles .................................... 73
3.16 Matching by using preprocessed dedicated even cycles ............... 74
3.17 Reducing $D_1$ .............................................................. 75
3.18 Matching by MODNDC-P ........................................................ 76
3.19 The graph $H$ .................................................................. 80

4.1 Routing of traffic from node 0 .................................................. 85
4.2 Demands routable on $e$ ........................................................ 90
4.3 Change in $f(e)$ ............................................................... 94
4.4 Change in a disjoint edge ............................................................ 95
5.1 The sets $V_{\lambda,j}$ and $\overline{S}_{\lambda,j} (k = 4)$ . . . . . . . . . . . . . . . . . . 111
Abstract

Third generation optical networks, also known as all-optical networks, enable us to transmit information in several wavelength ranges within a single fiber, independent of each other, and to route information at communication nodes without the need of examining its contents. Therefore, information travels from its source to its destination at the speed of light along a path called lightpath. We assume optical routers with no wavelength conversion, thus a lightpath uses the same wavelength on all the edges.

In single-hop communication a communication request from a node s to a node t should be assigned a path from s to t and a wavelength (color) in order to be transmitted, whereas in multi-hop communication it is assigned a sequence of paths (hops) each of which is assigned a wavelength independently. The problem of assigning paths and wavelengths to requests is called the WRA (Wavelength Routing and Assignment) problem. When the path assignment is already known, it is termed WLA (Wavelength Assignment) problem. Two paths sharing an edge should get distinct wavelengths, otherwise the information they carry interferes.

Most of the problems we discuss in our work are NP-Hard, therefore we are mainly interested in approximation algorithms.

The ADM minimization (MINADM) problem became of interest in the late 1990’s. Information enters a lightpath at its source and leaves it at its destination via equipment called ADMs. A lightpath uses an ADM at each endpoint. Obviously 2 |P| ADMs are sufficient for |P| paths. If two lightpaths with a common endpoint v get the same wavelength λ, then they may share an ADM operating at wavelength λ at node v. An ADM may be shared by at most two lightpaths. Therefore the minimum required number of ADMs is |P|, and the approximation ratio of any algorithm for this problem is at most 2.

The ring topology is important in communication networks because it provides protection for one failure with minimum edges. In our work we present a (10/7 + ε)-approximation algorithm for the MINADM problem in ring networks. We present also an improved analysis of an algorithm for general topology.

A communication request has a bandwidth which is an integer multiple of some basic unit of bandwidth. It is presented by an integer τi for each pk ∈ P. The bandwidth supplied by a wavelength is g times this basic unit, where g is an integer called the grooming factor. We assume τi ≤ g, because
otherwise it can not be accommodated in a single wavelength. A coloring is valid if for each edge \( e \) and wavelength \( \lambda \), the sum of the bandwidths of the paths using \( e \) and colored \( \lambda \) is at most \( g \). This family of problems is termed \textit{traffic grooming problems}. An ADM may service at most \( 2g \) paths, \( g \) at each side. Therefore the approximation ratio of any algorithm for this problem is at most \( 2g \).

In our work we present an efficient architecture in terms of number of ADMs used, for uniform all-to-all traffic in ring networks and multi-hop communication, using the bandwidth of the fiber optimally.

We also present an \( O(\log g) \)-approximation algorithm for ring networks, general traffic, single-hop communication and any grooming factor. The algorithm is usable in any topology, whereas our analysis holds in the ring topology.
# List of Terms

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDM</td>
<td>Wavelength Division Multiplexing</td>
</tr>
<tr>
<td>WLA</td>
<td>Wavelength Assignment</td>
</tr>
<tr>
<td>WRA</td>
<td>Wavelength Routing and Assignment</td>
</tr>
<tr>
<td>ADM</td>
<td>Add-Drop Multiplexer</td>
</tr>
<tr>
<td>LP</td>
<td>Linear Program</td>
</tr>
<tr>
<td>ILP</td>
<td>Integer Linear Program</td>
</tr>
<tr>
<td>$G = (V, E)$</td>
<td>Graph representing the network</td>
</tr>
<tr>
<td>$P$</td>
<td>The set of lightpaths</td>
</tr>
<tr>
<td>$g$</td>
<td>The grooming factor</td>
</tr>
<tr>
<td>$l(e)$</td>
<td>Load induced by $P$ on an edge $e \in E$, $=</td>
</tr>
<tr>
<td>$L_{\min}$</td>
<td>$\min_{e \in E} l(e)$</td>
</tr>
<tr>
<td>$L = L_{\max}$</td>
<td>$\max_{e \in E} l(e)$</td>
</tr>
<tr>
<td>$w$</td>
<td>The wavelength assignment $w : P \mapsto \mathbb{N}$</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>A wavelength</td>
</tr>
<tr>
<td>$W^w$ (or $W$)</td>
<td>Number of colors used by wavelength assignment $w$</td>
</tr>
<tr>
<td>$ADM_\lambda^w(v)$</td>
<td>ADMs operating at wavelength $\lambda$, at node $v$ under wavelength assignment $w$</td>
</tr>
<tr>
<td>$ADM_\lambda^w(v)$</td>
<td>ADMs used at node $v$ by wavelength assignment $w$</td>
</tr>
<tr>
<td>$ADM_\lambda^w$</td>
<td>ADMs operating at wavelength $\lambda$ under wavelength assignment $w$</td>
</tr>
<tr>
<td>$ADM^w$ (or $ADM$)</td>
<td>Number of ADMs used by wavelength assignment $w$</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Background

The demand for communication networks with ever-increasing capacity and service quality is the driving force behind ongoing fruitful research on physical network technology, protocols and resource management. Over the past decades users’ telecommunication needs have exploded. In the early 1970s, the highest-speed private lines were 9.6 kbps (kilobits per second). Today private lines at several gigabits per second are available. Within the public network in the USA, the 1.544 Mbps facilities of the 1970s have been replaced by 2.4 Gbps (gigabits per second) Synchronous Optical Network (SONET) facilities after the advent of Broadband ISDN (Broadband Integrated Services Digital Network, B-ISDN). Now, applications that require transfer rates of several Gbps are emerging. Such potential gigabit applications include supercomputer networking, remote visualization, medical imaging, collaborative group work, HDTV (high-definition television), VR (virtual reality) and telepresence. Fast and powerful networks are required on the LAN (local area network), MAN (metropolitan area network) and WAN (wide area network) levels.

The most promising technology for the implementation of gigabit networks is the so-called all-optical networks, i.e. networks in which data is transmitted on lightwaves through optical fiber and no optical-to-electrical conversions are required at intermediate nodes on the way from the sender to the receiver.

Changing characteristics of applications have also led to a shift from
packet switching to (virtual) circuit switching for the transmission of digital data. Traditionally, telephone networks were realized by circuit switching (a connection is established by reserving a channel on all links of a path between the communication endpoints), but digital data was transmitted using packet switching (a message is split into small packets that are forwarded from node to node until they reach the receiver). While packet switching is appropriate for file transfers or electronic messages, today more and more applications require connections with guaranteed available bandwidth and delay characteristics. For example, if moving image sequences (animations, movies) are transmitted in multimedia applications (e.g., multimedia teleconferencing), it is essential that all data arrive at the receiver in a timely fashion. Because of such applications, networks with guaranteed quality of service (QOS) are becoming increasingly important. In such networks, a connection request can specify its required bandwidth and other service parameters, and the network guarantees that the requested bandwidth and service quality will be available to the connection as long as it remains active. This is achieved by allocating resources along a path from sender to receiver; it can be viewed as establishing a virtual circuit.

At the WAN level, all-optical networks would be typically employed by big telecommunication providers. The network connects nodes located in different parts of the country or different countries. The connection requests correspond to the connectivity requirements predicted by the provider from information collected about telephone calls and data transmission over a period of time. The availability of optical amplifiers ensures that large distances do not cause a problem for all-optical networks.

In all-optical networks in particular and in networks with guaranteed quality of service in general, resource management is an important issue. Resources must be reserved for every established connection, and a bad resource management strategy may lead to reduced utilization and throughput of the network. In particular, call admission (deciding whether to accept or reject a connection request), call scheduling (establishing all connections in a set of connection requests and completing them in minimum time), wavelength routing and assignment (WRA) (assigning paths and wavelengths to connection requests in all-optical networks with wavelength-division multiplexing (WDM)) and wavelength assignment (WLA) (assigning wavelength to paths in all-optical networks with WDM) are problems that must be addressed. We study combinatorial optimization problems related to WRA and WLA. The focus is on simplified combinatorial models that capture the essen-
tial characteristics of the diverse problems encountered in practice. Most of
the problems encountered are known to be NP-complete. Therefore approxi-
imation algorithms with provable performance guarantees and lower bounds
for the approximability of the problems will be our main concern.

1.2 All-optical Networks

Three generations of physical-level technology for communication networks
can be defined. For networks of the first generation, up to the early 1980s,
fiber-optic technology was not available and copper was the main medium
for transmission of data. Networks of the second generation, still widely in
use today, are mainly characterized by upgrading individual copper links to
optical fiber. These networks take advantage of the higher bandwidth and
smaller bit-error rate of the optical fiber on individual links, but all switching
in the network nodes is still done electronically. The topologies and proto-
cols are the same as those used in the first generation. Finally, networks
of the third generation use optical technology in the nodes of the networks
too. The bottleneck caused by electronic switching is overcome by doing
all the switching optically (photonic switching). The key advantage of opti-
cal switching is that it avoids multiple optical-to-electrical conversions and
electronic switching operations at the intermediate nodes along a connection.

A single fiber-optic cable offers bandwidth of about 25,000 to 30,000
GHz. It can potentially carry information at the rate of several terabits
per second. No electronic device can process data at such speeds. In or-
der to utilize the potential of optical fiber, wavelength-division multiplexing
(WDM) is used. The bandwidth is partitioned into a number of channels at
different wavelength. A single channel supplies bandwidth in the range of
gigabits per second, and several signals can be transmitted through a fiber
link simultaneously on different channels. The typical number of channels
(wavelengths) available in WDM systems today ranges from two to several
hundred, with numbers near to 100 appearing practical. Tunable lasers or ar-
rays of fixed-wavelength lasers are used to generate the laser beams that are
to be transmitted on the optical channels. Add/drop multiplexers (ADM’s)
are employed at the network nodes to insert lightwaves into the fiber or to
extract them. Fixed-wavelength or tunable filters and receivers are used at
the receiving side of a transmission. The electronic equipment is not required
to operate faster than a single optical channel, thus, WDM allows existing
electronic equipment to fully use the enormous potential of optical fiber.

Several architectures for all-optical networks have been proposed. We assume an architecture with pairs of unidirectional fiber links or with single bidirectional links between adjacent nodes of the network. No distinction between access nodes and routing nodes is made. All nodes are assumed to have both capabilities.

An architecture for all-optical WDM networks based on the broadcast-and-select paradigm is proposed and studied extensively [BH94]. For example, the passive optical star topology has received a lot of attention. A number of nodes are connected to a central hub. All the transmissions from the various nodes are combined in the central hub and mixed optical information is broadcast to all nodes. Besides the star topology, linear bus and tree structures have been proposed. Broadcast-and-select networks can be classified into single-hop and multi-hop systems.

A reconfigurable wavelength-selective switch is capable of splitting the signals on the incoming links according to their wavelengths and switching them onto arbitrary outgoing links. While setting up the configuration of such a switch is time consuming (in the order of magnitude of tens of milliseconds), it can switch incoming signals onto outgoing links virtually without any delay. The routing of the signal depends on its wavelength (wavelength routing), i.e. no inspection of packet headers etc. is required. The latency of an all-optical network is therefore limited only by the propagation delay of light in fiber. Light travels in fiber at a speed of about $2 \times 10^8$ m/s, hence a distance of say 5,000 meters can be travelled in 25 microseconds. All-optical networks with reconfigurable wavelength-selective switches have been proposed and studied in numerous papers, for example [CGK92] and [RS95].

Most optical switches do not offer wavelength conversion, a signal that enters the switch on one wavelength must leave the switch on the same wavelength. All-optical networks with reconfigurable wavelength-selective switches, but without wavelength conversion are called wavelength-selective (WS) networks. In WS networks, a connection must use the same wavelength on the whole path from the transmitter to the receiver. Due to interference, no two signals may be transmitted through a fiber link on the same wavelength (on the same channel). Hence, a wavelength must be preserved for a connection on all links on a path from its transmitter to its receiver, the path is then called a lightpath.

The application of all-optical networks for distributed computing systems
seems very promising as well. Lightpaths can be used to establish a virtual topology on top of the underlying physical topology of the network. For example, a regular topology like hypercube or grid may be imposed on the nodes interconnected by an irregular all-optical network. It suffices to establish a lightpath for each connection of the virtual topology. Different virtual topologies can be achieved by reconfiguring the switches of the network. Once a certain virtual topology is established, the communication between the nodes can take place as if the links of the virtual topology where physically present, no delay beyond the propagation delay of light in fiber is introduced by the fact that a link of the virtual topology is realized by a lightpath in the physical topology.

Switches with wavelength conversion capabilities are a field of ongoing research. This technology is considered relatively immature and expensive. However, it is conceivable that technological progress will make these devices practical in the future. A switch with full wavelength conversion capabilities can switch any incoming signal onto any outgoing link and at the same time change its wavelength into any other wavelength. All-optical networks with full wavelength conversion switches are called wavelength-interchanging (WI) networks. In a WI network, a connection can use different wavelengths on different links of its path.

The complexity of building switches with full wavelength conversion capabilities is high, therefore switches with limited wavelength conversion capabilities are also considered. In particular, switches that can convert every wavelength to a subset of the wavelengths or even to adjacent wavelengths may be more practical than switches with full wavelength conversion capabilities.

In all-optical networks, the number of channels (wavelengths) is a scarce resource. The cost and complexity of the switches and add/drop multiplexers (ADM’s) grow substantially with increasing number of wavelengths.

A set of connections in a WS network can be established if each connection is assigned a path from its transmitter to its receiver, plus a wavelength such that paths of connections sharing a link are assigned different wavelengths. This gives rise to two kind of optimization problems. First, it is desirable to establish a given set of connection requests with a minimum number of wavelengths. This problem is relevant, for example when the provider designs the network and decides which add/drop multiplexing device should be employed. We view wavelengths as colors and model this wavelength assignment (WLA) problem as the path coloring problem. The second problem
arises when the capacity of an existing network is not sufficient for establishing all the request in a given set of connection requests simultaneously. If the network supports a certain number of wavelengths, the goal is to establish as many of the connection requests as possible simultaneously, while rejecting (or deferring) the remaining requests. This problem is modelled by the \textit{MAXPC} problem.

In a WI network, a set of connections can be established if they can be assigned paths such that no link is used by more connections than the number of available wavelengths. For WI networks the problems above are referred as the \textit{path packing} and \textit{MAXPP}, respectively.

These problems and their variants are studied extensively in the literature for various topologies: general graphs ([RS95], [AA98], [KK99], [BGP96]), rings ([RS95], [NZ97], [GK97], [GSKR97], [GRS98], [PSSV98]) and trees ([KS97], [KP96], [KPEJ97], [ACKP97], [GHP97], [Gar98], [ACKP98], [AAF96], [BL97]) are the most widely studied topologies. Trees of rings [BP99], hypercubes [MS99] and chordal rings [NOS99] may be stated among other topologies of interest.

While WDM appears the most promising alternative for employing all-optical networks, there is a competing \textit{space-division multiplexing} (SDM) technology. Here, several parallel single-wavelength links are installed between adjacent nodes. All the data is transmitted on the same wavelength. The advantage is that there are no wavelength assignment problems for such networks. A disadvantage is the cost for the additional fiber cables and their installation, especially in MAN and WAN distances.

It is also conceivable that SDM and WDM are combined in one all-optical network. The effect of multiplying each fiber by \( k \) fibers is analyzed in [MS00], the general provisioning problem is studied in [AA98]. The effect of multiplying the bandwidth of all the channels by a constant to the on-line version of the problem is studied in [BL97].

Under some circumstances, communication requests may be served by concatenating several lightpaths. In this case the request is converted to electronic format and then back to optical at the nodes between each pair of consecutive lightpaths. Each one of the lightpaths serving a request is a hop, and this kind of service is a \textit{multi-hop} service, as opposed to \textit{single-hop} service in which this conversion is not allowed except at the endpoints of the requests.
1.3 Approximation Algorithms and Approximation Ratio

A polynomial-time algorithm that always produces an optimal solution for a given optimization problem is called an exact algorithm. The existence of an exact algorithm for an NP-complete optimization problem would imply $P = NP$. Therefore, as mentioned earlier, one is interested in polynomial-time approximation algorithms for such problems.

An (deterministic) approximation algorithm $ALG$ for an optimization problem $\Pi$ is a deterministic algorithm whose running time is polynomial in the size of the input and that always computes a feasible solution. Denote by $ALG(I)$ the value of the output of $ALG$ on input $I$. Denote by $OPT(I)$ the value of the optimal solution to $I$. When $I$ is understood from the contest we will denote them shortly as $ALG$ and $OPT$ respectively. When $\Pi$ is a maximization problem, $ALG$ is a $\rho$-approximation algorithm for $\Pi$ if for every input $I$ the inequality

$$ALG(I)/OPT(I) \geq \rho$$

holds. Where $\Pi$ is a minimization problem, $ALG$ is a $\rho$-approximation if for every input $I$ it is true that

$$ALG(I)/OPT(I) \leq \rho.$$

Note that if $\Pi$ is a maximization problem then $\rho \leq 1$ and if $\Pi$ is a minimization problem $\rho \geq 1$ by definition. If $\rho = 1$ then $ALG$ is an exact algorithm.

The approximation ratio (or the performance guarantee) $\rho_{ALG}$ of $ALG$ is the supremum (resp. infremum) of the values $\rho$ such that $ALG$ is a $\rho$-approximation algorithm for the maximization (resp. minimization) problem $\Pi$.

1.4 Cost of Optical-Electronic (O-E-O) conversion

1.4.1 ADM Minimization Problem

As mentioned above, data is converted from electrical form to optical form at one endpoint of the lightpath and converted back to electronic form at
the other endpoint by electronic terminating equipment in both ends. This is often called O/E conversion. The electronic equipment is termed Add-Drop Multiplexers (ADMs).

When the various parameters comprising the switching mechanism in these rings become clearer, the focus of studies shifted, and today a large portion of the studies concentrates on the total hardware cost. The key point here is that each path uses two ADM's, one at each end point.

If two adjacent lightpaths are assigned the same wavelength, then they can use the same ADM. An ADM may be shared by at least two lightpaths. The total cost considered is the total number of ADM. The question of minimization of this cost is the main focus of our work. This family of problems are variants of the WLA and WRA problems in the sense that the inputs and output remain the same, but the objective function is different.

In these coloring (or routing and coloring) problems the goal is to minimize the number of add/drop multiplexers (ADMs) which will be used by the network.

Most of the works on this problem are for ring networks. This stems from the fact that SONET rings are the networks in which the WDM technology is most widely implemented. [GLS98, GLS99, LIWF00, CFW02] investigate the WLA problem in which each traffic stream has a predetermined routing. This is called also the arc version of the problem. The problem is proved to be in NP-complete [LIWF00]. Most of these heuristics which have approximation ratio at least \( \frac{3}{2} \leq r \leq 1.53 \ldots \) as 

The Preprocessed Iterative Matching algorithm proposed in [CW02b], some of these heuristics are proved to have approximation ratio at least \( \frac{3}{2} \leq r \leq 1.53 \ldots \) as well as part of the input and they are be determined by the solution approach as well.

In these works 1.6 and 1.5 + \( \epsilon \) approximation algorithms are presented, respectively.
1.4.2 Traffic Grooming Problem

Service providers serve users’ communication requests which are only a fraction of the capacity of one wavelength. In this case several of these requests can be joined together to fit into a single wavelength, is called traffic grooming. Several requests are "groomed" to one wavelength. For instance a lightpath of an OC-48 ST0NET ring, can support up to 16 OC-3 simultaneous communication requests. Generally communication request have a granularity which is a fraction $g$ of one wavelength’s capacity. This granularity is often termed as the grooming factor. This adds one more dimension to the above WLA and WRA problems. In graph-theoretic terms, in a valid coloring paths should be colored so that at most $g$ paths sharing a common edge may be colored with the same color. An ADM has two endpoints, each connected to one adjacent edge of the node its resides on. Given a valid coloring, an ADM can serve at most $2g$ paths colored with the same color. More specifically, for each one of its two endpoints, at most $g$ paths colored with its color and terminating at this endpoint. Obviously, the ADM minimization problems are special cases of the Traffic Grooming Problems and are obtained by setting $g = 1$.

The notion of traffic grooming ($g > 1$) was introduced in [GRS98] for the ring topology. The problem was shown to be NP-complete in [CM00] for ring networks and a general $g$. NP-completeness results for ring and path networks are shown in [Ung05] for any fixed value of $g$.

The uniform all-to-all traffic case, is widely studied. In this problem there is a demand with the same value between each pair of nodes. The problem is studied in [CM00, BC03] for various values of $g$. An optimal construction for the uniform all-to-all problem, for the case $g = 2$ in a path network was given in [BBC05].

1.5 Formal Description

In this section we give a formal definition of terms and general settings. Then we mention parameters and possible settings of each of them. Almost every combination of these settings is possible. Although, our work concentrates in a small fraction of this possible settings, we include this description for completeness.

A network is modelled by a graph (or a digraph) $G = (V, E)$. The directed
case will be explained in the sequel. The nodes $v \in V$ represent the nodes of the network on which the optical and electronic hardware reside, whereas the edges represent the optical fibers connecting them. Assuming that the bandwidth of one wavelength is our unit of traffic, a traffic $\mathcal{T}$ on this network is represented by an integer $|V| \times |V|$ matrix and an integer $g$. The traffic demand from node $u$ to node $v$ is $T_{u,v}/g$, and $g$ is the grooming factor. This is justified by the reasonable assumption that traffic demands are multiples of some basic traffic unit which is equivalent to $\frac{1}{g}$ of one wavelength bandwidth. An alternative representation of this traffic is a list of triples $(s_i, d_i, \tau_i) \in V \times V \times \mathbb{N}$ each of which corresponds to a traffic request, and an integer $g$. In the following discussion we assume the second representation. We further assume that in each triple, $\tau_i \leq g$, otherwise we split it into several triples. The routes for each traffic request are modelled by simple paths $p_i$ on $G$. The wavelengths are modelled by integers $\lambda \in \mathbb{N}$ and wavelength assignment is a function $w$ which maps a path $p_i$ to a wavelength $\lambda$. A wavelength assignment is $g$-feasible if for each edge $e \in E$, and every wavelength $\lambda$ the sum of the demands of the paths using $e$ and colored (i.e. mapped to) $\lambda$ is at most $g$. Formally, if $\forall e \in E, \lambda \in \mathbb{N}, \sum_{e \in p_i \land w[p_i] = \lambda} \tau_i \leq g$.

Variations of the problem can be obtained by restricting the input $G, \mathcal{T}, g$.

a. The topology

Some work deal with general topology, whereas mostly $G$ is restricted to rings. The tree, line, hypercube topologies are also studied in the literature.

b. The traffic

The most widely studied traffic types are general traffic and uniform all-to-all traffic in which the traffic matrix is uniform. Other type of traffics that are studied are one-to-all traffic (in which the matrix has all ones in one row and the rest is zero), permutation traffic (the matrix is a permutation matrix) and $t$-restricted (each row and column of the matrix contains at most $t$ non zero values). Note that permutation traffic is 1-restricted traffic.

c. The grooming factor

The problem of minimizing the number of ADMs in its basic case corresponds to grooming factor $g = 1$ and the traffic grooming problem corresponds to the general case ($g > 1$).
Other variations of the problem are obtained by altering the underlying assumptions.

a. WRA vs. WLA.

Generally the problem is a WRA problem. The input is a graph $G$ and a traffic $T$ with a grooming factor $g$; the output is a multi-set of paths $P$ and a $g$-feasible wavelength assignment $w$. In the literature the problem is often split into two subproblems:

- Routing: The input is a graph $G$ and a traffic $T$ with a grooming factor $g$; the output is a routing, which is a multi-set of paths $P$.
- WLA: The input is a graph $G$ and a multi-set of paths $P = \{P_1, P_2, \ldots\}$ and demands $\tau_i$ corresponding to these paths; the output is a $g$-feasible wavelength assignment $w$.

In graph theoretic terms both WLA and WRA problems are optimization problems of coloring, or routing and coloring, whose cost measures are taken from their applications for optical networks. In ring networks the WRA problem is also termed the chord version of the problem, and the WLA problem is termed the arc version of the problem.

b. Objective function.

The number of colors used by a wavelength $w$ assignment is $W^w = |\text{Range}(w)|$. In order to define the number $\text{ADM}^w$ of ADMs used by a wavelength assignment $w$ we need to define the following entities: $\text{ADM}^w_\lambda(v)$ is the number of ADMs used by wavelength assignment $w$ at node $v$, operating at wavelength $\lambda$. $\text{ADM}^w(v) = \sum_{\lambda=1}^{\infty} \text{ADM}^w_\lambda(v)$ is the number of ADMs used by wavelength assignment $w$ at node $v$. Finally $\text{ADM}^w = \sum_{v \in V} \text{ADM}^w(v)$.

For each of the scenarios that will be studied (e.g., a directed ring network, with grooming factor $g = 2$) the following cost functions will be considered:

- Minimize the number $W^w$ of wavelengths used.
- Minimize the number $\text{ADM}^w$ of ADMs.
- Minimize the number $\text{ADM}^w$ of ADMs under special constraints on $W^w$. In this setting we may consider the case where a bound
on the number of colors is given, and the goal is to find the best solution in terms of ADMs used, satisfying this bound. Another possible constraint on $W_w$ is that it should attain the minimum possible value.

**c. Mode of communication.**

Two modes of communication are possible. First, we describe them under the WRA settings:

- **Single-hop:** Each traffic request $(s_i, d_i, \tau_i)$ is routed through a single simple path $p_i$.
- **Multi-hop:** Each traffic request $(s_i, d_i, \tau_i)$ may be routed through a sequence of simple paths $p_{i,1}, ..., p_{i,k}$ where the starting node of $p_{i,j+1}$ is the ending node of $p_{i,j}$, the starting node of $p_{i,1}$ is $s_i$ and the ending node of $p_{i,k}$ is $d_i$. In this case the concatenation of the paths $p_{i,j}$ is a path from $s_i$ to $d_i$ and we say that this traffic request is routed in $k$ hops. This mode of communication is also called *splittable* (vs. *unsplittable*) requests.

For the WLA family of problems multi-hop is the case that each given path $p_i \in P$ may be split into sub-paths $p_{i,1}, ..., p_{i,k}$ and the assigned wavelengths independently. This is termed *splittable* (vs. *unsplittable*) paths.

**d. Direction.**

- **Directed graph, Directed traffic.**
  The information flow in a single fiber is in one direction. Therefore the natural representation of the network is a directed graph. On the other hand the traffic requests are also directed from the source to the destination.

- **Undirected graph, Directed traffic.**
  Usually, optical fibers are installed in pairs and the laser devices at the endpoints are installed so that the information in the fibers flows in opposite directions. In this case we can model the network by an undirected graph, but it represents a symmetric directed graph.

- **Undirected graph, Undirected traffic.**
Under most circumstances the traffic matrix is symmetric. Namely, for any pair $u, v$ of nodes the traffic request from $u$ to $v$ is equal to the traffic request from $v$ to $u$. However, it is still possible to route each of them on a different path. Usually this is not done for protection reasons. Most applications are duplex; if the communication path from $u$ to $v$ fails, the communication path from $v$ to $u$ becomes obsolete. Therefore it makes sense to route them on the same physical path on the opposite directions. Furthermore it is also customary to demand that they are colored with the same wavelength, for protection against ADM failures. Under this requirements which are rather seldom, the problem becomes completely undirected.

e. Bifurcation.

A traffic request either can be routed in one path and assigned a wavelength, or split into sub-requests each of which is routed and/or wavelength assigned independently. In the first case we say that bifurcation is not allowed. Even when bifurcation is allowed, we are allowed to split requests into integral values (i.e. multiples of $1/g$) only.

f. On-Line vs. Off-line

In the off-line problem, the paths (or requests) of the input are given upfront, whereas in the online problem they arrive one at a time.

In the online version, let $\sigma$ be the sequence of paths (or requests). The algorithm should make its decision on input $\sigma_i$ based on the subsequence $\sigma_1, \ldots, \sigma_i$ only. When $\sigma_i$ can be also a deletion of a previous request, this is also called the dynamic version of the problem, otherwise it is called the incremental problem.

1.6 Summary of Results and Published Work

In Chapter 2 we investigate the relationship between the approximability of the maximum disjoint paths problem and the approximability of the arc version of the minimum ADM problem in undirected rings. Using known approximation algorithms for the maximum disjoint paths problem, we improve the performance of known algorithms ([GLS98, LLWF00, EMZ02, CW02b]) and present a $10/7 + \epsilon$-approximation algorithm for the problem.
This work is published in [SZ04]. Recently, a 10/7 approximation algorithm using similar techniques but a completely different analysis was presented in [EL04].

In Chapter 3 we give an improved analysis of the algorithms in [CFW02] for the ADM minimization problem in general networks. We first discuss the algorithm without preprocessing and suggest a new proof for the known upper bound of $OPT + 0.6N$ for its performance, and we prove that this bound is tight. We then discuss the algorithm with preprocessing and prove a performance bound of $OPT + \frac{1}{2}(1 + \epsilon)|P|$, where $OPT$ is the cost of an optimal solution, $P$ is the set of lightpaths and $\frac{1}{2l+3} \leq \epsilon \leq \frac{1}{4(l+2)}$, for any given odd $l$, thus further closing the gap between the upper and lower bounds. The results shed more light on the structure of this basic algorithm. In addition, in our analysis we suggest a novel technique - including a new combinatorial lemma - to deal with this problem. This work is submitted for publication ([FSZ05]).

In Chapter 4 we consider the uniform all-to-all traffic pattern in ring networks and multi-hop communication. We analyze the problem of finding an architecture with as few ADMs as possible under the constraint that the bandwidth of the fiber is used maximally. This work appeared in [SZ05].

In Chapter 5 we present an algorithm for the Traffic Grooming problem in general networks, for general traffic pattern and any value of $g$. We show that this algorithm is a $2 \ln g + o(\ln g)$-approximation in some topologies including the ring topology. This work appears in [FMSZ05].
Chapter 2

Minimizing the number of ADMs in SONET Rings

2.1 Introduction

2.1.1 Background

Recent studies (e.g., [GLS98], [LLWF00]) argue that an important cost measure is the number of ADMs used by the network. Moreover, these studies concentrate on a ring topology for various reasons. One of the commonly stated reasons is that higher level networks which make use of the WDM network cannot necessarily support arbitrary topologies. The most widely deployed network above the WDM layer is the SONET/SDH self-healing rings. These networks have to be configured in rings for protection purposes.

We concentrate on the problem of minimizing the additional overhead resulting from the need of these lightpaths to be configured as rings. This can be split into two problems:

- Assign a route to a lightpath; namely, choose one of two possible directions on the ring such that the maximum number of lightpaths intersecting on an edge is minimal. This is called the ring loading problem. In [WW98] an optimal solution for the problem in directed rings is given. As for undirected rings, a polynomial time approximation scheme is given in [Kha97].

- Given the routing above, assign wavelengths to the paths such that the number of ADMs used by the system is minimized. In this chapter we
focus on this problem.

2.1.2 Previous Work

The problem is studied for general topology in [CFW02] and [EMZ02], although the motivation in the latter is slightly different. A number of previous works [GLS98, GLS99, LLWF00, CW02b] studied the minimum ADM problem in SONET rings in which each traffic stream has a predetermined routing. This is called also the arc version of the problem. The problem is proved to be NP-complete in [LLWF00]. Several heuristics are proposed in [GLS98, LLWF00, WCLF00], most of which have approximation ratio at least 3/2. Some of the heuristics are proved to have approximation ratio at least $\frac{3+\epsilon}{1+\epsilon} = 1.537...$. Note that an approximation ratio of 2 is trivial: optimum is at least the number of lightpaths and any solution will use at most twice the number of lightpaths (one at each endpoint).

The Preprocessed Iterative Matching heuristic proposed in [CW02b] solves the arc version of the minimum ADM problem and is shown to have an approximation ratio of 3/2.

A 10/7 approximation algorithm is presented recently in [EL04].

2.1.3 Our Contribution

Our main result is a $10/7 + \epsilon$ approximation algorithm for the minimum ADM problem. We start by giving a formal definition of the problem. We show that the approximation ratio of the Assign First algorithm presented in [GLS98] is at least 5/3, present an algorithm which is a modified version of this algorithm, and prove that its approximation ratio is between 3/2 and 11/7.

We then investigate the relationship between the approximability of the maximum disjoint paths problem and the approximability of the arc version of the minimum ADM problem. Using good approximation algorithms for this problem we manage to improve our first algorithm and obtain a second algorithm with approximation ratio less than 1.48. Finally by using the same technique on the algorithm presented in [EMZ02] we obtain an algorithm with approximation ratio at most $10/7 + \epsilon$. 
2.2 Problem Definition and Preliminary Results

2.2.1 Problem Definition

Given a WDM ring network $G = (V, E)$ such that $V = \{0, 1, ..., N - 1\}$ comprising optical nodes and a set of full-duplex lightpaths $P = \{p_1, p_2, \ldots\}$ such that for all $j$, $p_j = (s_j, e_j)$ and $s_j, e_j \in V$, a wavelength assignment $w$ assigns a wavelength to each lightpath $p_i$. Formally $w : P \mapsto \mathbb{N}$. The forward part of the duplex lightpath $(s_i, e_i)$ traverses from $s_i$ to $e_i$ and the reverse part traverses from $e_i$ to $s_i$. Call $s_i$ the starting node and $e_i$ the ending node. $s(p_i) \overset{def}{=} s_i$ and $e(p_i) \overset{def}{=} e_i$.

Without loss of generality we assume that each lightpath $p_i$ is routed clockwise on the ring from $s_i$ to $e_i$. Under this assumption the following definitions are valid.

**Definition 2.2.1** $p, p' \in P$ are conflicting or overlapping if $p$ and $p'$ have an edge in common. This is denoted as $p \succ p'$.

**Definition 2.2.2** $\text{len}(p)$ is the length of the lightpath $p$, namely $(e(p) - s(p)) \mod N$.

**Definition 2.2.3** For any edge $e \in E$, its load $l(e)$ is the number of lightpaths containing it, and $L_{\min} \overset{def}{=} \min_{e \in E} l(e)$.

**Definition 2.2.4** A proper coloring (or wavelength assignment) of $P$ is a function $w : P \mapsto \mathbb{N}$, such that $w(p) \neq w(p')$ whenever $p \succ p'$.

We assume that $P$ is given upfront, in other words we study the static (i.e. off-line) WLA problem. This assumption is reasonable for example in the case of very high-speed pipes in the telecom environment.

Electrical TDM line-terminals terminate the lightpaths. We assume this nodes are SONET/SDH add/drop multiplexers (ADMs). Each lightpath $p$ uses two ADMs, one at $s(p)$ and another at $e(p)$. Although in $s(p)$ (resp. $e(p)$) only the downstream (resp. upstream) ADM function is needed, full ADMs will be installed on both nodes in order to complete the protection path around the ring. The full configuration would result in a number of SONET rings all circumventing a single optical ring. It follows that if two
adjacent lightpaths are assigned the same wavelength, then they are used by the same SONET ring and the ADM in the common node can be shared by them. This would save the cost of one ADM. With this in mind, we define our goal, as follows:

Given a wavelength assignment $w$, for each node $i$ of the ring, and for every wavelength $\lambda \in \mathbb{N}$, $\text{ADM}_\lambda^w(i)$ is an indicator variable indicating whether or not there is an ADM operating at wavelength $\lambda$ at node $i$. In other words, it assumes the value 1, if there is a lightpath colored $\lambda$ starting or ending at $i$, and 0 otherwise. Formally:

$$\forall i \in V, \lambda \in \mathbb{N}, \text{ADM}_\lambda^w(i) \overset{\text{def}}{=} |\{p \in P | w(p) = \lambda \land (s(p) = i \lor e(p) = i)\}|$$

$\text{ADM}_\lambda^w(i)$ is the number of ADMs used at node $i$, formally: $\text{ADM}_\lambda^w(i) = \sum_{\lambda=1}^{\infty} \text{ADM}_\lambda^w(i)$.

The goal is to minimize the total cost function:

$$\text{ADM}^w = \sum_{i=1}^{N} \text{ADM}_\lambda^w(i).$$

A more convenient statement of the problem is discussed in Section 2.2.3.

### 2.2.2 Notation and Definitions

Given a wavelength assignment, we make the following definitions.

**Definition 2.2.5** A chain $c$ is a maximal sequence of distinct consecutive lightpaths $(p_1, p_2, ..., p_k)$ assigned the same wavelength by $w$, s.t. $\forall i > 1, s(p_i) = e(p_{i-1})$ and $e(p_k) \neq s(p_1)$. $s(p_1)$ (resp. $e(p_k)$) will be called the start of the chain and will be denoted as $s(c)$ (resp. $e(c)$). $\text{len}(c) \overset{\text{def}}{=} \sum_{i=1}^{k} \text{len}(p_i)$, or in other words $(e(c) - s(c)) \mod N$.

**Definition 2.2.6** A cycle $c$ is a sequence of distinct consecutive lightpaths $(p_1, p_2, ..., p_k)$ assigned the same wavelength by $w$, s.t. $\forall i > 1, s(p_i) = e(p_{i-1})$ and $e(p_k) = s(p_1)$. In this case $\text{len}(c) = N$.

As the elements of the chains and cycles are distinct, we will refer to these sequences sets too.
Definition 2.2.7 The wavelength of a chain/cycle $c$ is the unique wavelength assigned to its lightpaths: $w(c) \overset{def}{=} w(p_1)$.

Definition 2.2.8 The (unique) chain/cycle that contains a lightpath $p$ is denoted by $c(p)$.

For our algorithms we will use the following two definitions, following [GLS98] and [Eil00], respectively.

Definition 2.2.9 $\forall i \in V$:
\[ \tau_i \overset{def}{=} \{p \in P | s(p) = i\} \]
\[ \sigma_i \overset{def}{=} \{p \in P | e(p) = i\} \]
\[ X_i \overset{def}{=} \{p \in P | i \text{ is a node of } p\} \setminus (\tau_i \cup \sigma_i) \]
\[ Y_i \overset{def}{=} X_i \cup \tau_i \]

Definition 2.2.10 The node graph of a node $i \in V$ is the bipartite graph $G_i = (\tau_i, \sigma_i, E_i)$ where $(p, p') \in E_i \subseteq \tau_i \times \sigma_i$ whenever $p \neq p'$.

2.2.3 Restatement of The Problem

Clearly $ADM^w = \sum_{\lambda=1}^{\infty} \sum_{i=0}^{N-1} ADM^w_{\lambda}(i)$. For any wavelength $\lambda$, consider all the chains and cycles $c$ such that $w(c) = \lambda$, consider also a node $i \in V$. If there is no lightpath $p$ such that $w(p) = \lambda$ and $i \in \{s(p), e(p)\}$ then $ADM^w_{\lambda}(i) = 0$, otherwise $ADM^w_{\lambda}(i) = 1$. Therefore $ADM^w_{\lambda}$ is the number of nodes of the chains (and cycles) colored $\lambda$. The number of these nodes is exactly the number of lightpaths in all those chains/cycles, plus the number of the chains. This is because a cycle $c$ has $|c|$ nodes and a chain $c'$ has $|c'| + 1$ nodes. Summing over all $\lambda$ we conclude that $ADM^w = |P| + \text{The number of chains}$.

Since our goal is to minimize the number of ADMs and not the number of wavelengths, we slightly change the statement of the problem. In accordance with the above discussion we are not concerned with the wavelength assignment itself but only on the chains and cycles induced by it. Thus an optimal solution of the minimum ADM problem is a partitioning of $P$ into chains and cycles such that the number of chains is minimum.
2.3 Algorithm PAF

2.3.1 Algorithm AF

In this section we present algorithm PAF (Preprocessed Assign First) which is obtained by modifying the Assign First algorithm in [GLS98]. We use the notations and definitions of the previous section.

We briefly describe the Assign First algorithm:

- The nodes of the ring are renumbered from 0 to \( N - 1 \) such that 0 is a node minimizing some objective function (which is not relevant for our purposes).

- All the lightpaths in \( Y_0 \) are colored with distinct colors.

- The nodes are scanned from 1 to \( N - 1 \). At each node \( i \) the lightpaths in \( \tau_i \) are colored. This coloring is done in the following manner: The colors of the lightpaths in \( \sigma_i \) are preferred colors. The preferred colors are used first, if they are exhausted, other colors are used from lowest numbered first. If a color is not valid for a lightpath, the next color is tried.

Now we restate Assign First in our terminology:

- Remumber the nodes of the ring from 0 to \( N - 1 \).

- Designate each lightpath in \( Y_0 \) as a chain by itself.

- Scan the nodes from 1 to \( N - 1 \). At each node \( i \) first try to expand the chains \( c \) ending at \( i \), then form new chains of the lightpaths in \( \tau_i \) which have not yet joined a chain.

Claim 2.3.1 For every \( \epsilon > 0 \) there are infinitely many instances for which \( PAF/OPT > 5/3 + \epsilon \).

Proof: Consider the following infinite family of instances. For any given integer \( k > 1 \) the ring of the instance contains \( N = 2k + 1 \) nodes numbered \(-k + 1, -k + 2, \ldots, 0, 1, \ldots, k + 1\) and the \( 3k \) lightpaths

\[
(0, 1), (-1, 1), \ldots, (-k + 1, 1),
(1, 2), (1, 3), \ldots, (1, k + 1),
(2, -k + 1), (3, -k + 2), \ldots, (k + 1, 0)).
\]
A sample instance for $k = 4$ is depicted in Figure 2.1. As it can be easily seen in the figure, OPT consists of the $k$ cycles

$$(0, 1), (1, k + 1), (k + 1, 0)$$

$$(-1, 1), (1, k), (k, -1)$$

$$(-2, 1), (1, k - 1), (k - 1, -2)$$

... 

$$(-k + 1, 1), (1, 2), (2, -k + 1).$$

On the other hand one possible solution for ALG consists of the following $2k - 1$ chains:

$$(0, 1), (1, 2), (2, -k + 1)$$

$$(-1, 1), (1, k + 1)$$

$$(k + 1, 0)$$

$$(-2, 1), (1, k)$$

$$(k, -1)$$

... 

$$(-k + 1, 1), (1, 3)$$

$$(3, -k + 2)$$

We have $OPT = |P| = 3k, AF = 3k + 2k - 1 = 5k - 1$, therefore $AF/OPT = \frac{5k - 1}{3k} = 5/3 - \frac{1}{3k}$. We conclude that for any $\epsilon > 0$, $AF/OPT > 5/3 - \epsilon$.

\[ \square \]

**Corollary 2.3.1**

$\rho_{AF} \geq 5/3 = 1.66...$

### 2.3.2 Algorithm PAF

PAF has two major differences from Assign First:

- It has a preprocessing phase.
- The attempts to expand the chains are done by trying the maximum matching of the node graph. Note that this does not change the performance of the algorithm on the worst case input just described.
Figure 2.1: Worst Case instance for Assign First

PAF \((N, P)\) {

Preprocessing:
A) Remove a maximal set of cycles of two paths from \(P\).
B) Remove a maximal set of cycles from \(P\).

Processing:
For each path \(p \in P\) do \(c(p) = (p)\)
For each node \(i\) from 1 clockwise to 0 do{
    Find a maximum matching \(MM_i\) of the node graph \(G_i\).
    \(\tau'_i = \) The unmatched nodes of \(\tau_i\).
    \(\sigma'_i = \) The unmatched nodes of \(\sigma_i\).
    \(G'_i = \) The complete bipartite graph \((\tau'_i, \sigma'_i, \tau'_i \times \sigma'_i)\).
    Find a maximum matching \(MM'_i\) of \(G'_i\).
    \(\tau''_i = \) The unmatched nodes of \(\tau'_i\).
    \(\sigma''_i = \) The unmatched nodes of \(\sigma'_i\).
    For each edge \((a, b) \in MM_i\) {
        if \(\text{len}(c(a)) + \text{len}(c(b)) \leq n\)
            UNION \((c(a), c(b))\)
else

failure

\}

For each edge \((a, b) \in \mathcal{M}\mathcal{M}'\) unmatched

For each \(b\) in \(\tau''\) start

For each \(b\) in \(\sigma''\) end

The words failure, unmatched, start, end written in bold in the code are events which are generated for the sake of the analysis, otherwise they do nothing. Figure 2.2 describes the four cases that may cause such an extension to fail. A solid line depicts a chain consisting of one lightpath and dashed line depicts a chain with at least two lightpaths.

### 2.3.3 Correctness and Complexity

Once a lightpath \(p\) is added to a chain it is not added to another one. This could happen only at node \(s(p)\). This can not happen twice, because it is done by an edge of a matching. Therefore the output is a partitioning of \(P\).

During the execution of the algorithm a chain or cycle’s length can not exceed \(N\), because this is checked before every potential extension of a chain. Moreover a lightpath \(p\) is added to a chain \(c\) only at node \(e(c) = s(p)\) in a manner consistent with the definition of a chain. Thus every set in the partitioning is a valid chain or cycle.

The algorithm runs in polynomial time as implied by the following discussion: The removal of 2-cycles is done in linear time in the input. To check the existence of a cycle can be done with \(L_{\min}\) calls to \(BFS\) or any shortest
path algorithm, therefore in polynomial time. At each node maximum bi-
partite matching can be found in polynomial time using any maximum flow
algorithm, all other operations can be done in constant time.

2.3.4 Approximation Ratio

Let $ALG$ be any deterministic algorithm solving an approximation problem.
It is customary to denote by $ALG(I)$ or simply $ALG$ the cost of the solution
of algorithm $ALG$ on instance $I$. Similarly $OPT(I)$ or simply $OPT$ is the
cost of an optimal solution.

**Lemma 2.3.1** If there exist two lightpaths $p_1, p_2$ forming a cycle, there is an
optimal solution in which they form a cycle.

**Proof:** Consider an optimal solution $OPT$ in which $p_1$ and $p_2$ do not form
a cycle. In this solution $p_1$ and $p_2$ should be in different chains or cycles $c_1$
and $c_2$. We build a new solution $OPT'$ by taking all the chains and cycles
of $OPT$ except $c_1$ and $c_2$ and the cycle $c'_1 = (p_1, p_2)$ and $c'_2 = c_1 \cup c_2 \setminus c'_1$.
Consider three cases:

- Both $c_1$ and $c_2$ are cycles In this case $c'_1$ and $c'_2$ are cycles, thus $OPT = OPT'$.

- $c_1$ is a cycle, $c_2$ is a chain In this case $c'_1$ is a cycle and $c'_2$ is a chain,
  again $OPT = OPT'$.

- Both $c_1$ and $c_2$ are chains In this case $c'_1$ is a cycle and $c'_2$ forms at most
two chains $OPT' \leq OPT$.

In all cases $OPT' \leq OPT$, therefore optimal.

□

It follows from the above lemma that the first step of the preprocessing
phase removes cycles which are guaranteed to be in an optimal solution. In
other words if we can find an optimal solution for the rest of the problem,
our solution will be optimal. Let $P_2$ be set of lightpaths removed in the first
step of the preprocessing. Then:

$$PAF(N, P) = |P_2| + PAF(N, P \setminus P_2)$$
$$OPT(N, P) = |P_2| + OPT(N, P \setminus P_2)$$
$$\frac{PAF(N, P \setminus P_2)}{OPT(N, P \setminus P_2)} \geq 1$$

27
Therefore:

\[
P_{AF}(N, P) = \frac{|P_2| + P_{AF}(N, P \setminus P_2)}{|P_2| + OPT(N, P \setminus P_2)} \leq \frac{P_{AF}(N, P \setminus P_2)}{OPT(N, P \setminus P_2)}
\]

We conclude that the approximation ratio of the algorithm on an instance without 2-cycles can be only worse than a corresponding instance with 2-cycles. Without loss of generality, in the sequel we assume that no two lightpaths in the input form a cycle, or in other words, there are no 2-cycles in the instance.

During the execution of the algorithm each occurrence of an unmatched or failure event determines the end of a chain \( c_i \) and the start of a chain \( c_j \). In this case we write:

- \( c_i \prec_F c_j \)
- \( c_i \prec_U c_j \)

depending on the event occurred.

For every start (resp. end) event we introduce the dummy chain \( s_i \) (resp. \( e_i \)) and write:

- \( s_i \prec_S c_i \)
- \( c_i \prec_E e_i \)

**Observation 2.3.1** A chain \( c \) occurs once at the right side of a \( \prec \) relation and once at the left side of a \( \prec \) relation.

**Proof:** A chain \( c \) participates in a left (resp. right) side of a \( \prec \) relation, as a result of an event generated at node \( e(c) \) (resp. \( s(c) \)). It can be seen by code inspection that a chain is either extended or is involved in exactly one event.

\[ \square \]

**Observation 2.3.2** A dummy chain \( s_i \) (resp. \( e_i \)) occurs once at the right (resp. left) side of a \( \prec \) relation and never at the right (resp. left) side of a \( \prec \) relation.

Because of the preceding observations, the graph of the \( \prec \) relation can be partitioned into cycles and maximal chains. Moreover the maximal chains start with \( s_i \) nodes and end with \( e_i \) nodes.
Definition 2.3.1 In order to avoid confusion we will call the chains/cycles of the $\prec$ relation super chains (cycles) and will denote them by capital letters $(C_1, C_2, \ldots)$.

Let $C_i$ be a super cycle/chain. $P_i$ is the set of lightpaths in the super cycle/chain, namely $P_i \overset{def}{=} \bigcup C_i$.

$U_i$ (resp. $F_i$, $S_i$, $E_i$) is the number of the $\prec_U$ (resp. $\prec_F$, $\prec_S$, $\prec_E$) relationships in $C_i$. Note that:

$$S_i = E_i = \begin{cases} 1 & \text{if } C_i \text{ is a super chain} \\ 0 & \text{otherwise} \end{cases}$$

$U \overset{def}{=} \sum U_i$ and $F \overset{def}{=} \sum F_i$ are the total number of $\prec_U$ and $\prec_F$ relationships, or in other words the number of times event $U$ and event $F$ happen respectively.

$S \overset{def}{=} \sum S_i$ and $E \overset{def}{=} \sum E_i$ are the total number of $\prec_S$ and $\prec_E$ relationships, or in other words the number of times start and end events happen respectively. Note that $S = E$ which is in turn equal to the number of super chains. Moreover $E = \sum_{i=0}^{N-1} \max(0, |\sigma_i | - \tau_i )$. Note for any given instance $E$ is constant and does not depend on the output.

Let $R = |P|$, $C$ the number of cycles removed in the preprocessing phase of the algorithm and $R_C$ the number of lightpaths in these cycles.

Let $C^*$ be the number of cycles in an optimal solution, and $R_C^*$ be the number of lightpaths in these cycles.

Lemma 2.3.2

$$2E + 2U + 3F + 2R_C \leq 2R \quad (2.1)$$

Proof: Consider a super chain or cycle $C_i$ in the output of the algorithm. For each $\prec_U$ relationship in $C_i$ there are at least two lightpaths in $P_i$ which are involved. For each $\prec_F$ relationship there are at least three lightpaths involved. For each $\prec_S$ or $\prec_E$ relationship there is at least one lightpath involved. Each lightpath in $P_i$ is exactly in one chain thus involved in two relationships. Therefore: $2U_i + 3F_i + S_i + E_i \leq 2|P_i|$. Summing up over all the super chains/cycles we obtain: $S + E + 2U + 3F = 2U + 3F + 2E \leq 2 \sum |P_i|$. The lightpaths which are involved in events are those who survived the preprocessing phase, therefore in $\sum |P_i| = R - R_C$.

□
Lemma 2.3.3

\[ U + F + 2C \leq 2L_{\min} \]  

(2.2)

Proof: Consider the set \( Y_i \) of lightpaths crossing an edge \((i, i+1)\) such that \( |Y_i| = L_{\min}\). Every lightpath is involved in two relationships. This is in particular true for the lightpaths in \( Y_i \). On the other hand each \( U \) or \( F \) event involves at least one lightpath from \( Y_i \) which survived the preprocessing phase. The number of these lightpaths is \( L_{\min} - C \). Therefore \( U + F \leq 2(L_{\min} - C) \).

\[ \Box \]

Lemma 2.3.4

\[ OPT \geq R + E + U - R_C + C. \]  

(2.3)

Proof: At each node \( i \), the paths of \( \sigma_i \) can be classified as follows:

- \( R_C(i) \) paths removed by the preprocessing phase.
- \( E(i) + U(i) \): paths which did not take part in the maximum matching.
- \( |MM_i| \) paths participating in the maximum matching.

Therefore, \( |MM_i| = |\sigma_i| - R_C(i) - E(i) - U(i) \). Summing up over all nodes \( i \) and defining \( MM \) as \( \sum_{i=0}^{n-1} |MM_i| \) we have:

\[ MM = R - R_C - E - U. \]  

(2.4)

Consider a maximum matching \( MM_i \) of the node graph after the removal of the \( C \) cycles of \( \text{PAF} \) and a maximum matching \( MM^0_i \) of the node graph of node \( i \) before any preprocessing. Each pair of paths \( p_1 \in \sigma_i, p_2 \in \tau_i \) removed by the preprocessing phase reduces the value of the maximum matching at most by two. Therefore

\[ |MM_i| \geq |MM^0_i| - 2R_C(i) \]

Summing over all nodes we have \( MM \geq MM^0 - 2R_C \). In fact we will later prove:

\[ MM \geq MM^0 - 2R_C + C \]  

(2.5)

30
On the other hand as it is pointed out in [LLWF00] and [EMZ02]:\[\text{OPT} \geq 2R - MM^0\]

Combining with (2.5) and substituting the value of $MM$ in (2.4) we get:
\begin{align*}
\text{OPT} & \geq 2R - MM - 2R_C + C \\
& = 2R - (R - R_C - E - U) - 2R_C + C \\
& = R + E + U - R_C + C
\end{align*}
as required.

It remains to prove inequality (2.5). It is sufficient to show that in each one of the $C$ cycles removed in the preprocessing phase, there is at least one path that does not reduce $MM$ by two. Assume, by contradiction that there is a cycle $p_1, p_2, ..., p_k$ removed in the preprocessing phase such that each successive pair of paths $p_{i-1}, p_i$ reduces $MM$ by two. This means that both $p_{i-1}$ and $p_i$ are matched to two paths by OPT. Let these paths be $b_{i-1}$ and $a_i$ respectively (see Figure 2.3). Considering the fact that $a_i, p_i, b_i$ are part of a chain/cycle of OPT:

\[\text{len}(a_i) + \text{len}(p_i) + \text{len}(b_i) \leq N\]

Summing over all nodes $v_1, v_2, ..., v_k$:
\[\sum \text{len}(a_i) + \sum \text{len}(p_i) + \sum \text{len}(b_i) \leq Nk\]

\[\sum \text{len}(p_i) = N\] because they form a cycle, therefore
\[\sum \text{len}(a_i) + \sum \text{len}(b_i) \leq N(k - 1)\]
On the other hand:

\[ \text{len}(a_i) + \text{len}(b_{i-1}) > N \]

for, otherwise they can be added to any matching which do not include any of them, and \( MM_i \) is not reduced by two. Summing over all nodes, we get:

\[ \sum \text{len}(a_i) + \sum \text{len}(b_i) > Nk \]

a contradiction. \( \Box \)

Consider an edge \( e \) with maximum load (i.e. \( L_{\max} \)). Exactly \( C^* \) of the paths using it appear in cycles of \( OPT \). The rest, are in distinct chains of \( OPT \). Therefore there are at least \( L_{\max} - C^* \) chains in \( OPT \).

\[
OPT \geq R + L_{\max} - C^* \geq R + L_{\min} - C \cdot (2.6)
\]

Any algorithm should use at least one ADM for the beginning of a lightpath and at least \( |\sigma_i| - |\tau_i| \) ADMs at the end of lightpaths ending at node \( i \). Therefore:

\[
OPT \geq R + \sum_{i=0}^{N-1} \max(0, |\sigma_i| - |\tau_i|) = R + E
\]

By our assumption all cycles consist of at least 3 lightpaths, thus:

\[
3C \leq R_C \quad (2.7) \\
3C^* \leq R_C^* \quad (2.8)
\]

The preprocessing phase removes a maximal number of cycles. Therefore, each cycle of \( OPT \) should contain at least one lightpath from the cycles of \( PAF \), for, otherwise there would be an entire cycle which is not removed by \( PAF \) in the preprocessing phase. This would be a contradiction to the maximality of the cycles removed in the preprocessing phase. We conclude:

\[
C^* \leq R_C \cdot (2.9)
\]

Obviously, the number of lightpaths in the chains of any solution is at least as the load induced by them on any edge, in particular on an edge of minimum load. Thus:

\[
R - R_C^* \geq L_{\max} - C^* \geq L_{\min} - C^* \quad (2.10)
\]

32
Theorem 2.3.1

$$\rho_{PAF} \leq \frac{11}{7}$$

Proof: Assume the contrary, i.e. that for some $\rho > 11/7$, $PAF/OPT \geq \rho$. It is easy to see from the algorithm that $PAF = R + U + F + E$. Then:

$$R + U + F + E > \rho \cdot OPT$$ (2.11)

We substitute $\rho = 11/7$ and we seek for values of the variables which may satisfy all the constraints found so far. It can be shown that, the resulting Linear Program has no feasible solution. This LP is given in Appendix A.1. Therefore the corresponding ILP does not have a solution, a contradiction.

2.3.5 A Lower Bound

The maximum disjoint cycles problem in a ring (MDCR), is the problem of partitioning $P$ into chains and cycles, such that the number of cycles is maximum. Any algorithm solving the minimum ADM problem, is also solving the MDCR problem. Its performance with respect the two problems may of course, be different.

In [CW02b] the PIM (Preprocessed Iterative Matching) algorithm is presented and proven to have an approximation ratio between $\frac{4}{3}$ and $\frac{3}{2}$. A closer look to their proof reveals the following, general lower bound:

Any algorithm $ALG$ with no performance guarantee on the $MDCR$ problem beyond maximality, has approximation ratio no better than $4/3$ for the minimum ADM problem.

The following improved lower bound is recently established in [EL04]:

Lemma 2.3.5 Any algorithm $ALG$ with no performance guarantee on the $MDCR$ problem beyond maximality, has approximation ratio no better than $3/2$ for the minimum ADM problem.

Corollary 2.3.2 $1.5 \leq \rho_{PAF} \leq 11/7 < 1.572$. 

33
2.4 Algorithms with Improved Preprocessing - IPAF and IEMZ

2.4.1 The Motivation

In this section we develop algorithms with approximation ratio better than 3/2. In view of Lemma 2.3.5, this necessarily requires a better performance guarantee on the MDCR problem.

Other algorithms which do not have the preprocessing phase are proven in [CW02b] to have approximation ratio of exactly \( \frac{3}{2} \). On the other hand PAF has approximation ratio at most \( \frac{11}{7} \) which is better than \( \frac{5}{3} \). This clearly indicates that the preprocessing phase improves the algorithm. Thus it is natural to investigate the approximability of the problem with respect to this preprocessing phase, more precisely the part of the solution which consists of cycles.

In this section we analyze the performance of PAF for special cases and present an improved version of it which is essentially PAF with improved preprocessing phase and then combine the improved preprocessing phase with an algorithm with approximation ratio \( \frac{3}{2} \) and manage to reach an approximation ratio of \( \frac{10}{7} + \epsilon \).

**Lemma 2.4.1** A 2 approximation to the MDCR problem, implies a 7/5 approximation to the minimum ADM problem.

**Proof:** We add the constraint \( C^* \leq 2C \) to the LP in the proof of Theorem 2.3.1 and we show that the resulting LP has no solution for \( \rho > \frac{7}{5} \).

\[ \square \]

**Corollary 2.4.1** PAF is a \( \frac{7}{5} \)-approximation to the minimum ADM problem for instances with no cycles.

The above result indicates that a better approximation to the MDCR problem would lead a better approximation to the minimum ADM problem. We proceed with an algorithm with a better preprocessing phase.
2.4.2 Algorithm $IPA F_k$

$IPA F_k$

- Run preprocessing phase A of $PAF$
- Calculate all the possible cycles $c$ such that $|c| \leq k$
- Find a maximum set packing (MSP) of these cycles
- Remove the maximum packing from $L$
- Run preprocessing phase B of $PAF$
- Run processing phase of $PAF$

2.4.3 Analysis of $IPA F_k$

In the sequel short cycles are cycles containing at most $k$ lightpaths and long cycles are cycles with at least $k + 1$ lightpaths. The calculation of all the short cycles may be done by choosing an edge $e$ such that $l(e) = L_{\min}$ and trying all the possible clockwise extensions of the lightpaths passing through this edge. We repeat this process $k - 1$ times. The number of cycles with $k$ lightpaths or less is at most $L_{\min}(L_{\max})^{k-1}$, in other words there are a polynomial number of cycles, and they can be computed in polynomial time as described. Moreover each cycle is as a set with at most $k$ elements. A $(k/2 + \varepsilon)$-approximation for the MSP problem is given in ([HS89]), for all $k \geq 3$. For any fixed $\varepsilon$ and $k$, the running time of the algorithm is polynomial.

Note that for instances with cycles of at most 4 paths, our preprocessing is a 2-approximation for the $MDCR$ problem, then $IPA F_k$ is a $7/5$-approximation to the minimum ADM problem, for these instances. Generally:

**Theorem 2.4.1**

$$\rho_{IPA F_k} \leq 1.48$$

**Proof:** We define the following variables:

$C^-_k$ (resp. $C^+_k$) is the number of short (resp. long) cycles in an optimal solution. Similarly we define $C^-_k$ and $C^+_k$ are defined similarly with respect to the solution obtained by $IPA F_k$. In the same way we define $R^-_{C^-}$, $R^+_{C^+}$, $R_{C^-}$ and $R_{C^+}$ as the number of lightpaths in these cycles. The following
equalities are immediate:

\[
C^* = C^* - C^* \\
C = C_+ + C_+ \\
R^*_C = R^*_C - R^*_C \\
R_C = R_C - R_C
\]

as are the following inequalities:

\[
3C^* \leq R^*_C \leq kC^* \\
3C_+ \leq R_+ \leq kC_+ \\
(k + 1)C_+ \leq R_+ \\
(k + 1)C_+ \leq R_+
\]

Let \( \bar{C}_- \) be the maximum number of disjoint short cycles. The MSP algorithm guarantees \( C_ - \geq \frac{\bar{C}_-}{k/2 + \epsilon} \) for every \( \epsilon > 0 \). On the other hand the optimal solution can not include more than \( \bar{C}_- \) short cycles. Thus \( C_ - \geq \frac{C^*}{k/2 + \epsilon} \). For all \( \epsilon'' > 0 \), we have:

\[
(k + \epsilon')C_- \geq 2C^*
\]

We extend the linear program in the proof of Theorem 2.3.1 by adding the above constraints. It can be shown that for \( k = 5 \) and \( \epsilon > 1.48 \), the resulting linear program has no feasible solution. This linear program is given in Appendix A.2.

\[\square\]

2.4.4 Algorithm IEMZ_k

The following algorithm has the same preprocessing phase as IPAF_k, it solves the remaining instance using algorithm EMZ introduced in [EMZ02].

\[
IEMZ_k \ (N, P) \{
\text{Run preprocessing of } IPAF_k \\
\text{For each path } p \in P \text{ do } c(p) = (p) \\
\text{For each node } i \text{ from 1 clockwise to } 0 \text{ do} \\
\quad \text{Find a maximum matching } MM_i \text{ of the node graph } G_i.
\}
\]
$$\tau'_i = \text{The unmatched nodes of } \tau_i.$$
$$\sigma'_i = \text{The unmatched nodes of } \sigma_i.$$
$$G'_i = \text{The complete bipartite graph } (\tau'_i, \sigma'_i, \tau'_i \times \sigma'_i).$$
Find a maximum matching $MM'_i$ OF $G'_i$.
$$\tau''_i = \text{The unmatched nodes of } \tau'_i.$$
$$\sigma''_i = \text{The unmatched nodes of } \sigma'_i.$$  
For each edge $(a, b) \in MM_i$ UNION $(c(a), c(b))$
For each edge $(a, b) \in MM'_i$ unmatched
For each b in $\tau''_i$ start
For each b in $\sigma''_i$ end
}

For each chain/cycle $c$ do {  
Let $c = p_1, p_2, \ldots, p_k$

i=1;
For j=1 to k{
If $p_j+1 \approx p_k$ {
    split $c$ into two chains such that
    $p_j$ and $p_{j+1}$ are in different chains
    $i = j + 1$
    failure
}
}
}

2.4.5 Analysis of $IEMZ_k$

Lemma 2.4.2

$$2F + E + U + R_C \leq R$$

Proof: In the second phase of $EMZ$ algorithm, there is a one-to-one mapping from the $F$ events to the successful matchings. This can be seen by the following simple argument taken from [EMZ02]: The first matching of a chain can not be broken, because otherwise the total length of paths involved are summing up to at least $N + 1$, which means that there is no edge joining them in the node graph, therefore they can not be part of a matching. Therefore to any broken matching ($F$ event) there is a corresponding
unbroken matching. In our notation this is denoted as:

\[ F \leq MM - F. \]

Substituting the value of \( MM \) we get:

\[ 2F \leq R - R_C - E - U. \]

\[ \square \]

**Theorem 2.4.2**

\[ \rho_{IEMZ_5} \leq 10/7 + \epsilon \]

**Proof:** It is easy to show that all the inequations that hold form \( IPAF_k \) hold for \( IEMZ_k \) too, except Lemma 2.3.2. We replace the corresponding constraint in the linear program in the proof of Theorem 2.4.1 with the result of Lemma 2.4.2 and get a new linear program. This linear program has a solution for \( \rho = 10/7 \) but no solution for any \( \rho > 10/7 \). The linear program is given in Appendix A.3. We assume a solution for \( \rho = 10/7 + \delta \) for any \( \delta > 0 \), we compare it to a solution of \( \rho = 10/7 \). Considering the tightly satisfied constraints, we reach a contradiction.

\[ \square \]

### 2.5 Simulation Results, Conclusion and Possible Improvements

The calculation of OPT is in NP-complete. Therefore, we compared the performance of \( IPAF_5 \) and PIM on 200 random instances with \( 10 \leq n \leq 16 \) and \( 20 \leq R \leq 150 \). \( IPAF_5 \) led to better results for almost all the instances, where the difference in the performance grows with the size of the input, i.e. the number of the lightpaths.

In this work we investigated the relationship between the arc version of the minimum ADM problem and the maximum disjoint cycles problem. We saw that on instances without cycles we can obtain a \( 7/5 - approximation \) and generally we can not get better than \( 3/2 - approximation \) if we can not perform better than the trivial greedy algorithm for the \( MDCR \) problem. We
presented the algorithm $IPAF_5$ which has a provable upper bound of 1.48. Finally we presented algorithm $IEMZ_k$ which has the same preprocessing phase and proved it to have an approximation ratio at most $10/7 + \epsilon$.

A possible improvement to the preprocessing phase is to modify it to choose the value of $k$ as a function of $R/L_{\text{min}}$ or alternatively to try different values for $k$ and get the best solution among them. This direction might lead to a provable increase in the performance.

Another possible direction is to improve the preprocessing phase by replacing the algorithm [HS89] which solves the general MSP problem for $k-sets$ with an algorithm that achieves better performance by taking advantage of the properties of the $k-cycles$.

The $10/7$-approximation algorithm presented in [EL04] has a preprocessing phase removing short cycles and paths, whereas our preprocessing phase removes short cycles only, thus answering affirmatively an open question mentioned in [EL04].

The techniques used in our work can be combined with the techniques used used in [EL04] with possibly leading to an algorithm with better approximation ratio.
Chapter 3

Minimizing the number of ADMs in General Networks

3.1 Introduction

3.1.1 Background

Given a WDM network $G = (V, E)$ and a set of full-duplex lightpaths $P = \{p_1, p_2, ..., p_N\}$ of $G$, the wavelength assignment (WLA) task is to assign a wavelength to each lightpath $p_k$.

In the following discussion we also assume that each lightpath $p \in P$ is contained in a cycle of $G$. As already mentioned, each lightpath $p$ uses two ADMs, one at each endpoint. Although only the downstream ADM function is needed at one end and only the upstream ADM function is needed at the other end, full ADMs will be installed on both nodes in order to complete the protection path around some ring. The full configuration would result in a number of SONET rings. It follows that if two adjacent lightpaths are assigned the same wavelength, then they can be used by the same SONET ring and the ADM in the common node can be shared by them. This would save the cost of one ADM. An ADM may be shared by at most two lightpaths. A more detailed technical explanation can be found in [GLS98].

Lightpaths sharing ADMs in a common endpoint can be thought as concatenated, so that they form longer paths or cycles. Each of these longer paths/cycles does not use any edge $e \in E$ twice, for, otherwise they cannot use the same wavelength and this is a necessary condition to share ADMs.
3.1.2 Previous Work

Minimizing the number of electronic switches in optical networks is a main research topic in recent studies. The problem was introduced in [GLS98] for ring topology. An approximation algorithm for ring topology with approximation ratio of $3/2$ was presented in [CW02b], and was improved in [SZ04, EL04] to $10/7 + \epsilon$ and $10/7$, respectively.

For general topology [EMZ02] describes an algorithm with approximation ratio of $8/5$. The same problem was studied in [CFW02] and an algorithm was presented that has a preprocessing phase during which cycles of length at most $l$ are included in the solution; this algorithm was shown to have a performance guarantee of

$$OPT + \frac{1}{2}(1 + \epsilon)N, \quad 0 \leq \epsilon \leq \frac{1}{l + 2} \quad (3.1)$$

where $OPT$ is the cost of an optimal solution, $N$ is the number of lightpaths, for any given odd $l$. The dominant part in the running time of the algorithm is the preprocessing phase, which is exponential in $l$.

3.1.3 Our Contribution

We improve the analysis of the algorithm of [CFW02] in several stages. We first discuss the algorithm without preprocessing and prove a performance of $OPT + 0.6N$, namely

$$OPT + \frac{1}{2}(1 + \epsilon)N, \quad \epsilon = \frac{1}{5}. \quad (3.2)$$

Specifically, we show that the algorithm guarantees to satisfy an upper bound of $OPT + 0.6N$, and we show that this bound is tight by demonstrating an infinite family of instances for which the performance of the algorithm is exactly $OPT + 0.6N$.

We then discuss the algorithm with preprocessing of cycles of length at most $l$ and prove a performance of

$$OPT + \frac{1}{2}(1 + \epsilon)N, \quad \frac{1}{2l + 3} \leq \epsilon \leq \frac{1}{2(l + 2)}. \quad (3.3)$$

Specifically, we show that the algorithm guarantees to satisfy an upper bound of $OPT + \frac{1}{2}(1 + \epsilon)N$, where $\epsilon \leq \frac{1}{2l + 2}$, and we demonstrate an infinite family
of instances for which the performance of the algorithm is \( OPT + \frac{1}{2}(1 + \epsilon)N \), where \( \epsilon \geq \frac{1}{2} \).

Our analysis sheds more light on the structure and properties of the algorithm, by closely examining the structural relation between the solution found by the algorithm vs an optimal solution, for any given instance of the problem.

As the running time of the algorithm is exponential in \( l \), our results imply an improvement in the analysis of the running time of the algorithm. For any given \( \epsilon > 0 \), the exponent of the running time needed to guarantee the approximation ratio \( (3 + \epsilon)/2 \) is reduced by a factor of \( 3/2 \).

In our analysis we use a novel technique. We first develop it for the case of the algorithm without preprocessing, where we prove the tight bound \((3.2)\), we then extend it to the case of the algorithm with preprocessing, where we prove the bound \((3.3)\). In the development of our bounds we use a purely combinatorial problem, which is of interest by itself.

In Section 3.2 we describe the problem and some preliminary results. The algorithm without the preprocessing phase is presented in Section 3.3 and the algorithm with the preprocessing phase is presented in Section 3.4. We conclude with discussion and open problems in Section 3.5.

### 3.2 Problem Definition and Preliminary Results

#### 3.2.1 Problem Definition

An instance \( \alpha \) of the problem is a pair \( \alpha = (G, P) \) where \( G = (V, E) \) is an undirected graph and \( P \) is a set of simple paths in \( G \). Given such an instance we define the following:

**Definition 3.2.1** The paths \( p, p' \in P \) are conflicting or overlapping if they have an edge in common. This is denoted as \( p \sim p' \). The graph of the relation \( \sim \) is called the conflict graph of \((G, P)\).

**Definition 3.2.2** A proper coloring (or wavelength assignment) of \( P \) is a function \( w : P \mapsto \mathbb{N} \), such that \( w(p) \neq w(p') \) whenever \( p \sim p' \).

Note that \( w \) is a proper coloring if and only if for any color \( \lambda \in \mathbb{N} \), \( w^{-1}(\lambda) \) is an independent set in the conflict graph.
Definition 3.2.3 A valid chain (resp. cycle) is a path (resp. cycle) formed by the concatenation of distinct paths \( p_0, p_1, \ldots, p_{k-1} \in P \) that do not go over the same edge twice. Note that the paths of a valid chain (resp. cycle) constitute an independent set of the conflict graph.

Definition 3.2.4 A solution \( S \) of an instance \( \alpha = (G, P) \) is a set of chains and cycles of \( P \) such that each \( p \in P \) appears in exactly one of these sets.

In the sequel we introduce the shareability graph, which together with the conflict graph constitutes another (dual) representation of the instance \( \alpha \). Except for one particular case, we will use the dual representation of the problem.

Definition 3.2.5 The shareability graph of an instance \( \alpha = (G, P) \), is the edge-labelled multi-graph \( G_\alpha = (P, E_\alpha) \) such that there is an edge \( e = (p, q) \) labelled \( u \) in \( E_\alpha \) if and only if \( p \neq q \), and \( u \) is a common endpoint of \( p \) and \( q \) in \( G \).

Example: Let \( \alpha = (G, P) \) be the instance in Figure 3.9. Its shareability graph \( G_\alpha \) is the graph at the left side of Figure 3.10. In this instance \( P = \{a, b, c, d, e\} \), and it constitutes the set of nodes of \( G_\alpha \). The edges together with their labels are \( E_\alpha = \{(b, c, u), (d, e, v), (a, c, w), (a, b, x), (a, d, x), (b, e, x)\} \). Because \( b \) and \( c \) can be joined in their common endpoint \( u \), etc.. Note that, for instance \((b, d, x) \notin E_\alpha\), because although they share a common endpoint \( x \), they cannot be concatenated, as they have the edge \((x, u)\) in common. The corresponding conflict graph is the graph at the right side of Figure 3.10. It has the same node set and the edge set is \( \{(c, d), (b, d), (c, e), (a, e)\} \). The paths \( c, d \in P \) are conflicting because they have a common edge, i.e. \((u, v)\), etc..

Note that the edges of the conflict graph are not in \( E_\alpha \). This immediately follows from the definitions.

Note also that, for any node \( v \) of \( G_\alpha \), the set of labels of the edges adjacent to \( v \) is of size at most two.

Definition 3.2.6 A valid chain (resp. cycle) of \( G_\alpha \) is a simple path \( p_0, p_1, \ldots, p_{k-1} \) of \( G_\alpha \), such that any two consecutive edges in the path (resp. cycle) have distinct labels and its node set is properly colorable with one color (in \( G \)), or in other words constitutes an independent set of the conflict graph.
Note that the valid chains (resp. cycles) of $G_\alpha$ correspond to valid chains (resp. cycles) of the instance $\alpha$. In the above example the chain $a, d$ which is the concatenation of the paths $a$ and $d$ in the graph $G$, corresponds to the simple path $a, d$ in $G_\alpha$ and the cycle $a, b, c$ which is a cycle formed by the concatenation of three paths in $G$ corresponds to the cycle $a, b, c$ in $G_\alpha$. Note that no two consecutive labels are equal in this cycle. On the other hand the paths $b, a, d$ can not be concatenated to form a chain, because this would require the connection of $a$ to both $b$ and $d$ at node $x$. The corresponding path $b, a, d$ in $G_\alpha$ is not a chain because the edges $(b, a)$ and $(a, d)$ have the same label, namely $x$.

**Definition 3.2.7** The sharing graph of a solution $S$ of an instance $\alpha = (G, P)$, is the following subgraph $G_{\alpha, S} = (P, E_S)$ of $G_\alpha$. Two lightpaths $p, q \in P$ are connected with an edge labelled $u$ in $E_S$ if and only if they are consecutive in a chain or cycle in the solution $S$, and their common endpoint is $u \in V$. We will usually omit the index $\alpha$ and simply write $G_S$. $d(p)$ is the degree of node $p$ in $G_S$.

In our example, $S = \{(a, d)(b, e), (c)\}$ is a solution with three chains. The sharing graph of this solution is depicted in Figure 3.11. Note that for a solution consisting of chains of size at most two, the distinct labelling condition is satisfied vacuously, and the independent set condition is satisfied because no edge of $G_\alpha$ can be edge of the conflict graph.

In our example, another possible solution is $S'$ consisting of the cycles $(a, b, c)$ and $(d, e)$, the corresponding sharing graph contains two cycles $(a, b, c)$ and $(d, e)$. In this case we need to check that the remaining two conditions are satisfied. Indeed, no two consecutive labels are equal and each cycle constitutes an independent set of the conflict graph.

We define:

$$\forall i \in \{0, 1, 2\}, \quad D_i(S) \overset{\text{def}}{=} \{p \in P \mid d(p) = i\}$$

and

$$d_i(S) \overset{\text{def}}{=} |D_i(S)|.$$

Note that $d_0(S) + d_1(S) + d_2(S) = |P| = N$.

An edge $(p, q) \in E_S$ with label $u$ corresponds to a concatenation of two paths with the same color at their common endpoint $u$. Therefore these two endpoints can share an ADM operating at node $u$, thus saving one ADM. We
conclude that every edge of $E_S$ corresponds to a saving of one ADM. When no ADMs are shared, each path needs two ADMs, for a total of $2N$ ADMs. Therefore the cost of a solution $S$ is

$$\text{cost}(S) = 2N - |E_S|.$$  

The objective is to find a solution $S$ such that $\text{cost}(S)$ is minimum, in other words $|E_S|$ is maximum.

### 3.2.2 Preliminary Results

Given a solution $S$, $d(p) \leq 2$ for every node $p \in P$. Therefore, the connected components of $G_S$ are either paths or cycles. Note that an isolated vertex is a special case of a path. Let $P_S$ be the set of the connected components of $G_S$ that are paths. Clearly, $|E_S| = N - |P_S|$. Therefore

$$\text{cost}(S) = 2N - |E_S| = N + |P_S|$$

Let $S^*$ be a solution with minimum cost. For any solution $S$ we define

$$\epsilon(S) \stackrel{\text{def}}{=} \frac{d_0(S) - d_2(S) - 2|P_{S^*}|}{N}.$$ 

**Lemma 3.2.1** For any solution $S$

$$\text{cost}(S) = \text{cost}(S^*) + \frac{1}{2}N(1 + \epsilon(S)).$$

**Proof:** As mentioned before, $|E_{S^*}| = N - |P_{S^*}|$. On the other hand $2|E_S|$ is the sum of the degrees of the nodes in $G_S$, namely

$$2|E_S| = d_1(S) + 2d_2(S) = N - d_0(S) + d_2(S)$$

We conclude:

$$\text{cost}(S) - \text{cost}(S^*) = |E_{S^*}| - |E_S| = N - |P_{S^*}| - \frac{N - d_0(S) + d_2(S)}{2}$$

$$= \frac{N}{2} + \frac{d_0(S) - d_2(S) - 2|P_{S^*}|}{2}$$

$$= \frac{1}{2}N \left(1 + \frac{d_0(S) - d_2(S) - 2|P_{S^*}|}{N}\right)$$

45
The following definition extends the concept of a chord from cycles to paths.

**Definition 3.2.8** Given an instance $\alpha = (G,P)$ and a solution $S$ of $\alpha$, an edge $(p,q)$ of $G_{\alpha}$ is a chord of $S$ if both $p$ and $q$ are in the same connected component of $G_{S}$ and $(p,q) \notin E_{S}$.

**Lemma 3.2.2** For every instance $\alpha = (G,P)$ and there is an optimal solution $S^{*}$ without chords.

**Proof:** Note that any two solutions $S_{1}, S_{2}$ of $\alpha$ such that $\text{cost}(S_{1}) = \text{cost}(S_{2})$, have the same number of chains, whereas the number of cycles may differ. Let $S^{*}$ be a solution with maximum number of cycles among the solutions with minimum cost, i.e. optimal. We will prove that $S^{*}$ satisfies the claim.

In this paragraph we work on the graph $G$. We claim that there is no node $v$ and no chain (resp. cycle) $C$ of $S^{*}$, such that $v$ is used more than once as an endpoint of a paths in $C$. Assume the contrary. Consider two occurrences of $v$ in $C$ (see Figure 3.1). It is impossible that $C$ is a path and $v$ terminates both ends $C$. In this case $C$ can be closed to a cycle, and get a solution with one path less, contradicting the optimality of $S^{*}$. Consider the sequence of paths between these two occurrences of $v$. This is a valid cycle, say $C'$. Consider the solution $S'$ obtained by taking $S^{*}$ and separating $C$ into two parts. The first part is $C'$ and the second part is the sequence obtained by the concatenation of the paths before the first occurrence of $v$ with the paths after the second occurrence of $v$, where one of these but not both may be empty. $S'$ has the same number of paths as $S^{*}$, therefore $\text{cost}(S') = \text{cost}(S^{*})$, therefore optimal. Moreover, $S'$ has one more cycle than $S^{*}$, contradictory to the way $S^{*}$ was chosen.

Assume that $(p,q)$ is a chord of $S^{*}$. Let $x$ be its label. Then $x$ is an endpoint of both $p$ and $q$. Because $(p,q)$ is a chord, $(p,q) \notin E_{S^{*}}$, in other words $p$ and $q$ do not have the node $x$ as common endpoint in this connected component. Then $x$ appears at least twice in the connected component, a contradiction. Therefore there are no chords of $S^{*}$.

The above proof is the last time that we used the primal representation of the problem. In the sequel we will always use the dual representation. Henceforth, an element $p$ of $P$ is referred as a node (of $G_{\alpha}$), and a path refers to a path of $G_{\alpha}$.
The occurrences of $v$ are endpoints of a path.

At least one occurrence of $v$ is not an endpoint.

Figure 3.1: Nodes are not repeated in a connected component.
3.3 Algorithm MM

In this section we give a short description of the algorithm in [CW02b], without a preprocessing phase. The algorithm begins with chains consisting of a single node (which are always valid). At each iteration, we try to combine a maximum number of pairs of chains to obtain longer chains (in fact, less chains). This is done by constructing an appropriate graph and computing a maximum matching on it. The algorithm ends when the maximum matching is empty, namely no two chains can be combined into a longer chain.

Phase 0) \( E_S = \emptyset \)

// the chains of \( G_S \) are isolated nodes.

Phase 1) Do {

Build the graph \( G'_\alpha \) in which each node is a chain of \( G_S \) and there is an edge labelled \( u \) between two chains if and only if the chains can be merged into one bigger chain by joining them at a common endpoint \( u \).

// In the first iteration \( G'_\alpha = G_\alpha \)

Find a maximum matching \( MM \) of \( G'_\alpha \).

For each edge \( e = (c, c') \) of \( MM \) labelled \( u \) do {

Merge the corresponding chains into one chain by joining them in the common endpoint \( u \).

// Note that the chains may have an additional endpoint, say \( v \), which is not affected

} Until \( MM = \emptyset \).

3.3.1 Correctness

After Phase 0, the chains of \( S \) consist of single nodes. Trivially, these are valid chains. At each iteration of Phase 1, a new chain is constructed only if it is valid, because edges are added to \( G'_\alpha \) only if the corresponding chains can be merged into one chain. Each edge of a matching represents a valid merging operation. Moreover two such valid operations do not affect each other, because each such operation is performed on two chains matched by an edge of some matching. Therefore after each iteration the solution consists of valid chains.
3.3.2 Analysis

We will modify the algorithm MM so that its performance can be only worse and then analyze a solution returned by the modified algorithm. We make two modifications:

- The algorithm performs only two iterations.
- In the second iteration instead of a maximum matching the algorithm finds a maximal bipartite matching where one set are isolated nodes of $G_S$ and and the second set of nodes are paths of length one in $G_S$.

After the first iteration $G_S$ contains isolated nodes and paths of length one. After the second iteration $G_S$ contains paths of length at most two.

In the sequel $S$ is a solution returned by the modified algorithm and $S^*$ is an optimal solution without chords, whose existence is guaranteed by Lemma 3.2.2.

We direct each edge of $G_{S^*}$, such that each path becomes a directed path and each cycle becomes a directed cycle. The direction chosen for every path (resp. cycle) is arbitrary. Let $\overrightarrow{G}_{S^*}$ be the digraph obtained by this process. Unless otherwise stated, $d_{in}(p)$ and $d_{out}(p)$, denote the in and out degrees of $p$ in $\overrightarrow{G}_{S^*}$, respectively. Clearly, $\forall p \in P$, $d_{in}(p) \leq 1$ and $d_{out}(p) \leq 1$. The following definitions refer to $\overrightarrow{G}_{S^*}$:

$LAST^*$ is the set of nodes that do not have successors in $\overrightarrow{G}_{S^*}$, namely

$$LAST^* \overset{\text{def}}{=} \{ p \in P | d_{out}(p) = 0 \}.$$ 

Note that $|LAST^*| = |P_{S^*}|$.

The functions $Next^*$ and $Prev^*$ are defined as expected: $Next^*$ (resp. $Prev^*$) maps a node $p$ to the next (resp. previous) node in $\overrightarrow{G}_{S^*}$ whenever such a node exists, namely:

$$Next^* : P \setminus LAST^* \mapsto P$$

and $Next^*(p)$ is the unique node $u$ such that there is an edge from $p$ to $u$ in $\overrightarrow{G}_{S^*}$. $Prev^* = Next^{*-1}$.

Let MM be the maximum matching found by the algorithm in the first iteration of Phase 1. We make the following observation:
**Observation 3.3.1**

- a) An edge \( e = (p, q) \in E_S \) such that \( d(p) = d(q) = 1 \) is in \( MM \).
- b) Let \( p, q, r \) be a maximal path of \( G_S \). We can assume that either \( e = (p, q) \in MM \) and \( e' = (q, r) \notin MM \) or vice versa.

**Proof:**

- a) Assume \( e \notin MM \), then at the end of the first iteration \( d(p) = d(q) = 0 \). This implies that \( MM \) is not a maximum matching, a contradiction.

- b) Obviously, either \( e \) or \( e' \) is in \( MM \). Otherwise one of them can be added to \( MM \) and augment it. Assume \( e \in MM \), then \( e' \notin MM \). \( MM' = MM - \{e\} \cup \{e'\} \) is a maximum matching too. As the algorithm may return any maximum matching in its first phase we may equally assume that \( MM' \) is the matching returned in the first phase and \( e \) is added to the solution in a subsequent phase.

\[ \square \]

Using the notation in Lemma 3.2.1, we will prove

**Lemma 3.3.1** For any solution \( S \) returned by algorithm \( MM \)

\[ \epsilon(S) \leq 1/5. \]

**Proof:** We partition \( D_0(S) \) into the sets \( A, B, C \) and \( D \) using the following classification procedure **CLASSIFY**:

Given \( p \in D_0(S) \), **CLASSIFY** finds a sequence \( f(p) = (p_0, p_1, \ldots) \) of elements of \( P \).

**CLASSIFY** \((p \in P) \) \{ 

- \( p_0 = p \)
- For \( i \geq 1 \) do:
  - a) If \( p_{i-1} \in D_2(S) \) then \( p \in A \), \( f(p) = (p_0, \ldots, p_{i-1}) \), **return**.
  - b) If \( p_{i-1} \in LAST^* \) then \( p \in B \), \( f(p) = (p_0, \ldots, p_{i-1}) \), **return**.

\[ \]
- c) If there is a node repeated at least twice in the sequence \( p_0, \ldots, p_{k-1} \) then \( p \in C \), \( f(p) = (p_0, \ldots, p_{k-1}) \), return.

- d) If \( i \) is even:

\[
\begin{align*}
\text{If } p_{i-1} &\in D_1(S) \text{ then } \\
p_i &\text{ is the (unique) neighbor of } p_{i-1} \text{ in } G_S, \\
\text{else } &\text{ /* } p_{i-1} \notin D_0(S) */ \\
p &\in D, \ f(p) = (p_0, \ldots, p_{k-1}), \text{ return.}
\end{align*}
\]

- e) If \( i \) is odd:

\[p_i = \text{Next}(p_{i-1}). \]  // note that this is always possible because \( p_{i-1} \notin \text{LAST}^*.\]

Clearly, the above procedure terminates, with a finite sequence \( f(p) = (p_0, \ldots, p_{k-1}) \). This is because in each iteration of the loop, either the procedure ends, or a node is added to the sequence. \( P \) is finite, therefore, eventually a node will be added twice to the sequence, unless we terminate earlier. Whenever this happens the procedure terminates in the next iteration.

We define the following sets which will be useful in the sequel.

\[E_{odd} = \{(p_j, p_j + 1) \in E_a \mid j \text{ is odd} \}\]

\[E_{even} = \{(p_j, p_j + 1) \in E_a \mid j \text{ is even} \}.\]

Note that nodes are added to \( f(p) \) only in steps d) and e). By inspection of the code of these steps we conclude \( E_{odd} \subseteq E_S \) and \( E_{even} \subseteq E_S^e \).

Claim 3.3.1 \( D = \emptyset. \)

Proof: Assume \( p \in D \). Then the classification procedure \( \text{CLASSIFY}(p) \) ends with a sequence \( p = p_0, \ldots, p_{i-1} \) such that \( p_{i-1} \in D_0(S) \) (see Figure 3.2). \( p_0 \) and \( p_{i-1} \) are isolated vertices of \( G_S \) and the other nodes are of degree one. Then \( E_{odd} \subseteq MM \subseteq E_S \). Note also that \( i \) is even. Then \( MM' = MM \setminus E_{odd} \cup E_{even} \) is a matching such that \( |MM'| = |MM| + 1; \) in other words \( p_0, \ldots, p_{i-1} \) is an augmenting path for the maximum matching \( MM \), a contradiction.

\[\square\]
Claim 3.3.2 The sets \( f(p) \) are pairwise disjoint.

Proof:
Assume by contradiction that \( p \neq q \) and \( f(p) \cap f(q) \neq \emptyset \). Let \( r \) be the first element of \( f(p) \) in the intersection. Recall that \( d(r) \in \{0,1,2\} \). We consider the three cases separately:

- \( d(r) = 0 \): \( p,q \in D \) is impossible by Claim 3.3.1. Therefore a node with degree 0 may appear only as the first node. Then, the only degree 0 node in \( f(p) \) is \( p \) and the only degree 0 node in \( f(q) \) is \( q \). Then \( p = r = q \), a contradiction.

- \( d(r) = 1 \): We divide this case into subcases:
  - \( r \) has odd index in one of the sequences (say \( f(p) \)) and even index in the other (say \( f(q) \)). In this case the path \( (p = p_0,p_1,...,r,...,q_1,q_0 = q) \) is an augmenting path for \( MM \), a contradiction. (See figure 3.3).
  - \( r \) has odd indices in both sequences. In this case \( \text{prev}(r) \in f(p) \cap f(q) \), and has an index lower than \( r \) in \( f(p) \), a contradiction.
Figure 3.3: \( d(r) = 1 \), with different indices

- \( r \) has even indices in both sequences. Let \( r' \) be the unique neighbor of \( r \) in \( G_S \). Then \( r' \in f(p) \cap f(q) \) and occurs before \( r \) in \( f(p) \), a contradiction.

- \( d(r) = 2 \): In this case, the procedure ends at step a), for both \( p \) and \( q \). Therefore \( p, q \in A \). Let \( r' \) and \( r'' \) be the neighbors of \( r \) in \( G_S \). We consider three subcases as before:

  - \( r \) has odd index in one of the sequences (say \( f(p) \)) and even index in the other (say \( f(q) \)). One of \( r', r'' \) is in \( f(q) \). Without loss of generality assume \( r' \in f(q) \). Then \( d(r') = 1 \). Therefore \( (r', r, r'') \) is a maximal path in \( G_S \) (see Figure 3.4). By Observation 3.3.1 we may assume \( (r, r') \in MM \) and \( (r, r'') \notin MM \). Then \( q = q_0, q_1, ..., r', r, ..., p_1, p_0 = p \) is an augmenting path for \( MM \), a contradiction.

  - \( r \) has odd indices in both sequences. In this case \( prev(r) \in f(p) \cap f(q) \), and has an index lower than \( r \) in \( f(p) \), a contradiction.

  - \( r \) has even indices in both sequences (see Figure 3.5). Then without loss of generality \( r' \in f(p) \) and \( r'' \in f(q) \). Therefore
Figure 3.4: $d(r) = 2$ with different indices

$d(r') = d(r'') = 1$. Therefore $(r', r, r'')$ is a maximal path in $G_S$. We may assume $(r, r') \in MM$ and $(r, r'') \notin MM$. Then the path $q = q_0q_1...r''$, is an augmenting path, a contradiction.

\[\square\]

**Claim 3.3.3** *If* $p \in C$ *then* $|f(p)| \geq 5$.

**Proof:** Let $p \in C$. Then, the node $p_{i-1}$ is the first node repeated twice in the sequence, namely $\exists j \leq i - 2$ such that $p_j = p_{i-1}$. First, we will prove that $i$ is even: Assume $i$ is odd, then $(p_{i-2}, p_{i-1}) = (p_{i-2}, p_j) \in E_{odd}$ (see Figure 3.6). If $j = 0$ then $d(p) = d(p_0) \geq 1$, a contradiction. If $j > 0$ then $d(p_j) \geq 2$, in this case the procedure would have stopped earlier in step a), and then $p \in A$, a contradiction.

Therefore $|f(p)| = |\{p_0,...,p_{i-2}\}| = i - 1$ is odd. If $j = 0$ then $f(p)$ is a cycle of $G_\alpha$, otherwise $f(p)$ contains a cycle of $G_\alpha$, denoted by $c(p)$. In the latter case $f(p)$ is called a spoon and $h(p) \overset{def}{=} f(p) \setminus c(p)$ is the handle of the spoon (see Figure 3.7). $|f(p)| > 1$, because a self loop in $G_\alpha$ can not be a simple path of $G$. Assume by contradiction that $|f(p)| = 3$. Then $f(p)$ is a
Figure 3.5: $d(r) = 2$ with even indices

Figure 3.6: $i$ cannot be even
cycle \{p, q, r\} (see figure 3.8). In this case \( u \neq v \), for \( rpq \) is in a connected component of \( G_\ast \). Then \( x \) is different from at least one of \( u, v \). Assume w.l.o.g \( x \neq u \). Then \( (p, q) \) could be added to the \( E_\ast \) in the second iteration. Therefore \( d(p) \geq 1 \), a contradiction.

We claim that \( |c(p)| \geq 5 \). Assume, by contradiction that \( c(p) \) is the cycle \{p, q, r\} as in Figure 3.8. Then the edge \((q, r)\) labelled \( x \) is a chord, a contradiction. We conclude \( |f(p)| \geq |c(p)| \geq 5 \).

\[
\square
\]

Now we complete the proof of the lemma:

- For \( p \in A \), \( f(p) \) contains exactly one node \( p' \neq p \) from \( D_2(S) \). Therefore \( |A| \leq |D_2(S)| = d_2(S) \).
- For \( p \in B \), \( f(p) \) contains exactly one node from \( \text{LAST}^* \). Therefore \( |B| \leq |\text{LAST}^*| \).
Figure 3.8: A cycle of length 3

- For \( p \in C \), \( f(p) \) contains at least 5 nodes, therefore \( |C| \leq N/5 \).

\[
D_0(S) = A \uplus B \uplus C \uplus D, \text{ then:}
\]
\[
d_0(S) = |A| + |B| + |C| + |D| \leq d_2(S) + |LAST^*| + N/5
\]
\[
\epsilon(S) = \frac{d_0(S) - d_2(S) - 2|P_S^*|}{N} \leq \frac{d_0(S) - d_2(S) - |LAST^*|}{N} \leq \frac{1}{5}.
\]

Two important corollaries of the preceding lemma are

**Corollary 3.3.1** For any solution \( S \) returned by Algorithm IM
\[
\text{cost}(S) \leq \text{cost}(S^*) + 0.6N
\]

**Proof:** Apply Lemma 3.2.1.

**Corollary 3.3.2** If \( G_0 \) has no cycles, then \( \epsilon(S) \leq 0 \).

**Proof:** Obviously in the proof of the lemma \( C = \emptyset \), because the classification procedure may never construct cycles and spoons.

Therefore in this case \( \text{cost}(S) \leq \text{cost}(S^*) + N/2 \). Note that \( G_0 \) may contain cycles even if the instance does not contain feasible cycles.
3.3.3 A lower bound

**Lemma 3.3.2** There are infinitely many instances \((G, P)\) and solutions \(S\) returned by MM, such that

\[
    \text{cost}(S) = \text{cost}(S^*) + \frac{1}{2} \left( 1 + \frac{1}{5} \right) N = \text{cost}(S^*) + 0.6N
\]

**Proof:** Consider the instance obtained by duplicating \(k\) times the graph in Figure 3.9. The corresponding shareability and conflict graphs are obtained by duplicating \(k\) times the graphs in Figure 3.10. The maximum matching phase may return the maximum matching in Figure 3.11. In this case no edge can be added in phase 2 of the algorithm. It can be checked that the addition of any single edge would violate the conflict graph.

![Figure 3.9: A worst case input](image)

\[
    \text{cost}(S) = 2N - |E_S| = 10k - 2k = 8k
\]

On the other hand \(S^*\) uses 5\(k\) ADMs by partitioning into the cycles \((a, b, c)\) and \((d, e)\). Then

\[
    \text{cost}(S) - \text{cost}(S^*) = 8k - 2k = 3k = \frac{3N}{3} = 0.6N
\]
Figure 3.10: A worst case $G_a$ and conflict graph

Figure 3.11: A worst case solution
One crucial point in this example is that there are two cycles, one of length 2 and one of length 3, in the optimal solution. As stated earlier, if there are no cycles in $G_0$, then $\epsilon(S) \leq 0$. If this is not the case, but there is a high lower bound on number of nodes in a cycle, $\epsilon(S)$ would be close to zero. We proceed to the following section for an improved algorithm exploiting this idea.

### 3.4 Algorithm $PMM(l)$

In this Section we present an algorithm with a preprocessing phase which removes cycles of size at most $l$, where $l$ is an odd number. After the preprocessing phase, we run algorithm $MM$ from the previous section.

Preprocessing:
- Find a maximal set $S_0$ of disjoint valid cycles of length $\leq l$.
- $P_0$ is the set of nodes of the cycles of $S_0$.
- $P_1 \leftarrow P \setminus P_0$. // $S_0$ is maximal, therefore $P_1$ does not contain feasible cycles of length $\leq l$

$MM$:
- Find a solution $S_1$ for the instance $\alpha_1 = (G, P_1)$ using algorithm $MM$.
- $E_S \leftarrow E_{S_0} \cup E_{S_1}$

Note that when $l$ is set to 1, algorithm $PMM(l)$ reduces to algorithm $MM$.

#### 3.4.1 Correctness

$S_0$ consists of disjoint valid cycles. $S_1$ consists of disjoint valid chains because of the correctness of $MM$. Moreover $P_0 \cap P_1 = \emptyset$, therefore $S$ is a set of disjoint valid cycles and chains, i.e. a solution.

#### 3.4.2 Analysis

We begin our analysis with Lemma 3.4.1 which is proven, although in other terminology, in [CW02b]. This will be helpful in understanding the main
result of this section, i.e. the improved upper bound. The proof is based on
the existence of a matching \( M \) having certain size. This matching consists
solely of edges of the connected components of \( G_S \). In our proof we show
that using other edges of \( G_a \) we can build a larger matching which leads to
a higher upper bound. In Subsection 3.4.2 we develop a lower bound on the
number of edges in \( E_S \setminus E_S^* \). In Subsection 3.4.2 we prove a combinatorial
lemma, which helps us to to build our matching. In Subsection 3.4.2 we build
the improved matching and prove our upper bound. In Subsection 3.4.2 we
give a lower bound for the performance of the algorithm.

\textbf{An upper bound}

In the sequel \( S \) is a solution returned by the algorithm and \( S^* \) is an optimal
solution without chords, whose existence is guaranteed by Lemma 3.2.2.

\textbf{Lemma 3.4.1}

\[ e(S) \leq \frac{1}{l+2}. \]

\textbf{Proof:} Let \( P_2 \) be such that \( P_1 \subseteq P_2 \subseteq P \). For \( i \in \{1,2\} \), let \( G^i_a = G_a[P_i] \) be
the subgraph of \( G_a \) induced by \( P_i \). Let \( M_2 \) be a matching of \( G^2_a \) and \( MM \) be
the maximum matching calculated by the algorithm at the first iteration of
phase \textbf{MM}: We first, show that \( d_0(MM) \leq d_0(M_2) + |P_0| \):

Let \( M_1 \) the sub-matching of \( M_2 \) induced by \( P_1 \). Then \( d_0(M_1) \leq d_0(M_2) +
|P_2 \setminus P_1| \), because, the removal of a node from \( P_2 \) may leave at most one
matched node of \( M_2 \) unmatched in \( M_1 \). On the other hand \( MM \) is calculated
by the algorithm as a maximum matching of \( G^1_a \). Therefore

\[ d_0(MM) \leq d_0(M_1) \leq d_0(M_2) + |P_2 \setminus P_1| \leq d_0(M_2) + |P \setminus P_1| = d_0(M_2) + |P_0|. \]

(3.4)

The number of isolated nodes of \( d_0(S) \) of a solution \( S \) returned by \textbf{PMM}(\( l \))
is at most the number of isolated nodes of the maximum matching \( MM \)
found in the first iteration after the preprocessing phase, because no edges
are deleted from \( G_S \) in subsequent iterations. Then

\[ d_0(S) \leq d_0(MM). \]

(3.5)

In the sequel we will construct a matching \( M \) of some subgraph of \( G_S \cup G_S^* \),
induced by some \( P_2 \) chosen as above, having a small number of isolated nodes.
The matching $M$ is obviously a matching of $G_a^2$. By letting $M = M_2$ in (3.4) and combining with (3.5) we get

$$d_0(S) \leq d_0(M) + |P_0|.$$ 

We now give the construction of $M$. We partition the connected components of $G_{S^*}$ as follows.

- $\mathcal{O}_L$ is the set of all odd cycles of $G_{S^*}$ which do not intersect with $P_0$.
  Note that any cycle in this set contains at least $(l + 2)$ nodes.

- $\mathcal{O}_P$ is the set of all odd cycles of $G_{S^*}$ which intersect with $P_0$.

- $\mathcal{E}$ is the set of even cycles of $G_{S^*}$.

- $P_{S^*}$, the set of maximal paths of $G_{S^*}$.

Initially $M$ is the empty matching.

**Phase 1- Cover $\mathcal{E}$:** For every cycle $C_e$ in $\mathcal{E}$, $C_e$ admits a perfect matching. Add this matching to $M$.

**Phase 2- Cover $\mathcal{O}_P$:** For every (odd) cycle $C \in \mathcal{O}_P$. Pick arbitrarily a node $p \in C \cap P_0$. $C \setminus \{p\}$ is an even path, therefore admits a perfect matching. Add this matching to $M$.

**Phase 3- Partly Cover $\mathcal{O}_L$:** For every (odd) cycle $C \in \mathcal{O}_L$, pick a node $p$ arbitrarily. $C \setminus \{p\}$ is an even path, therefore admits a perfect matching. Add this matching to $M$. $p$ remains to be an isolated node of $M$.

**Phase 4- Partly Cover $P_{S^*}$:** Every path $Q \in P_{S^*}$, is either even or odd. In the first case it admits a perfect matching, otherwise we can remove one of its endpoints so that it admits a perfect matching. Add this matching to $M$. This endpoint remains to be an isolated node of $M$.

By the construction we have $d_0(M) \leq |\mathcal{O}_L| + |P_{S^*}|$. Therefore

$$d_0(S) \leq d_0(M) + |P_0| \leq |\mathcal{O}_L| + |P_{S^*}| + |P_0|$$

$$(l + 2)(d_0(S) - d_2(S) - 2 |P_{S^*}|) \leq (l + 2) |\mathcal{O}_L| \leq |P_0|$$

$$\epsilon(S) = \frac{d_0(S) - d_2(S) - 2 |P_{S^*}|}{N} \leq \frac{1}{l + 2}. $$

$\square$
A more careful analysis will show that there is always a matching with bigger cardinality than the matching \( M \) constructed in the above lemma. Our upper bound is based on a construction of such a matching.

We first begin by developing some results which will be used in our proof. The first family of results gives a lower bound on the number of edges "touching" cycles of \( G_S^* \).

**Lower bounds for edges of** \( E_S \setminus E_S^* \).

**Definition 3.4.1** For every \( X \subseteq P \), \( \text{OUT}(X) \) \( \overset{\text{def}}{=} \) \( C(X, \overline{X}) \) is the cut of \( X \) in \( G_S \), namely the set of edges of \( G_S \) having exactly one endpoint in \( X \).

**Lemma 3.4.2** Let \( C \) be a cycle of \( G_S^* \), then

\[
|\text{OUT}(C)| \geq \frac{1}{3}(|C| + |D_0(S) \cap C| - |D_2(S) \cap C|).
\]

**Proof:** Let \( k \) be the number of edges of \( C \) which are not part of \( G_S \) and \( \forall i \in \{0, 1, 2\} , dc_i \overset{\text{def}}{=} |D_i(S) \cap C| \).

The sum of the degrees (in \( G_S \)) of the nodes of \( C \) is

\[
dc_1 + 2dc_2 = |C| - dc_0 + dc_2.
\]

On the other hand each edge of \( G_S \) connecting nodes of \( C \) contributes 2 to this sum and each edge in \( \text{OUT}(C) \) contributes 1. As there are no chords of \( C \), the number of edges contributing 2 is \( |C| - k \). Therefore

\[
|C| - dc_0 + dc_2 = 2(|C| - k) + |\text{OUT}(C)|
\]

\[
|C| + dc_0 - dc_2 = 2k - |\text{OUT}(C)|.
\]

For the following discussion consult Figure 3.12. Consider an edge \( e = (p, q) \) of \( C \) which is not in \( G_S \). This edge was not added to \( E_S \) by the algorithm. This could be only because a node \( p' \) in the connected component of \( p \) in \( G_S \) is conflicting with a node \( q' \) in the connected component of \( q \) in \( G_S \). Either \( p' \notin C \) or \( q' \notin C \), otherwise they would not be conflicting. Assume w.l.o.g. that \( p' \notin C \). Let \( q' \) be the node closest to \( p \) among such nodes. By the choice of \( p \), there is an edge \( e' \) connecting \( p' \) to a node in \( C \). We call \( e' \) the blocking edge of \( e \). Moreover \( e' \in \text{OUT}(C) \). Therefore, any edge \( e \) of \( C \) which is not
in $\mathcal{G}_S$ has a blocking edge, and any edge in $OUT(C)$ may be a blocking edge of at most two edges. Therefore

$$k \leq 2 |OUT(C)|. \quad (3.7)$$

Combining (3.6) and (3.7) we get

$$|C| + dc_0 - dc_2 = 2k - |OUT(C)| \leq 3 |OUT(C)|$$

$$|OUT(C)| \geq \frac{1}{3}(|C| + dc_0 - dc_2).$$

Figure 3.12: Blocking and blocked edges

**Definition 3.4.2** The $i$-neighborhood $N_i(X)$ of $X$ is the set of all the nodes having exactly $i$ neighbors from $X$ in $\mathcal{G}_S$, but are not in $X$. $N(X) \triangleq N_1(X)$.

The following claim is a generalization of the previous lemma to a set of cycles.
Lemma 3.4.3 Let $C$ be a set of cycles of $G_{S*}$. Let $P_C \overset{def}{=} \sqcup C$ be the set of nodes of these cycles. Let $IN(C)$ be the set of edges of $G_S$ connecting two cycles of $C$. Then

$$|N(P_C)| \geq \frac{1}{3} |P_C| + \frac{1}{3} |D_0(S) \cap P_C| - \frac{1}{3} |D_2(S) \cap P_C| - 2 |IN(C)| - 2 |N_2(P_C)|$$

Proof: (Consult Figure 3.13). Consider the sum $\sum_{C \in C} |OUT(C)|$. Each edge in $OUT(P_C)$ is counted in this sum. On the other hand each edge in $IN(C)$ is counted twice (once for each cycle it connects) where it should not be counted at all. Similarly each edge having one endpoint in $N_2(P_C)$ is counted once where it should not be counted at all. The number of these edges is $2 |N_2(P_C)|$.

$$|N(P_C)| = \sum_{C \in C} |OUT(C)| - 2 |IN(C)| - 2 |N_2(P_C)|$$

$$\geq \frac{1}{3} (|P_C| + |D_0(S) \cap P_C| - |D_2(S) \cap P_C|) - 2 |IN(C)| - 2 |N_2(P_C)| .$$

Figure 3.13: Edges of $G_S$ with respect to cycles of $G_{S*}$. 

\[ \Box \]
**Definition 3.4.3** The odd cycles graph $\mathcal{O}G_S = \langle \mathcal{O}C_S, \mathcal{O}E_S \rangle$ of a solution $S$ is a graph in which each node corresponds to an odd cycle of $G_S$, which does not intersect with $P_0$ and two nodes are connected with an edge if and only if there is an edge connecting the corresponding cycles in $E_S$.

**Lemma 3.4.4** Let $X \subseteq \mathcal{O}C_S$. Then

$$|N(P_X)| \geq \frac{1}{3} |P_X| - 2|IN(X)| - 2(d_2(S) - |P_0|).$$

**Proof:** First, we show that $N_2(P_X) \subseteq D_2(S) \setminus P_0 \setminus P_X$. Let $p \in N_2(P_X)$. By definition $p$ has degree 2, namely $p \in D_2(S)$. Still by definition $p \notin P_X$. It remains to show that $p \notin P_0$. By definition $p$ has both of its neighbors in $P_X$. Assume $p \in P_0$, then $p$ is in some cycle of $S_0$. Then both of its neighbors are in this cycle, thus in $P_0$. But they are also in $P_X$, contradicting the fact that by definition, the cycles of $X$ do not intersect with $P_0$. Therefore $|N_2(P_X)| \leq d_2(S) - |P_0| - |D_2(S) \cap P_X|$. Substituting this in 3.4.3 we get

$$|N(P_X)| \geq \frac{1}{3} |P_X| + \frac{1}{3} |D_0(S) \cap P_X| - \frac{1}{3} |D_2(S) \cap P_X| - 2|IN(X)| - 2(d_2(S) - |P_0|)$$

$$\geq \frac{1}{3} |P_X| - 2|IN(X)| - 2(d_2(S) - |P_0|).$$

\[\square\]

**Corollary 3.4.1** Let $I$ be an independent set of $\mathcal{O}G_S$. Then

$$|N(P_I)| \geq \frac{1}{3} |P_I| - 2(d_2(S) - |P_0|).$$

**Proof:** By definition $\forall u, v \in I, (u, v) \notin \mathcal{O}E$. This means that these are not connected by an edge in $E_S$. In other words $IN(I) = \emptyset$.

\[\square\]
Odd Distanced Nodes with Distinct Colors

In this subsection we develop a result which will be an essential tool in building the matching in Subsection 3.4.2 and proving a lower bound on its size. For this purpose we define the "maximum odd distanced nodes with distinct colors" family of problems which are pure combinatorial problems of their own interest.

The cycle version of the problem, \((MODNDC - C)\) is defined as follows:

**Input:** A cycle \(C\) with \(n\) nodes numbered from 1 to \(n\) clockwise, some of which are colored and the rest are not. If a node is colored, \(c(v) \in \mathbb{N}\) denotes its color, otherwise \(c(v) = 0\) and it is termed *uncolored*.

**Output:** A cyclic subsequence \(V = (v_0, v_1, ..., v_{k-1})\) of the nodes of \(C\) such that:

- **Odd distanced:** Between every pair of successive nodes \(v_i, v_{j=i+1 \mod k} \in V\), the clockwise distance \(d(v_i, v_j)\) from \(v_i\) to \(v_j\) is odd. Note that in particular if \(k = 1\) then the \(d(v_0, v_0) = n\) is be odd.

- **Distinct Colors:** Every node \(v_i\) in the sequence is colored (i.e. \(c(v_i) \neq 0\)) and for every pair of distinct nodes \(v_i\) and \(v_j\) in the sequence, \(c(v_i) \neq c(v_j)\).

**Measure:** Our goal is to find \(V\) maximizing the number of nodes of \(C\) which are colored with colors from \(\{c(v_0), c(v_1), ..., c(v_{k-1})\}\). In the sequel it will be easier to measure a solution \(V\) by the number of nodes of \(C\) which are colored with colors from \(\{c(v_0), c(v_1), ..., c(v_{k-1})\}\), plus the number of nodes which are not colored. In other words, given a solution, we first set \(c(v) = 0\) for all \(v\) such that \(c(v) = \{c(v_0), c(v_1), ..., c(v_{k-1})\}\) and we count the number of nodes \(v\) with \(c(v) = 0\). We define as \(B_c(V)\) the set of nodes colored \(c\) after this uncoloring, formally \(B_c(V) \overset{\text{def}}{=} \{v \in C | c(v) = c\}\). \(W(V) \overset{\text{def}}{=} B_0\) is the set of uncolored nodes. \(B(V) \overset{\text{def}}{=} \biguplus_{c > 0} B_c\) is the set of colored nodes. Our target is to find a solution \(V\) such that \(|W(V)|\) is maximized. Obviously \(C = B(V) \uplus W(V)\).

**Definition 3.4.4** A cycle \(C\) is dedicated if it contains nodes colored with one color and possibly some uncolored nodes. Formally, \(|\{c(v) | v \in C\} \setminus \{0\}| = 1\).

**Lemma 3.4.5** Given an instance of the \((MODNDC - C)\) problem, one of the following is true:
• (a) \(C\) is a dedicated even cycle.

• (b) There is a solution \(V\) with measure \(|W(V)| \geq \left[\frac{n}{4}\right]\)

Proof: Let \(V\) be an optimal solution. We consider the following cases:

• **Case 1:** \(V = \emptyset\). It follows from the definition that, if \(V'\) and \(V''\) are two solutions such that \(V' \subset V''\), then \(W(V') \subset W(V'')\), thus \(|W(V')| < |W(V'')|\). In particular, for any solution \(V' \neq \emptyset\), \(|W(V')| > |W(\emptyset)|\). As we assumed \(V = \emptyset\), it follows that no other solution is feasible.

If all the nodes are uncolored then \(|W(\emptyset)| = n\), thus (b) holds. Otherwise there are some colored nodes. If \(n\) is odd, then any singleton of the colored nodes is a non-empty solution, a contradiction. Therefore \(n\) is even. If there is only one color, then this is a dedicated even cycle and (a) holds. Otherwise there are at least two colors. Since \(V = \emptyset\) no pair of nodes is a solution. Then, for any pair \(u, v\) of nodes, either they are an even distance apart, or \(c(u) = c(v)\). Fix some node \(v\) and let \(c(v) = a >\). Then all the nodes \(u\) such that \(c(u) = a\) are at even distance from \(v\). We claim that all the nodes \(u'\) such that \(c(u') = a\) are also at even distance from \(v\). Assume that there is a node \(u'\) such that \(c(u') = a\) at odd distance from \(v\), then it is at odd distance from the nodes \(u\) such that \(c(u) \neq a\). Then \(u'\) together with one of the \(u\) nodes is a solution, contradiction our assumption. Then all the colored nodes are at even distance from \(u\). We conclude that all the nodes at odd distance from \(u\) are uncolored. Then \(|W(\emptyset)| \geq \frac{n}{2}\).

• **Case 2:** \(V \neq \emptyset\). We want to show that \(|W(V)| \geq \frac{n}{3} = \frac{|W|}{3} + \frac{|B|}{3}\) which is equivalent to \(|B| \leq 2|W|\). For this purpose we will partition the set \(B\) into two disjoint sets \(X, Y\), and then prove \(|X| \leq |W|\) and \(|Y| \leq |W|\).

Let \(V = \{v_0, v_1, \ldots, v_{k-1}\}\). Consider two consecutive nodes \(v_i, v_j \in V\). Note that \(i = j\) if \(k = 1\), thus these nodes need not be distinct. Recall also that the clockwise distance \(d(v_i, v_j)\) from \(v_i\) to \(v_j\) is odd.

Observe that if there are two colored nodes \(x, y \in B(V)\) between these two nodes such that \(x\) is closer to \(v_i\) and that \(d(v_i, x)\) and \(d(x, y)\) are odd, then \(c(x) = c(y)\). For, otherwise the set \(V \cup \{x, y\}\) is a better solution than \(V\), a contradiction.

68
Case 1: y occurs before x

Case 2: y occurs after x

Figure 3.14: The nodes between two nodes of the solution

We use this observation to characterize the colored nodes of the solution, i.e., the nodes of $B(V)$. For the following discussion consult Figure 3.14. Let $x \in B(V)$ be the colored node which is closest to $v_i$ when going clockwise from $v_i$ to $v_j$ and is at odd distance from $v_i$. Let $y \in B(V)$ be the colored node which is farthest from $v_i$ when going from $v_i$ to $v_j$ and is at even distance from $v_i$. Note that $y$ is the first node in $B(V)$ at odd distance from $v_j$ when going counterclockwise from $v_j$ to $v_i$. By these choices, all the colored nodes before $x$ are at even distance from $v_i$ and all the colored nodes after $y$ are at odd distance from $v_i$. If $y$ occurs before $x$ then there are no colored nodes between $x$ and $y$, or in other words, all the colored nodes are either before $y$ or after $x$. Note that this statement holds even if one or both of $x, y$ do not exist. In all these cases we define $X_i = \emptyset$. If $y$ occurs after $x$ then by the observation in the previous paragraph $c(x) = c(y) = c$. Furthermore, by the same observation, for every colored node $z$ between $x$ and $y$, $c(z) = c$. In this case we define $X_i$ be the set of all the colored nodes from $x$ to $y$ including $x$ and $y$. Let also $Y_i$ be the set of all other colored nodes between $v_i$ and $v_j$. Let $X \overset{def}{=} \cup_{i=0}^{k-1} X_i$ and $Y \overset{def}{=} \cup_{i=0}^{k-1} Y_i$.

Obviously $|Y| \leq |W|$, for the nodes of $Y$ are separated by at least one
node in $W$.

Let $V'_i \subseteq W$ be the set of nodes having originally the same color as $v_i$. Note that $X_i$ has at least one node $x$ which is at even distance from $v_i$. Therefore $V' = V \setminus \{v_i\} \cup \{x\}$ is a solution. If $|X_i| > |V_i|$ then $|W(V')| > |W(V)|$, a contradiction, hence $|X_i| \leq |V_i|$. Summing up from $i = 0$ to $k - 1$ we have $|X| \leq \sum_{i=0}^{k-1} |V_i| \leq |W|$.

We conclude that $|B(V)| = |X| + |Y| \leq 2|W(V)|$ as required.

\[ \square \]

The path version of the problem $(MODNDC - P)$ is defined similarly.

**Input:** A path $P$ with $n$ nodes numbered from 1 to $n$ some of which are colored and the rest are not. If a node is colored, $c(v) \in \mathbb{N}$ denotes its color, otherwise $c(v) = 0$ and it is termed *uncolored*.

**Output:** A subsequence $(v_0, v_1, ..., v_{k-1})$ of the nodes of $P$ such that:

- **Odd distanced:** For every pair of successive nodes $v_i$ and $v_{i+1}$ in the sequence, the distance from $v_i$ to $v_{i+1}$ is odd.

- **Distinct Colors:** Every node $v_i$ in the sequence is colored (i.e. $c(v_i) \neq 0$) and for every pair of distinct nodes $v_i$ and $v_j$ in the sequence, $c(v_i) \neq c(v_j)$.

**Measure:** $|W(V)|$ as defined for the cycle version of the problem.

**Corollary 3.4.2** Given an instance of the $(MODNDC - P)$ problem, there is always a solution $V$ with measure $|W(V)| \geq \lceil \frac{n}{3} \rceil$

**Proof:** If all the nodes are uncolored, then the empty set is a solution with measure $n$. If all the colored nodes have the same color, then any one of these nodes constitutes a solution with measure $n$. Otherwise, there are at least two nodes colored with two different colors. In this case we construct a cycle $C$ by connecting the endpoints of $P$ by an edge. Now we have an instance of the $MODNDC - C$ problem with $n$ nodes and $C$ is not a dedicated cycle. By the previous lemma, there is a solution of this instance with measure at least $\lceil \frac{n}{3} \rceil$. This solution satisfies the conditions of the $MODNDC - P$ problem too.

\[ \square \]
A better upper bound

Our main result is the following:

Lemma 3.4.6

\[ \epsilon(S) \leq \frac{1}{3/2(l + 2)}. \]

Proof: The outline of the proof is as in Lemma 3.4.1, using a different (and larger) matching \( M \). We keep the same notations and definitions of Lemma 3.4.1.

We partition the connected components of \( \mathcal{G}_{S^*} \) as follows.

- \( \mathcal{I} \) is some maximum independent set of \( \mathcal{OG}_S \).
- \( \mathcal{D} = \mathcal{OC}_S \setminus \mathcal{I} \).
- \( \mathcal{O} \) is the set of all odd cycles of \( \mathcal{G}_{S^*} \) except those in \( \mathcal{OC}_S \), in other words all the odd cycles of \( \mathcal{G}_{S^*} \) which intersect with \( P_0 \).
- \( \mathcal{E} \) is the set of even cycles of \( \mathcal{G}_{S^*} \).
- \( P_{S^*} \), the set of maximal paths of \( \mathcal{G}_{S^*} \).

Note that each cycle in \( \mathcal{OC}_S = \mathcal{I} \cup \mathcal{D} \) has at least \( l + 2 \) nodes, because it is odd and it does not intersect with \( P_0 \).

We further partition these sets as follows.

- \( \mathcal{I} = \mathcal{I}_1 \cup \mathcal{I}_2 \cup \mathcal{I}_D \)
- \( \mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \)
- \( \mathcal{O} = \mathcal{O}_1 \cup \mathcal{O}_2 \)
- \( \mathcal{E} = \mathcal{E}_D \cup \mathcal{E}_2 \)

Initially \( \mathcal{I}_D = \mathcal{I}_2 = \mathcal{D}_2 = \mathcal{O}_2 = \mathcal{E}_2 = \emptyset \), thus \( \mathcal{I}_1 = \mathcal{I}, \mathcal{D}_1 = \mathcal{D}, \mathcal{O}_1 = \mathcal{O}, \mathcal{E}_D = \mathcal{E} \), and \( M \) is the empty matching.

Phase 1- Coloring: Choose a distinct color \( c_i \) for each (odd) cycle \( C_i \) of \( \mathcal{I} \). We try to color the neighbors of each cycle \( C_i \) with color \( c_i \). If there is a conflict in the coloring of a node, we do not color it. Formally, we color all the nodes of \( N(C_i) \cap N(P_T) \) with color \( c_i \).

At this point the following two invariants are obviously true.
• **INV1:** All the nodes in the cycles of $\mathcal{E}_2 \cup \mathcal{D}_2 \cup \mathcal{I}_2$ are covered by $M$.

• **INV2:** There is a one to one correspondence between the set of colors and the set of cycles in $\mathcal{I}_i \cup \mathcal{I}_D$.

Invariants INV1 and INV2 and other invariants that will be in the rest of the construction will hold at the end of each phase of the construction, and in particular at the end of the construction.

**Phase 2- Uncoloring by MODNDC-C of even cycles:** As long as there is an even cycle $C$ in $\mathcal{E}_D$, admitting a solution with measure at least $\left\lceil \frac{|C|}{3} \right\rceil$ to the MODNDC-C problem, do the following processing which is described in Figure 3.15:

Pick an optimal solution of the MODNDC problem for $C$ with the current colors. Let $x_1, x_2, ..., x_k$ be the nodes of the solution. Note that $k$ is even, since the sum of $k$ odd distances is even. Let $y_i$ be the neighbor node $x_i$ which gave it its color in Phase 1. As the colors of each $x_i$ are distinct, the nodes $y_i$ belong to distinct odd cycles $C_i \in \mathcal{I}$. Let $p_i$ be the path on $C$ from $x_i$ to $x_{i+1}$ excluding $x_i$ and $x_{i+1}$. This is a path of odd length, therefore it has an even number of nodes. As such, these paths admit a perfect matching. The induced subgraph resulting from the removal of $y_i$ from $C_i$ is a path with an even number of nodes, and admits a perfect matching too. Add these matchings and the edges $\{i \leq k (x_i, y_i)\}$ to $M$. Now $M$ covers perfectly the cycles $C, C_1, C_2, ..., C_k$. In particular if $k = 0$ then $M$ covers perfectly $C$. Uncolor all the nodes with colors $c(x_1), c(x_2), ..., c(x_k)$ in $\mathcal{G}_a$, then

\[
\begin{align*}
\mathcal{I}_1 &\leftarrow \mathcal{I}_1 \setminus \{C_1, C_2, ..., C_k\} \\
\mathcal{I}_2 &\leftarrow \mathcal{I}_2 \cup \{C_1, C_2, ..., C_k\} \\
\mathcal{E}_D &\leftarrow \mathcal{E}_D \setminus \{C\} \\
\mathcal{E}_2 &\leftarrow \mathcal{E}_2 \cup \{C\}
\end{align*}
\]

Note that invariants INV1 and INV2 hold. Additionally the following is true:

$\mathcal{E}_D$ may contain only dedicated even cycles $C$ with at least $\frac{2|C|}{3} \geq \left\lceil \frac{|C|}{2} \right\rceil$ colored nodes.

**Phase 3- Uncoloring by preprocessed dedicated even cycles:** For every even cycle $C_e$ such that $C_e \cap P_0 \neq \emptyset$ do the following processing which is described in Figure 3.16:
Figure 3.15: Matching by MODNDC-C of even cycles
Figure 3.16: Matching by using preprocessed dedicated even cycles

Pick arbitrarily a node $p \in C_e \cap P_0$. There are at least $\left\lceil \frac{|C_e|}{2} \right\rceil$ colored nodes in $C_e$, therefore there is at least one colored node $x$ at odd distance from $p$. This node has a neighbor $y$ in a cycle $C_o \in \mathcal{I}_1$. $C_o \setminus \{y\}$ is an even path. $C_e \setminus \{p, x\}$ consists of two even paths. They admit perfect matchings. Add these matchings and $\{x, y\}$ to $M$. Uncolor all the nodes with colors $c(x)$ in $\mathcal{G}_0$, then

$$\mathcal{I}_1 \leftarrow \mathcal{I}_1 \setminus \{C_o\}$$
$$\mathcal{I}_2 \leftarrow \mathcal{I}_2 \cup \{C_o\}$$
$$\mathcal{E}_D \leftarrow \mathcal{E}_D \setminus \{C_e\}$$
$$\mathcal{E}_2 \leftarrow \mathcal{E}_2 \cup \{C_e\}.$$ 

Note that at this point invariants $INV1, INV2$ hold, and the following invariant also holds.

- **INV3**: $\mathcal{E}_D$ contains only dedicated even cycles which do not intersect with $P_0$.

This invariant will hold until the end of the construction, for the simple reason that we will never modify neither $\mathcal{E}_D$ nor a coloring of a cycle in it.

**Phase 4- Uncoloring by MODNDC of odd cycles:** For every odd cycle $C \in \mathcal{D}_1 \cup \mathcal{O}$ we do the following:

Pick an optimal solution of the MODNDC problem for $C$ with the current colors. Let $x_1, x_2, \ldots, x_k$ be the nodes of the solution. Note that $k$ is either
zero or odd, since the sum of \( k \) odd distances is odd. If \( k > 0 \) build a perfect matching as in Phase 2 and then:

If \( C \in \mathcal{D}_1 \) then

\[
\mathcal{D}_1 \leftarrow \mathcal{D}_1 \setminus \{C\}
\]
\[
\mathcal{D}_2 \leftarrow \mathcal{D}_2 \cup \{C\}
\]

otherwise

\[
\mathcal{O}_1 \leftarrow \mathcal{O}_1 \setminus \{C\}
\]
\[
\mathcal{O}_2 \leftarrow \mathcal{O}_2 \cup \{C\}.
\]

Note that invariants \( INV1, INV2 \) and \( INV3 \) hold.

**Phase 5 - Match odd cycles in \( \mathcal{D}_1 \):** Find a maximum matching of \( \mathcal{D}_1 \) (consult Figure 3.17 for this phase). For each pair of cycles \( C, C' \) in this matching do the following:

Pick arbitrarily an edge joining these two cycles in \( \mathcal{G}_S \), add it to \( M \). The remaining parts of \( C \) and \( C' \) are paths with an even number of nodes each of which admits a perfect matching. Add these perfect matchings to \( M \).

\[
\mathcal{D}_1 \leftarrow \mathcal{D}_1 \setminus \{C, C'\}
\]
\[
\mathcal{D}_2 \leftarrow \mathcal{D}_2 \cup \{C, C'\}.
\]

Note that invariants \( INV1, INV2 \) and \( INV3 \) hold.

**Phase 6 - Uncoloring by MODNDC-P:** For every path \( Q \) in \( P_{S^*} \), do the following processing which is depicted in Figure 3.18:

Pick an optimal solution of the MODNDC problem for \( Q \) with the current colors. Let \( x_1, x_2, ..., x_k \) be the nodes of the solution. As the colors of the \( x_i \)
are distinct, the neighbor nodes $y_k$ of $x_i$ which gave the $x_i$ their colors in Phase 1 belong to distinct odd cycles $C_i$. Let $p_i$ be the path on $C$ from $x_i$ to $x_{i+1}$ excluding $x_i$ and $x_{i+1}$. This is a path with odd length and admits a perfect matching. The induced subgraph resulting from the removal of $y_k$ from $C_i$ is a path with odd length, and also admits a perfect matching. Add these matchings and the edges $\{i \leq k (x_i, y_i)\}$ to $M$. Now $M$ perfectly covers the cycles $C_1, C_2, ..., C_k$. Uncolor all the nodes with colors $c(x_1), c(x_2), ..., c(x_k)$ in $G_2$, then

\[ \mathcal{I}_1 \leftarrow \mathcal{I}_1 \setminus \{C_1, C_2, ..., C_k\} \]
\[ \mathcal{I}_2 \leftarrow \mathcal{I}_2 \cup \{C_1, C_2, ..., C_k\} . \]

The remaining paths at both ends of $Q$ may or may not admit a perfect matching. We add a maximum matching of each of them to $M$. We remain with at most two uncovered nodes of $Q$.

Note that invariants INV1, INV2 and INV3 hold.

**Phase 7- Cover $\mathcal{E}_D$:** Recall that by invariant INV3 $\mathcal{E}_D$ contains only dedicated even cycles. For every cycle $C_e$ in $\mathcal{E}_D$ find the odd cycle $C_o \in \mathcal{I}$ corresponding to the unique color of its colored nodes. $C_e$ admits a perfect matching. Add this matching to $M$ and then

76
\[ I_1 \leftarrow I_1 \setminus \{ C_o \} \]
\[ I_D \leftarrow I_D \cup \{ C_o \}. \]

At this point in addition to INV1, INV2 and INV3 the following invariant holds.

- **INV4**: $M$ covers the nodes of the cycles in $E_D$.

**Phase 8- Cover $O_1$**: Recall that $O_1$ contains odd cycles which intersect with $P_0$. For every (odd) cycle $C \in I_o$. Pick arbitrarily a node $p \in C \cap P_0$. $C \setminus \{ p \}$ is an even path, therefore admits a perfect matching. Add this matching to $M$.

**Phase 9- Partly Cover $I_1 \cup I_D \cup D_1$**: For every (odd) cycle $C \in I_1 \cup I_D \cup D_1$, pick a node arbitrarily. The remaining nodes of $C$ form a path with an even number of nodes, and admit a perfect matching. Add this matching to $M$.

At this point the construction of $M$ is completed. The invariants INV1, INV2, INV3 and INV4 hold. In the sequel we will calculate an upper bound for $d_0(S)$.

By the construction we have $d_0(M) \leq |I_1| + |I_D| + |D_1| + 2|P_{S^*}|$. By the discussion in the beginning of the proof of Lemma 3.4.1, $d_0(S) \leq d_0(M) + |P_0|$. Therefore

\[
d_0(S) \leq |I_1| + |I_D| + |D_1| + 2|P_{S^*}| + |P_0| \\
(d_0(S) - d_2(S) - 2|P_{S^*}|) \leq |I_1| + |I_D| + |D_1| - (d_2(S) - |P_0|). \tag{3.8}
\]

Each dedicated cycle in $E_D$ has its nodes colored with one color. Then the number of colors used in all the cycles of $E_D$ is at most $|E_D|$. These colors have a one-to-one correspondence with the cycles of $I_D$. Therefore

\[
|I_D| \leq |E_D|. \tag{3.9}
\]

**Claim 3.4.1**

\[
|D_1| \leq |I_2|. \tag{3.10}
\]
\textbf{Proof:} Consider a cycle \( C \in \mathcal{D}_1 \). This means that \( C \) could not be moved to \( \mathcal{D}_2 \) neither in phase 4, nor in phase 5. Therefore in phase 4 when \( C \) was considered, MODNDC-C returned \( k = 0 \). This implies that all the nodes of \( C \) were uncolored by that time, since otherwise any colored node would constitute a solution of MODNDC-C with \( k = 1 \). This means that all the neighbors of \( C \) in \( \mathcal{I} \) were uncolored by that time. As our construction does not color any nodes after Phase 1, these neighbors are uncolored at the end of the construction, therefore no neighbor of \( C \) is in \( \mathcal{I}_1 \cup \mathcal{I}_D \). On the other hand \( C \) could not be moved to \( \mathcal{D}_2 \) in Phase 5, therefore no neighbor of \( C \) is in \( \mathcal{D}_1 \). Then \( \mathcal{I}_1 \cup \mathcal{I}_D \cup \mathcal{D}_1 \) is an independent set. Assume by contradiction that \( |\mathcal{D}_1| > |\mathcal{I}_2| \). Then \( \mathcal{I}' = \mathcal{I} \setminus \mathcal{I}_2 \cup \mathcal{D}_1 = \mathcal{I}_1 \cup \mathcal{I}_D \cup \mathcal{D}_1 \) is an independent set with \( |\mathcal{I}'| > |\mathcal{I}| \), contradicting the fact that \( \mathcal{I} \) is a maximum independent set. This is the only place we need the maximality of the independent set \( \mathcal{I} \) in our proof.

\[ \square \]

We combine (3.9) and (3.10), multiply both sides by \( \frac{1}{2}(l + 2) \) and get

\[
\frac{1}{2}(l + 2)(|\mathcal{D}_1| + |\mathcal{I}_D|) \leq \frac{1}{2}(l + 2)|\mathcal{E}_D| + \frac{1}{2}(l + 2)|\mathcal{I}_2| \\
\leq (l + 1)|\mathcal{E}_D| + (l + 2)|\mathcal{I}_2| \\
\leq |P_{\mathcal{E}_D}| + |P_{\mathcal{I}_2}| \tag{3.11}
\]

where the last inequality is true by invariant INV 3. We also have

\[
(l + 2)(|\mathcal{D}_1| + |\mathcal{I}_1| + |\mathcal{I}_D|) \leq |P_{\mathcal{D}_1}| + |P_{\mathcal{I}_1}| + |P_{\mathcal{I}_D}|. \tag{3.12}
\]

For a component (cycle or chain) \( C_i \) of \( \mathcal{G}_{S^*} \), let \( \text{col}_i \) be the number of the colored nodes in it, and let \( \text{uncol}_i \) be the number of uncolored nodes in it. The nodes of \( N(P_{\mathcal{I}_1}) \) are all colored and they are in \( P_{\mathcal{D}_2} \cup P_{\mathcal{O}_2} \cup P_{\mathcal{E}_2} \cup P_{S^*} \), therefore

\[
|N(P_{\mathcal{I}_1})| \leq \sum_{C_i \in \mathcal{D}_2 \cup \mathcal{O}_2 \cup \mathcal{E}_2 \cup \mathcal{P}_{S^*}} \text{col}_i \tag{3.13}
\]

As there are no dedicated even cycles in \( \mathcal{D}_3 \cup \mathcal{O}_2 \cup \mathcal{E}_2 \cup \mathcal{P}_{S^*} \), by the results on the MODNDC problems for each component \( C_i \) we have \( \text{col}_i \leq 2 \cdot \text{uncol}_i \). Then

\[
\frac{1}{2}|N(P_{\mathcal{I}_1})| \leq \sum_{C_i \in \mathcal{D}_2 \cup \mathcal{O}_2 \cup \mathcal{E}_2 \cup \mathcal{P}_{S^*}} \text{uncol}_i \tag{3.14}
\]

78
Combining (3.13) and (3.14) and substituting \( |C_i| = \text{col}_i + \text{uncol}_i \) we obtain
\[
\frac{3}{2} |N(P_{Z_1})| \leq |P_{D_2}| + |P_{O_2}| + |P_{E_2}| + |P_{P_{S^*}}|.
\]
By Corollary 3.4.1 we have
\[
|N(P_{Z_1})| \geq \frac{1}{3} |P_{Z_1}| - 2(d_2(S) - |P_0|) \geq \frac{1}{3}(l + 2) |I_1| - 2(d_2(S) - |P_0|)
\]
We combine to obtain
\[
\frac{1}{2}(l + 2) |I_1| - 3(d_2(S) - |P_0|) \leq |P_{D_2}| + |P_{O_2}| + |P_{E_2}| + |P_{P_{S^*}}| \tag{3.15}
\]
Now by summing up (3.11), (3.12) and (3.15) we obtain
\[
\frac{3}{2}(l + 2)(|I_1| + |D_1| + |D_2|) - 3(d_2(S) - |P_0|) \leq N \leq \frac{N}{\frac{3}{2}(l + 2)} \tag{3.16}
\]
By (3.8) and (3.16) we get
\[
d_0(S) - d_2(S) - 2 |P_{S^*}| \leq \frac{N}{\frac{3}{2}(l + 2)}
\]
or
\[
\epsilon(S) = \frac{d_0(S) - d_2(S) - 2 |P_{S^*}|}{N} \leq \frac{1}{\frac{3}{2}(l + 2)}
\]
which completes the proof.

\[\Box\]

A lower bound

**Lemma 3.4.7** There are infinitely many instances \((G,P)\) and solutions \(S\) returned by PMM(l), such that
\[
\epsilon(S) = \frac{1}{2l + 3}.
\]
**Proof:** Consider the graph $H$ depicted in Figure 3.19. $H$ contains a cycle $H_1$ of length $l + 1$ and one cycle $H_2$ of length $l + 2$. For each $k$ consider an instance $\alpha$ such that $G_\alpha$ consists of $k$ copies of $H$ and the conflict graph (not shown in the figure) contains all the possible edges except the edges of $H$ and the chords of the cycles $H_1$ and $H_2$. Note that for $l = 1$ we get the instance in the proof of Lemma 3.3.2 as a special case.

$G_{S^*}$ consists of the $k$ copies of $H_1$ and $H_2$.

Any cycle $C$ of $H$ with $l$ nodes or less has at least four nodes, two from each of $H_1$ and $H_2$. At least two pairs of these nodes will be in conflict. Thus, there are no feasible cycles of length up to $l$. It follows that the algorithm will not make any changes during the preprocessing phase. The matching consisting of the $k(l + 1)$ edges between the $k$ copies of the cycles $H_1$ and $H_2$ is a maximum matching. If the algorithm finds this maximum matching in the first iteration, it will not be able to extend it in any manner in the next phase and the algorithm will terminate $G_{S^*}$ being this maximum matching. We therefore have

$$d_0(S) = k, d_2(S) = 0, |P_{S^*}| = 0, N = k(2l + 3)$$

and

$$\epsilon(S) = \frac{d_0(S) - d_2(S) - 2|P_{S^*}|}{N} \cdot \frac{1}{2l + 3}.$$  

From Lemma 3.4.6 and Lemma 3.4.7 we get the following theorem as a corollary.
**Theorem 3.4.1** For any solution $S$ returned by algorithm $PMM(l)$, $\epsilon(S) \leq \frac{1}{2^{l+2}}$ and there are infinitely many instances for which $\epsilon(S) \geq \frac{1}{2^{l+3}}$.

Substituting $l = 1$ on the above result we obtain

**Corollary 3.4.3** For any solution $S$ returned by algorithm $MM$, $\epsilon(S) \leq \frac{2}{5}$ and there are infinitely many instances for which $\epsilon(S) \geq \frac{1}{5}$.

In fact in Section 3.3 we have proven the following tight result for this special case. For any solution $S$ returned by algorithm $MM$, $\epsilon(S) \leq \frac{1}{5}$ and there are instances for which this value is achieved.

### 3.5 Conclusion and Possible Improvements

We presented an improved analysis for the algorithm in [CFW02] for a network of a general topology.

We analyzed the algorithm $MM$ without the preprocessing phase and proved that $MM = OPT + \frac{1}{2}(1 + \epsilon)N$, where $\epsilon = \frac{1}{5}$; we then discussed the algorithm $PMM$ with preprocessing and proved $PMM = OPT + \frac{1}{2}(1 + \epsilon)N$, where $\frac{1}{2^{l+3}} \leq \epsilon \leq \frac{1}{2^{l+2}}$. For any given $\epsilon > 0$ this improves the analysis of the time complexity of the algorithm. In addition we use a novel technique in our analysis.

Open problems that are directly related to our work are (1) to further close the gap between the upper and lower bound, and (2) to extend to use of our technique to related problems. As we measure the performance of any algorithm $ALG$ by $ALG \leq OPT + cN$ for some $0 < c < 1$, two other open problems are (3) to find an upper bound smaller than $1/2c$, and (4) to determine whether there exists a positive lower bound for $c$.

An interesting fact is that the tight bound of $OPT + \frac{1}{2}(1 + 1/5)N = OPT + 0.6N$ proven in our work for algorithm $MM$ matches the performances of the algorithm in [EMZ02] and algorithm $MCC - WS$ in [CFW02]. The question of whether or not this is a lower bound for algorithms without preprocessing is an open problem.

The MODNDC family of problems are of their own interest. We do not know whether the measure bound of $n/3$ given in Lemma 3.4.5 is tight. In all instances we were able to construct, there was a solution with measure $|W| \geq \lfloor \frac{n}{2} \rfloor$. To close the gap between $n/3$ and $n/2$ is an open problem. If
one can guarantee a solution with measure $n/2$ for the MODNDC-C problem, this will immediately imply $\epsilon \leq \frac{1}{\frac{1}{2}(l+2)}$. 
Chapter 4

Uniform All-to-all traffic in SONET Rings

4.1 Introduction

4.1.1 Background and Previous Work

The ADM minimization problem for ring networks and general networks are described in the previous chapters. An important generalization of the ADM minimization problem is the traffic grooming problem which is defined in [GRS98] and received much attention in recent works (e.g. [WCLF00, CM00, BC03]). [ZM03] is an excellent review of the various variants of the problem.

In studying the hardware cost, or more specifically the ADM minimization problem, the issue of grooming became central. This problem stems from the fact that the network usually supports traffic that is at rates lower than the full wavelength capacity, and therefore the network operator has to be able to put together (= groom) low-capacity demands into the high capacity fibers.

In this problem the basic traffic unit is 1/g of the capacity of one wavelength, where g is the grooming factor. The input is a set of traffic requests between pairs of nodes having sizes which are multiples of this basic traffic unit. In this variant a coloring is valid if for every edge e and color λ, the number of paths using e and colored λ is at most g. A set of (at most 2g) paths colored with the same color sharing a common endpoint uses one ADM. Note that the special case of g = 1 is the ADM minimization problem.
4.1.2 Our Contribution

As an attempt to investigate the relationship between the number of wavelengths and the number of ADMs, in this work we concentrate on the uniform all-to-all traffic pattern and require maximum utilization of the bandwidth of $W$ wavelengths. Furthermore we assume the multi-hop communication model and splitable requests, and investigate the problem of minimizing the number of ADMs used under these conditions.

We propose an architecture of successive nested polygons and give a necessary and sufficient condition for a solution in this architecture to be feasible. Using this condition we give an optimal solution for $W = 2$ and a solution using $O(W \log W + N)$ ADMs for the general case, where $W$ is $o(W)$ and $N$ is the size of the ring. In other words, if $W < N/\log N$ the cost depends asymptotically only on $N$, and if $W > N^{1+\epsilon}/\log N$ for any $\epsilon > 0$, then the cost depends only on $W$. Our technique is extendable to sequences of polygons which are not necessarily nested.

In Section 4.2 we give a formal definition of the problem, investigate the basic properties of the solution. In Section 4.3 we introduce our demand function which constitutes an essential tool for our analysis of the result. In Section 4.4 we present the architecture and analyze its performance. In Section 4.5 we generalize the results and discuss application to the uniform all-to-all multi-hop Traffic Grooming problem. In Section 4.6 we summarize the results and suggest further research directions.

4.2 Problem Definition and Preliminary Results

4.2.1 Problem Definition

Consider $W$ bidirectional SONET rings with (the same) $N$ nodes $\{0, 1, ..., N - 1\}$, each operating on a separate wavelength and one ADM for each wavelength at each node. Each ring consists of $N$ lightpaths and traffic can be switched between the rings at each node. This architecture is called PPWDM ([GLS99]).

Consider also the uniform all-to-all traffic where the traffic from node $i$
to node $j$ is

$$\mathcal{T}(i, j) = \begin{cases} 
0 & \text{if } i = j \\
\frac{\tau}{2} & \text{otherwise.}
\end{cases}$$

As we will be interested mostly in asymptotic results, and the differences in the results between odd and even values of $N$ are small, we will assume for simplicity that $N$ is even.

Consider the shortest path routing of the above traffic $\mathcal{T}$, where traffic from node $i$ to node $(i + N/2) \mod N$ is split and routed equally on both directions. Figure 4.1 shows the routing of the demands $\mathcal{T}(0, j)$. The load induced on the system by any node in any direction is

$$\sum_{j=1}^{N/2-1} \tau j + \frac{\tau N}{2} = \left(\frac{(N/2 - 1) N/2}{2} + \frac{N}{4}\right) \tau = \frac{N^2}{8} \tau$$

To obtain the total load induced by all nodes we multiply the above by the number of nodes ($N$). Since it is clear that the load is the same on each directed edge, we conclude that in this specific routing of the traffic demand $\mathcal{T}$, the load on every directed edge is $\frac{N^2}{8} \tau$.

Figure 4.1: Routing of traffic from node 0

Clearly the above total load is the minimum possible because of the shortest path routing. Moreover, this total load is distributed evenly on all the
edges. Therefore in any routing there is at least one edge with this load or more, in other words this is the minimum possible maximum edge load.

Assuming that the unit of traffic is the capacity of one wavelength, the capacity of each edge is \( W \). The maximum all to all uniform traffic that can be routed on the above PPWDM ring satisfies: \( \frac{N^2}{8}\tau = W \), or \( \tau = \frac{8W}{N^2} \). We define \( n \equiv \frac{N}{2} \), therefore:

\[
\tau = \frac{2W}{n^2}.
\]

As traffic can be switched freely between the rings at each node and the capacity of each edge is equal to its load, this traffic can be routed on a PPWDM ring which uses \( N \) ADMs.

Our goal is to find an architecture that uses the same number \( W \) of wavelengths, supports the same traffic demand \( (T) \) and uses smallest possible number of ADMs.

**Proposition 4.2.1** Every solution should use a shortest path routing.

Otherwise the total load will increase and the average load will be greater than \( \frac{N^2}{2}\tau \) (= \( W \)). Therefore, there will be at least one edge with load greater than its capacity.

For this reason and the fact that each edge has the same capacity in each direction, the problem can be separated into two identical "directed" instances, one for each direction. We will deal with the "clockwise" problem, in which there are \( N \) directed edges \((i, i + 1)\) and the traffic from node \( i \) to node \( j \) is positive only when the shortest path from \( i \) to \( j \) is "clockwise".

An architecture is defined by its lightpaths and the routing of the traffic over these lightpaths.

- **Lightpaths:**

**Definition 4.2.1** A lightpath is a dipath \( p \) of the cycle and a wavelength (color) \( w(p) \in \mathbb{N} \) assigned to it.

**Definition 4.2.2** A coloring \( w \) is valid if any two lightpaths \( p \) and \( p' \) such that \( w(p) = w(p') \) have no edges in common.
Definition 4.2.3 A Lightpath Graph is a directed multigraph with $N$ nodes, and an edge $e = (i, j)$ for each lightpath from node $i$ to node $j$. For such an edge $l(e) \overset{\text{def}}{=} (j - i) \mod N$ and $w(e)$ is the color assigned to the lightpath it represents.

The number of ADMs used at each node $v$ of the lightpath graph is the number of colors "touching" $v$, namely $|\{w(e) | e \text{ is adjacent to } v\}|$. The number of ADMs used by a lightpath graph is the sum of the number of ADMs used at each node.

- **Routing:** The routing problem is the following multi-commodity flow problem:

  - **Input:**
    * A Lightpath Graph and capacities $c(e) = 1$ for all edges.
    * A demand matrix:

      $$D(i, j) = \begin{cases} 
      \tau & \text{if } 0 < (j - i) \mod N < N/2 \\
      \tau/2 & \text{if } (j - i) \mod N = N/2 \\
      0 & \text{otherwise}
      \end{cases}$$

      of different commodities.

  - **Output:** A flow of the above commodities, completely satisfying the demands.

Our problem is to find a lightpath graph (and a valid coloring of it) with as few ADMs as possible, admitting a routing of the commodities $D(i, j)$.

The shortest path routing described earlier satisfies the following symmetry property: When a directed request of one of the directed instances is reversed, a directed request of the dual instance is obtained. Therefore, by reversing the directions of the lightpaths and the flows of a solution we obtain a solution of the dual instance. This means that we can view any solution of a directed instance, as a solution of the undirected instance, by simply disregarding the directions.
4.2.2 Preliminary Results

**Proposition 4.2.2** The circle \((0, 1, ..., N - 1, 0)\) is a subgraph of the lightpath graph of any solution.

This is because for each edge \(e = (i, i+1)\) the traffic \(\mathcal{T}(i, i+1)\) is non-zero and should be routed on the shortest path which is formed by a single lightpath consisting of \(e\) only. Therefore each such edge \(e\) is an edge of the lightpath graph.

**Proposition 4.2.3** The Lightpath Graph of a solution is Circular Eulerian:

Consider the links \(e\) and \(e'\) entering and leaving a node. Their capacities are both \(W\) and fully used. The capacity dedicated to passthrough traffic is the same in both of them, and uses the same set of wavelengths. Therefore, the capacity dedicated to the remaining traffic is the same and uses the same wavelengths in both edges. Therefore the out degree of any node equals to its in degree. The underlying graph is connected, otherwise there are two distinct nodes \(i\) and \(j\) such that \(D(i, j)\) can not be routed.

As such, this graph can be decomposed into simple cycles, each of which will be called a *polygon*.

**Definition 4.2.4** A *polygon* is a sequence of distinct nodes beginning with the least numbered node. The multiplicity of a polygon is the number of maximal increasing subsequences of this sequence. A polygon with multiplicity \(1\) is a *convex polygon*.

**Lemma 4.2.1** Any solution can be decomposed into convex polygons.

**Proof:** Any solution is a valid coloring of the lightpaths with \(W\) colors. Let \(E_c\) be the set of edges in the lightpath graphs such that the corresponding lightpath is colored (assigned wavelength) \(c\). Let: \(l(c) = \sum_{e \in E_c} l(e)\). For any color \(c\), \(l(c) \leq N\), because otherwise there is at least one edge containing two or more lightpaths with the same color, rendering the coloring invalid. On the other hand \(\sum_{c \in E} l(e) = \sum_{c=1}^{W} l(c) = WN\), because there are \(W\) edges (lightpaths) using any physical link. By the pigeonhole principle for all \(c\), \(l(c) = N\). The lightpaths of \(E_c\) do not overlap and the sum of their lengths is \(N\), therefore they form a convex polygon.

\[\square\]
**Corollary 4.2.1** The number of ADMs used by a color \(c\) is the number \(|E_c|\) of the edges of the corresponding convex polygon.

In view of the preceding results our design problem can be formulated as follows: Find \(W\) polygons with minimum total number of edges (nodes) such that routing problem has a solution.

### 4.3 The Demand Function

In this section we introduce the demand function which is important in our analysis:

**Definition 4.3.1** Given a demand matrix \(d\) and an edge \(e\) of a Lightpath Graph, we define:

\[
d(e) \buildrel d \over = \sum_{i,j} d(i, j)
\]

where the sum is taken over all the node pairs \(i, j\) such that \(e\) is in the direction of the path (on the directed circle) from \(i\) to \(j\).

In other words, \(d(e)\) is the total demand that can potentially be routed on the edge \(e\).

For the demand matrix \(D\) in Section 4.2, we define similarly \(D(e) = \sum_{i,j} D(i, j)\).

In Figure 4.2 we present a network with \(N = 12\) nodes. For the edge \(e\) depicted in the figure, we have \(D(e) = D(u_3, v_0) + D(u_2, v_0) + D(u_2, v_1) + D(u_1, v_0) + D(u_1, v_1) + D(u_0, v_0) + D(u_0, v_1) + D(u_0, v_2) + D(u_0, v_3)\).

Note that the summation includes the pairs \((u_i, v_j)\) for which the shortest path from \(u_i\) to \(v_j\) include \(e = (u_0, v_0)\).

**Definition 4.3.2** Given a polygon \(P\), we define:

\[
d(P) \buildrel d \over = \min \{d(e) | e \in P\}
\]

**Proposition 4.3.1** In every solution, all the edges of the lightpath graph satisfy \(D(e) \geq 1\).
Otherwise there is an edge with unused capacity under any shortest path routing. But the demand matrix can be routed only by using the full capacity of all the edges.

**Corollary 4.3.1** In every solution, all the polygons of the lightpath graph satisfy $D(P) \geq 1$.

**Lemma 4.3.1** Given a lightpath graph, we have:

$$D(e) > \tilde{D}(l(e)) \overset{\text{def}}{=} W \left(1 - \frac{l(e)}{n}\right)^2$$

for every edge $e$.

**Proof:** Consider an edge $e = (a, b)$ with $l(e) = l$ and a pair of nodes $u$ and $v$ such that $e$ is on the shortest path from $u$ to $v$. Let $i = a - u$ and $j = v - b$. Clearly $v - u \leq n$. But $v - u = (v - b) + (b - a) + (a - u) = j + l + i$, therefore $i + j \leq n - l$. The pairs of nodes satisfying this condition contribute $\tau$ to
\[ D(e) = \sum_{(i,j) \in E} \frac{\tau}{2} + \sum_{(i,j) \in E} \frac{\tau}{2} (n-l+1) \]
\[ = \tau \left[ \frac{(n-l)(n-l+1)}{2} + \frac{n-l+1}{2} \right] \]
\[ = \frac{W}{n^2} (n-l+1)^2 = W \left( 1 - \frac{1}{n} \right)^2 + O\left( \frac{W}{n} \right) \]

□

Note that \( D(e) \) depends only in the length \( l(e) \) of \( e \). With some abuse of notation, we will use \( D(l(e)) \) and \( D(e) \) with the same meaning. We will use the following simple properties of the demand function:

\[ \lim_{n \to \infty} (D(e) - D(l(e))) = 0 \]

\[ D^{-1}(x) \geq \tilde{D}^{-1}(x) = \left( 1 - \frac{x}{\sqrt{W}} \right) n \]

\( \tilde{D} \) and \( \tilde{D}^{-1} \) are both decreasing functions.

## 4.4 Nested Polygons

### 4.4.1 Definitions

**Definition 4.4.1** Consider two edges \( e = (i, j) \) and \( e' = (i', j') \) of a lightpath graph. Assume w.l.o.g. that \( i = 0 \). The edges are said to be:

- **disjoint** if \( j \leq i' < j' \)
- **crossing** if \( i' < j < j' \)
- **contained** if \( i' < j' \leq j \)

In the last case \( e' \) is said to be contained in \( e \).

**Definition 4.4.2** A convex polygon \( P' \) is nested in another convex polygon \( P \), if \( P \) is a cyclic permutation of some subsequence of \( P' \).
Proposition 4.4.1 If a polygon \( P' \) is nested in polygon \( P \), then any pair of edges \( e \in P \) and \( e' \in P' \), are either disjoint or \( e' \) is contained in \( e \).

Definition 4.4.3 A sequence \( P_1, P_2, \ldots, P_k \) of polygons is a nested sequence of polygons if for all \( i < k \), \( P_{i+1} \) is nested in \( P_i \).

4.4.2 Properties of Nested Polygons

Lemma 4.4.1 If a nested sequence of polygons is a solution, namely it admits a routing of the demand matrix \( D \), then:

\[
\forall i \leq W, \quad D(P_i) \geq i.
\]

Proof: Assume a nested sequence of polygons \( P_1, P_2, \ldots \) is a solution. Consider any routing admitted by this solution and any edge \( e_i \in P_i \). This edge is contained in exactly \( i - 1 \) edges \( e_1 \in P_1, e_2 \in P_2, \ldots, e_{i-1} \in P_{i-1} \). All the demands routed on the edges \( e_1, e_2, \ldots, e_{i-1} \) could be potentially routed on \( e_i \) too. Therefore the sum of the demands that could potentially routed on \( e_i \) is at least the sum of the demands actually routed on these edges, which is their total capacity, namely \( i \). Therefore, \( D(e_i) \geq i \). This is true for any edge \( e_i \in P_i \), we conclude \( D(P_i) \geq i \).

\[ \square \]

Lemma 4.4.2 If a nested sequence of polygons satisfies:

\[
\forall i \leq W, \quad D(P_i) \geq i.
\]

then there is a routing of the uniform demand matrix \( D \).

Proof: We present an algorithm constructing the claimed routing and prove its correctness:

\[
\text{RandomRoute}(\text{Demand } d, \text{ Polygon } P)\{
\]

// Routes as much as possible of the demand matrix \( d \\
// over the polygon \( P \\
\forall e \in P, f(e) = 0 \\
\text{For each pair of nodes } u, v \{ \\
\text{Let } e_1, e_2, \ldots, e_k \text{ be the edges of } P \text{ which are on the shortest} \\
\text{path from } u \text{ to } v
\]
and \( e_i = (a_i, b_i) \)
if \((k > 0)\) 
\[
\text{For } (i = 1; i \leq k; i++) \{ \\
\quad x_i = \min(d(u, v), 1 - f(e_i)) \\
\quad f(e_i) += x_i \\
\quad d(a_i, b_i) += d(u, v) - x_i \\
\}
\]
\[
d(u, a_1) += d(u, v) \\
d(b_k, v) += d(u, v) \\
d(u, v) = 0
\]
}

Route(Demand \( D \))
\[
d = D; \\
\text{for } \{i = 1; i \leq W; i++\} \{
\quad \text{RandomRoute}(d, P_i); \\
\}
\]

**Claim 4.4.1** If RandomRoute\((d, P)\) is invoked when \(d(P) \geq 1\), upon its return \(\forall e \in P, f(e) = 1\).

**Proof:** Consider any edge \(e \in P\). \(d(P) \geq 1\), therefore \(d(e) = \sum_{i,j} d(i, j) \geq 1\). Each pair contributing to this sum is considered exactly once for this edge by the algorithm. The value of \(d(i, j)\) when it is considered by the algorithm is at least equal to its value in the beginning of RandomRoute. This is because \(d(i, j)\) is decreased only after it is considered. It contributes \(d(i, j)\) to \(f(e)\) until \(f(e) = 1\). The total contribution to \(f(e)\) is therefore \(\min(1, d(e)) = 1\). \(f(e)\) does not decrease through RandomRoute, which means that the value of \(f(e)\) upon return is 1.

\[
\square
\]

**Claim 4.4.2** If RandomRoute\((d, P)\) is invoked when \(d(P) \geq 1\), upon its return \(d(P')\) is decremented by 1 for all polygons \(P'\) nested in \(P\).
**Proof:** Consider a polygon $P'$ nested in $P$ and an edge $e' \in P'$. There is exactly one edge $e \in P$ containing $e'$. All other edges of $P$ are disjoint to $e'$. A decrease in $d(e')$ occurs only when $d(u,v)$ changes for some pair $u,v$. This is done always with a change (increase) in $f(e)$ or $f(e'')$ where $e''$ is a disjoint edge.

Case 1, The change is in $f(e)$: Whenever $f(e)$ is increased, $d(e')$ is decreased by the same amount (see Figure 4.3). By the previous claim these decreases sum up to 1.

![Figure 4.3: Change in $f(e)$](image)

Case 2, The change is in a disjoint edge $e''$: This change will not affect $d(e')$ because the corresponding increase in the demand $d(b'', v)$ (see Figure 4.4).

Therefore, $d(e')$ is decreased exactly by 1, and consequently, so is $d(P')$.

\[ \square \]

By induction on $W$, using the above results, we prove that: If $\forall i \leq W, \ D(P_i) \geq i$, the above algorithm ends with $f(e) = 1$ for all the edges. Therefore the nested sequence of polygons is a feasible solution.

\[ \square \]

Lemma 4.4.1 and Lemma 4.4.2 imply:
Theorem 4.4.1 A nested sequence of polygons $P_1, P_2, ..., P_W$ is a solution if and only if

$$\forall i \leq W, \quad D(P_i) \geq i.$$  \hfill (4.1)

4.4.3 Optimum Solution for $W=2$

As previously stated, any solution contains the circle $(0, 1, ..., N, 0)$. Furthermore, we know that this solution consists of two convex polygons, namely, the above circle and one convex polygon. The circle is nested in all convex polygons, therefore any solution for $W=2$ is a nested sequence of polygons $P_1, P_2$ where $P_2$ is the circle itself. It remains to find $P_1$:

We know that $P_1$ must satisfy $D(P_1) \geq 1$. Therefore all the edges $e \in P_1$ must satisfy $D(e) \geq 1$, implying $l(e) \leq D^{-1}(1)$. We want to find the polygon with minimum total number of edges, therefore we choose $l = \lceil D^{-1}(1) \rceil$ and build the polygon which consists of $\lceil 2n/l \rceil$ edges such that $l(e) = l$ and at most one edge such that $l(e) < l$. The number of edges in this polygon is

$$\left\lceil \frac{2n}{\lceil D^{-1}(1) \rceil} \right\rceil \leq \left\lceil \frac{2n}{\lceil D^{-1}(1) \rceil} \right\rceil = \left\lceil \frac{2n}{(1 - \frac{1}{\sqrt{2}})n} \right\rceil \approx \frac{2}{1 - \frac{1}{\sqrt{2}}} \approx 6.8$$
This solution uses $N + 7$ ADMs instead of $2N$ ADMs in PPWDM, still getting the same throughput. From the above discussion it follows that:

**Theorem 4.4.2** For $W = 2$ any optimum solution uses $N + 7$ ADMs.

### 4.4.4 A solution for any $W$ using Nested Polygons

We build a nested sequence $S$ of polygons $P_i$ as follows: all the polygons in the sequence share a special node 0, and the lengths of their edges are powers of two, except of possibly one shorter edge.

For each $1 \leq i < W$,

$$k_i = \begin{cases} \left\lfloor \log \tilde{D}^{-1}(i) \right\rfloor & \text{if } \tilde{D}^{-1}(i) \geq 1 \\ 0 & \text{otherwise.} \end{cases}$$

Starting at node 0, and going clockwise, all the edges of each polygon $P_i$ have length $l_i = 2^{k_i}$, except possibly for the last one which is shorter. Clearly, $k_i$ is a non increasing sequence. We thus get a nested sequence of polygons. The lengths $l_i$ satisfy $l_i = 2^{k_i} \leq \tilde{D}^{-1}(i)$. Therefore $\tilde{D}(l_i) \geq i$ (see end of Section 4.3). It follows that $D(P_i) \geq i$, which is the sufficient condition (4.1) for the solution to be feasible.

**Theorem 4.4.3** The number of ADMs used by the solution $S$ is at most $8W\ln W + 2N + O(W)$.

**Proof:** The number $ADM_i$ of the ADMs of polygon $P_i$ is: $ADM_i = \left\lfloor \frac{N}{2^i} \right\rfloor$.

For every $i$ such that $\tilde{D}^{-1}(i) \geq 1$ we have:

$$ADM_i - 1 \leq \frac{N}{2\left\lfloor \log \tilde{D}^{-1}(i) \right\rfloor} < \frac{N}{2\left( \log \tilde{D}^{-1}(i) - 1 \right)} = \frac{2N}{\tilde{D}^{-1}(i)} = \frac{4}{\left(1 - \frac{\tilde{D}}{\sqrt{W}}\right)} = \frac{4\sqrt{W}}{\sqrt{W} - \sqrt{i}}$$

For other values of $i$ we have $ADM_i = N$. Note that, $\tilde{D}^{-1}(i) \geq 1$ if and only if $i \geq \tilde{D}(1) = W(1 - 1/n)^2$. The total number of ADMs satisfies:

$$\sum_{i=1}^{W} ADM_i \leq W + 4\sqrt{W} \sum_{i=1}^{\left\lfloor W(1-1/n)^2 \right\rfloor-1} \frac{1}{\sqrt{W} - \sqrt{i}} + \sum_{i=\left\lfloor W(1-1/n)^2 \right\rfloor}^{W} N$$

96
The first sum above is bounded by
\[
4\sqrt{W} \sum_{i=1}^{W-2} \frac{1}{\sqrt{W} - \sqrt{i}} \leq 4\sqrt{W} \int_1^{W-1} \frac{1}{\sqrt{W} - x} dx \\
= 8\sqrt{W} \left( \sqrt{W} \ln(\sqrt{W} - x) + x \right)_{w-1}^1 \\
= 8W \ln(\sqrt{W} - 1) - 8W \ln(\sqrt{W} - \sqrt{W - 1}) + 8\sqrt{W}(1 - \sqrt{W - 1}) \\
\leq 8W \ln \frac{\sqrt{W} - 1}{\sqrt{W} - \sqrt{W - 1}} \\
= 8W \ln(\sqrt{W} - 1)(\sqrt{W} + \sqrt{W - 1}) \\
< 8W \ln(2W) \\
= 8W \ln W + O(W)
\]
and the second sum is bounded by:
\[
N(W - W(1 - 1/n)^2 + 2) \\
= N(W(1 - (1 - 1/n)^2) + 2) = N \left( W \frac{1}{n} \left( 2 - \frac{1}{n} \right) + 2 \right) \\
< 4W + 2N
\]

Summing both bounds we get:
\[
\sum_{i=1}^{W} ADM_i \leq 8W \ln W + 2N + O(W)
\]

Conclusion: If \( W = O(N/\log N) \) the asymptotic cost depends only on \( N \), and if \( W = \Omega(N^{1+\varepsilon}/\log N) \) for any \( \varepsilon > 0 \), then the asymptotic cost depends only on \( W \).

4.4.5 An improved upper bound

Now we show how the \( O(W \log W + N) \) upper bound of Theorem 4.4.3 can be further improved.

Lemma 4.4.3 The problem is sub-additive in \( W \).
**Proof:** A solution $S_1$ for $W_1$ wavelengths and $N$ nodes and a solution $S_2$ for $W_2$ wavelengths and $N$ nodes can be superposed to obtain a solution for $W_1 + W_2$ wavelengths. This is true because:

$$
\tau = \frac{2(W_1 + W_2)}{n^2} = \frac{2W_1}{n^2} + \frac{2W_2}{n^2} = \tau_1 + \tau_2
$$

where $\tau$ (resp. $\tau_1, \tau_2$) are the uniform demands for $W$ (resp. $W_1, W_2$) wavelengths.

\[ \square \]

**Theorem 4.4.4** There is a solution using $O(W \log \overline{W} + N)$ ADMs, where $\overline{W} = o(W)$.

**Proof:** We omit constant factors. We consider two cases:

- $W \log W$ is $O(N)$. In this case our upper bound is $O(N)$ which is optimal.

- $N$ is $o(W \log W)$. In this case let $\overline{W}$ be such that $N = \overline{W} \log \overline{W}$. $\overline{W}$ is $o(W)$. Let $X = W/\overline{W}$. Because of the sub-additivity, the superposition of $X$ solutions of an instance with $\overline{W}$ wavelengths and $N$ nodes is a solution for our instance of $W$ wavelengths and $N$ nodes. This solution uses

$$
X \cdot O(\overline{W} \log \overline{W} + N) = X \cdot O(N) = O(\frac{W}{\overline{W}} \overline{W} \log \overline{W}) = O(W \log \overline{W})
$$

ADMs.

\[ \square \]

### 4.5 Generalization and Implications to Traffic Grooming

In this section we generalize the above results for all $N$ and analyze the applicability to the traffic grooming problem. We begin with a discussion on the traffic grooming problem in Subsection 4.5.1. In Subsection 4.5.2 we generalize our shortest path routing for any value of $N$. In Subsection 4.5.3 we show that the demand functions implied by these routings have the desired properties, and we end by summarizing the applicability of our results to the Traffic Grooming problem in Subsection 4.5.4.
4.5.1 The Traffic Grooming Problem

In the grooming problem, traffic requests have capacities which are multiples of some basic traffic unit which is equivalent to $1/g$ of the capacity of one wavelength, where $g$ is an integer called the grooming factor. In this context it is more convenient to express the capacities in terms of this basic unit. The capacity of the fiber is $W \cdot g$ and $\tau$ must be an integer.

We consider two variants of the uniform all-to-all multi-hop traffic grooming problem. In the first variant bifurcation is not allowed, i.e. a demand from node $i$ to node $j$ should be routed entirely on one sequence of lightpaths. In the second variant, bifurcation is allowed, but it is restricted to integer units, i.e. a demand may be split into sub-demands and each sub-demand may use its own sequence of lightpaths as long as the sub-demands have integer capacities.

In our discussion $N$ was restricted to be even. In this section we extend the results to any $N$, with the grooming problem in mind.

We restrict our attention to the instances in which $\tau$ is an integer, otherwise the capacity of the fiber cannot be entirely used. Note that if $\tau$ divides $g$, the algorithm RandomRoute at Section 4.4 will not cause demands to be bifurcated. The problem is that the shortest path routing that we use splits some of demands into two before they are input to RandomRoute, therefore our solution bifurcates traffic. Even if bifurcating is allowed we need that $\tau$ be even, so that each sub-demand is an integer. In the sequel we will generalize our solution to any $N$ and consider other shortest path routings so that traffic is not bifurcated as far as it is possible.

4.5.2 Shortest Path Routings

Under any shortest path routing, the load induced by node $i$ to the bidirectional cycle is $\sum_{j=1}^{N-1} \tau \min(j, N-j)$. Therefore the total load is $N \sum_{j=1}^{N-1} \tau \min(j, N-j)$. When this load is distributed evenly among the $2N$ directed edges, the
load $l(e)$ on each directed edge becomes

$$
\frac{1}{2} \sum_{j=1}^{N-1} \tau \min(j, N - j) = \frac{1}{2} \left\{ \begin{array}{ll}
2\tau \sum_{j=1}^{\left\lfloor \frac{N}{2} \right\rfloor} j & \text{if } N \text{ is odd} \\
2\tau \sum_{j=1}^{\left\lfloor \frac{N}{2} \right\rfloor} j + \tau \frac{N}{2} & \text{if } N \text{ is even}
\end{array} \right.
$$

$$
= \tau \left\{ \begin{array}{ll}
\frac{1}{2} \left\lfloor \frac{N}{2} \right\rfloor \left( \left\lfloor \frac{N}{2} \right\rfloor + 1 \right) & \text{if } N \text{ is odd} \\
\frac{1}{2} \left( \left\lfloor \frac{N}{2} \right\rfloor - 1 \right) + \frac{N}{4} = \frac{N^2}{4} & \text{if } N \text{ is even}
\end{array} \right.
$$

In the sequel $n \overset{\text{def}}{=} \left\lfloor \frac{N}{2} \right\rfloor$ which is consistent with its earlier definition. The shortest path routing is uniquely defined for all demands except the ones traveling the half length of the cycle. These demands can be routed in any one of the directions, or may be split and routed in both directions. We consider three cases.

- **N is odd.** In this case the shortest paths routing is unique. From symmetry considerations the load on each $e$ is the same, i.e., $\frac{x}{2} \left\lfloor \frac{N}{2} \right\rfloor \left\lceil \frac{N}{2} \right\rceil = \frac{x}{2} n(n + 1)$, and there is no bifurcation. Moreover the routing satisfies the symmetry property.

- **N is even, but not divisible by four.** In this case $l(e)$ is not an integer multiple of $\tau$, therefore any shortest path routing inducing a uniform load on the edges, should split some of the demands. The routing depicted in Figure 4.1, for which we proved our results, is an example of such a routing.

- **N is divisible by four.** The routing in which traffic from node $i$ to node $i + \frac{N}{2}$ is routed clockwise (resp. counterclockwise) when $i$ is even (resp. odd), induces a uniform load on the edges. This is because for any even (resp. odd) $i$, the traffic from $i$ to $i + \frac{N}{2}$ together with traffic from $i + \frac{N}{2}$ to $i$ is routed clockwise (resp. counterclockwise) and induces a uniform load on all the edges in this direction. This routing does not bifurcate traffic. Note that it does not satisfy the symmetry property, however we show that our solution remains valid.

### 4.5.3 The demand function

We now show that Lemma 4.3.1 holds. The case of $N$ even, but not divisible by four is already analyzed. We remain with the two other cases:
\( \bullet \) \( N \) is odd.

\[
D(e) = \sum_{\{i,j\} \mid i+j \leq n-l} \tau = \tau \sum_{s=1}^{n-l+1} s \\
= \tau \frac{(n - l + 1)(n - l + 2)}{2} = \frac{W}{n(n+1)}(n - l + 1)(n - l + 2) \\
= W \left( 1 - \frac{l}{n} + \frac{1}{n} \right) \left( 1 - \frac{l}{n+1} + \frac{2}{n+1} \right) = W \left( 1 - \frac{1}{n} \right)^2 + O\left( \frac{W}{n} \right)
\]

\( \bullet \) \( N \) is divisible by four. Let \( e = (a,b) \), then for the clockwise direction (the other direction is symmetric) we have

\[
D(e) = \sum_{\{i,j\} \mid i+j \leq n-l} \tau + \sum_{\{i,j\} \mid i+j = n-l, a-i \text{ is odd}} \tau \\
\geq \tau \left( \sum_{s=1}^{n-l} s + \left[ \frac{n - l + 1}{2} \right] \right) \\
\geq \tau \left( \frac{(n - l)(n - l + 1)}{2} + \frac{n - l}{2} \right) = \frac{\tau}{2}(n - l)(n - l + 2) \\
= \frac{W}{n^2}(n - l)(n - l + 2) = W \left( 1 - \frac{l}{n} \right) \left( 1 - \frac{l}{n} + \frac{2}{n} \right) \\
= W \left( 1 - \frac{1}{n} \right)^2 + O\left( \frac{W}{n} \right).
\]

The above discussion implies that Lemma 4.3.1 holds for every value of \( N \) using the appropriate shortest path routings, and therefore it is clear that the rest of the results apply as well for any \( N \).

### 4.5.4 Implications to the Grooming Problem

Recall that \( \tau = \frac{2W_n}{\left\lceil \frac{n}{2} \right\rceil \left\lfloor \frac{n}{2} \right\rfloor} \). The following table summarizes the sufficient conditions for our construction to be a solution for the uniform, all-to-all, multi-hop grooming problem in a bidirectional ring with maximal usage of the fiber capacity.

\[\text{Table Here}\]
\[
\begin{array}{|c|c|c|}
\hline
N \mod 4 & \text{Bifurcation Allowed} & \text{Bifurcation not Allowed} \\
\hline
1, 3, 4 & \frac{2W}{g} \in \mathbb{N} & \frac{W}{2} \in \mathbb{N} \\
2 & \frac{W}{g} \in \mathbb{N} & \text{No solution exists} \\
\hline
\end{array}
\]

4.6 Conclusion and Possible Improvements

We dealt with the splitable, multi-hop communication. We considered the all-to-all uniform traffic instance under the constraint that the full bandwidth of the fiber is used.

We presented an architecture that uses \(O(W \log \mathcal{W} + N)\) ADMs (\(\mathcal{W} = o(W)\)) which provides a solution for the traffic grooming problem under certain conditions.

An \(O(W + N)\) lower bound for the number of ADMs is immediate. The problem of closing the gap between our \(O(W \log \mathcal{W} + N)\) solution and this lower bound remains open. Another open question is whether the problem considered is NP-complete.

Our technique can be extended to non-nested polygons. The question of the performance of a feasible solution using this architecture, is open.
Chapter 5

Traffic Grooming in Ring Networks

5.1 Introduction

5.1.1 Background

In graph-theoretic terms, the traffic grooming problem can viewed as assigning colors to paths so that at most $g$ of them ($g$ being the grooming factor) can share one edge. In terms of ADMs, each light path uses two ADMs, one at each endpoint, and in case $g$ light paths of the same wavelength enter through the same edge to one node, they can all use the same ADM (thus saving $g-1$ ADMs). The goal is to minimize the total number of ADMs.

In this work we deal with the single hop problem, where a connection is carried along one wavelength. A nice review on traffic grooming problems can be found in [ZM03].

5.1.2 Previous Work

The notion of traffic grooming ($g > 1$) was introduced in [GRS98] for the ring topology. The problem was shown to be NP-complete in [CM00] for ring networks and a general $g$. The uniform all-to-all traffic case, in which there is the same demand between each pair of nodes, is studied in [CM00, BC03] for various values of $g$; an optimal construction for the uniform all-to-all problem, for the case $g = 2$ in a path network was given in [BBC05].
The hardness results of [EMZ02, CM00] are for $g = 1$ and for general $g$, respectively. NP-completeness results for ring and path networks are shown in [Ung05] for any fixed value of $g$.

### 5.1.3 Our Contribution

We present an approximation algorithm for the general instance of the traffic grooming problem, namely general topology and general set of requests. The approximation ratio of our algorithm is $2 \ln g + o(\ln g)$ in ring networks, with arbitrary set of requests. The ring topology is the most widely studied topology due to its implementation in SONET networks. Therefore and for matter of presentation, our discussion deals only with ring topologies. The extensions are briefly discussed in Section 5.5. Note that the approximation ratio of any algorithm for this problem is between 1 and $2g$. To the best of our knowledge this is the first approximation algorithm for the grooming problem with a general grooming factor $g$. In Section 5.2 we describe the problem and make some preliminary observations. The algorithm presented in Section 5.3, and analyzed in Section 5.4. We conclude in Section 5.5 with possible extensions of this result and some open problems.

### 5.2 Problem Definition

An instance of the traffic grooming problem is a triple $(G, P, g)$ where $G = (V, E)$ is an undirected graph, $P$ is a set of simple paths in $G$ and $g$ is a positive integer, namely the grooming factor.

Given such an instance we define the following:

**Definition 5.2.1** Given a subset $Q \subseteq P$ and an edge $e \in E$, $Q_e$ is the set of paths from $Q$ using edge $e$. $l_Q(e)$ is the number of these paths, or in networking terminology, the load induced on the edge $e$ by the paths in $Q$. $L_Q$ is the maximum load induced by the paths in $Q$ on any edge of $G$. When $Q = P$, we will omit the indices and simply write $l(e)$ and $L$ instead of $l_P(e)$.
and $L_P$ respectively. Formally,

$$ \forall Q \subseteq P, \forall e \in E : $$

$$ Q_e \overset{\text{def}}{=} \{ p \in Q | e \in p \} $$

$$ l_Q(e) \overset{\text{def}}{=} |Q_e| $$

$$ L_Q \overset{\text{def}}{=} \max_{e \in E} l_Q(e) $$

**Definition 5.2.2** A coloring (or wavelength assignment) of $(G, P)$ is a function $w : P \mapsto \mathbb{N}^+ = \{1, 2, \ldots\}$. We extend the definition of $w$ on any subset $Q$ of $P$ as $w(Q) = \bigcup_{p \in Q} w(p)$. For a coloring $w$, a color $\lambda$ and any $Q \subseteq P$, $Q^w_\lambda$ is the subset of paths from $Q$ colored $\lambda$ by $w$ and $Q^w_{e, \lambda}$ is the set of paths from $Q$, using edge $e$ and colored $\lambda$ by $w$. Formally,

$$ Q^w_\lambda \overset{\text{def}}{=} w^{-1}(\lambda) \cap Q = \{ p \in Q | w(p) = \lambda \} $$

$$ Q^w_{e, \lambda} \overset{\text{def}}{=} Q_e \cap Q^w_\lambda. $$

**Definition 5.2.3** A proper coloring (or wavelength assignment) $w$ of $(G, P, g)$ is a coloring of $P$, in which for any edge $e$ at most $g$ paths using $e$ are colored with the same color. Formally, $\forall \lambda \in \mathbb{N}^+, L^w_\lambda \leq g.$

**Definition 5.2.4** A coloring $w$ is a $W$-coloring of $Q \subseteq P$, if it colors the paths of $Q$ using exactly $W$ colors. Formally, if $|w(Q)| = W$. A set $Q$ is $W$-colorable if there exists a proper $W$-coloring for it.

For a $W$-coloring of $P$, we will assume w.l.o.g. that $w(P) = 1, 2, \ldots, W$.

Observe that a set $Q \subseteq P$ is 1-colorable iff $L_Q \leq g$.

Now we define the cost function $ADM$, under the assumption that $G$ is a cycle.

**Definition 5.2.5** For a coloring $w$ of $P$, a subset $Q \subseteq P$ and a node $v \in V$, $Q_v$ is the subset of paths from $Q$ having an endpoint in $v$. $Q^w_{v, \lambda}$ is the subset of paths from $Q_v$ colored $\lambda$ by $w$. $ADM^w_\lambda (v)$ is the number of $ADM$’s operating at wavelength $\lambda$ at node $v$.

For each pair $v \in V, \lambda \in \{1, 2, \ldots, W\}$ we need one $ADM$ operating at wavelength $\lambda$ in node $v$ iff there is at least one path colored $\lambda$ among the
paths having an endpoint at \( v \). Formally,

\[
Q_v \overset{\text{def}}{=} \{ p \in Q | v \text{ is an endpoint of } p \}
\]

\[
touches(Q, v) \overset{\text{def}}{=} \begin{cases} 
0 & \text{if } Q_v = \emptyset \\
1 & \text{otherwise} 
\end{cases}
\]

\[
\text{endpoints}(Q) \overset{\text{def}}{=} \sum_{v \in V} touches(Q, v)
\]

\[
Q^w_{v, \lambda} \overset{\text{def}}{=} Q_v \cap Q^w_{\lambda}
\]

\[
ADM^w_{\lambda}(v) \overset{\text{def}}{=} touches(P^w_{\lambda}, v)
\]

\[
ADM^w_{\lambda}(Q) \overset{\text{def}}{=} \text{endpoints}(Q^w_{\lambda})
\]

\[
ADM^w_{\lambda}(P) = \sum_{\lambda} ADM^w_{\lambda}
\]

**Definition 5.2.6** For any subset \( Q \subseteq P \) and any subset \( U \subseteq V \), \( Q_U \) is the set of paths in \( Q \) having at least one endpoint in \( U \). Formally,

\[
Q_U \overset{\text{def}}{=} \bigcup_{u \in U} Q_u.
\]

The traffic grooming problem is the optimization problem of finding a proper coloring \( w \) of \( (G, P, g) \) minimizing \( ADM^w \).

Observe that \( \text{endpoints} \) and consequently \( ADM^w_{\lambda} \) are monotone non-decreasing functions. Formally, if \( R \subseteq Q \subseteq P \) then

\[
\text{endpoints}(R) \leq \text{endpoints}(Q)
\]

\[
ADM^w_{\lambda}(R) \leq ADM^w_{\lambda}(Q).
\]

### 5.3 Algorithm GROOMBYSC(k)

Given an instance \( (G, P, g) \) of the traffic grooming problem, our algorithm has a parameter \( k \) which depends only on \( g \). The value of \( k \) will be determined in the analysis (see Section 5.4).

The algorithm has three phases. During phase 1 it computes 1-colorable sets and their corresponding weights. It considers subsets of the paths \( P \),
of size at most $k \cdot g$. Whenever a 1-colorable set is found, it is added to the list of relevant sets, together with its corresponding weight. In phase 2 it finds a set cover of $P$ using subsets calculated in phase 1. It uses the GREEDYSC approximation algorithm for the minimum weight set cover problem presented in [Chv79]. In phase 3 it transforms the set cover into a partition by eliminating intersections, then colors the paths according the partition. Each set in the partition is colored with one color.

a Phase 1- Prepare the input for GREEDY:

$$S \leftarrow \emptyset$$

For each $U \subseteq V$, such that $|U| \leq k$ {
   For each $Q \subseteq P_V$, such that $|Q| \leq k \cdot g$ {
      If $Q$ is 1-colorable then {
         $$S \leftarrow S \cup \{Q\}$$
         $$\text{weight}[Q] = \text{endpoints}(Q)$$ // weight[] is an associative
         // array containing a weight for each set
      }
   }
}

b Phase 2- Run GREEDYSC:

$$SC \leftarrow \text{GREEDYSC}(S, \text{weight}).$$ // Assume w.l.o.g $SC = \{S_1, S_2, \ldots, S_W\}$

c Phase 3- Transform the Set Cover $SC$ into a Partition $PART$:

$$PART \leftarrow \emptyset$$

For $i = 1$ to $W$ {
   $$PART_i \leftarrow S_i$$
}

As long as there are two intersecting sets $PART_i, PART_j$ {
   $$PART_i \leftarrow PART_i \setminus PART_j$$
}

For $\lambda = 1$ to $W$
   $$PART \leftarrow PART \cup \{PART_\lambda\}$$
   For each $p \in PART_\lambda$ $(w(p) = \lambda)$

107
5.4 Analysis

5.4.1 Correctness

Claim 5.4.1 \( w \) calculated by the algorithm is a coloring.

Proof: During phase 1, each path \( p \in P \) is included at least in one set \( Q \in S \). This is because the set \( \{p\} \) is considered during the loop and it is clearly found to be 1-colorable. As \( SC \) is calculated in phase 2 a set cover of these sets, \( p \) is an element of at least one set \( S_i \in SC \). During phase 3 intersections are eliminated, therefore \( p \) is an element of exactly one set of \( PART \). Therefore each \( p \) is assigned exactly one value \( w(p) \) during phase 3.

\( \square \)

Lemma 5.4.1 \( w \) calculated by the algorithm is a proper coloring.

Proof: For every color \( \lambda \in \{1, 2, ..., W\} \) the set of paths colored \( \lambda \) is exactly \( PART_\lambda \). It suffices to show that the sets \( PART_\lambda \) are 1-colorable.

A subset of an \( x \)-colorable set is \( x \)-colorable. By the code of phase 3 \( PART_\lambda \subseteq S_\lambda \). By phase 1, \( S_\lambda \) is 1-colorable, therefore \( PART_\lambda \) is 1-colorable.

\( \square \)

5.4.2 Running Time

Claim 5.4.2 The running time of GROOMBYSC(\( k \)) is polynomial in \( n = |P| \) and \( m = |E| \), for any given \( g \) and for all instances \((G, P, g)\).

Proof: We will show that the running time of each one of the three phases is \( poly(n, m) \).

- Phase 1:
  The number of subsets of \( P \) considered during the first phase is \( O(n^{g_k}) \) since their sizes are at most \( g \cdot k \). To check whether a set is 1-colorable takes \( O(g \cdot k \cdot m) \) time. To calculate \( endpoints(Q) \) can be done in \( O(g \cdot k \log |Q|) = O(g \cdot k \cdot \log m) \) time.

  For any constant \( g, k \) is determined as a function of \( g \) only. Then \( g \cdot k \) is a constant. Therefore the running time of phase 1 is polynomial in \( n \) and \( m \) for any given \( g \).
\begin{itemize}
  \item **Phase 2:**
  The number of the sets in $S$ is at most $n^{g^k}$. The running time of GREEDYSC is polynomial in $|S|$ and $|P|$, namely $poly(n^{g^k}, n) = poly(n)$.
  \item **Phase 3:**
  The running time of phase 3 is polynomial in the size of the cover which is in turn polynomial in $n$.
\end{itemize}

5.4.3 Approximation Ratio

**Lemma 5.4.2** Let $H_n = 1 + \frac{1}{2} + \ldots + \frac{1}{n}$ be the $n$-th harmonic number. \(GROOMBYSC(k)\) is a $H_{g^k}(1 + \frac{g^k}{k})$ approximation algorithm for the traffic grooming problem in ring networks.

**Proof:** Recall that in the Minimum Weight Set Cover problem, each subset $S_i$ has an associated weight, \textit{weight}[$S_i$]. The weight of a cover is the sum of the individual weights of its sets.

Let $w$ be the coloring returned by \(GROOMBYSC(k)\) and $w^*$ an optimal coloring. We will use the shortcut $ADM^*$ for $ADM^{w^*}$.

On one hand

$$ADM^w = \sum_{\lambda} ADM^w_\lambda = \sum_{\lambda} \text{endpoints}(PART_\lambda)$$

$$\leq \sum_{\lambda} \text{endpoints}(S_\lambda) = \sum_{\lambda} \text{weight}[S_\lambda] = \text{weight}(SC). \quad (5.1)$$

On the other hand GREEDYSC is an $H_f$-approximation algorithm, where $f$ is the maximum cardinality of the sets in the input. In our case $f = g \cdot k$. In other words if $SC^*$ is a minimum weight set cover on the set $S$, we have

$$\text{weight}(SC) \leq H_{g^k} \text{weight}(SC^*). \quad (5.2)$$

Clearly if $SC$ is an arbitrary set cover of $S$, by definition

$$\text{weight}(SC^*) \leq \text{weight}(SC). \quad (5.3)$$

Combining the inequalities (5.1), (5.2) and (5.3) we get

$$ADM^w \leq H_{g^k} \text{weight}(SC)$$

109
for any set cover $\overline{SC}$ of $S$.

In the following claim we will show the existence of a set cover $\overline{SC}$ satisfying $\text{weight}(\overline{SC}) \leq ADM^* \left(1 + \frac{2g}{k}\right)$, which implies

\[ ADM^w \leq ADM^* H_{y_k} \left(1 + \frac{2g}{k}\right). \]

\[ \Box \]

Claim 5.4.3 There exists a set cover $\overline{SC}$ of $S$, such that

\[ \text{weight}(\overline{SC}) \leq ADM^* \left(1 + \frac{2g}{k}\right). \]

Proof: Let $w^*(P) = \{1, 2, \ldots, W^*\}$ and $1 \leq \lambda \leq W^*$. Consider the set $V^*_\lambda$ of nodes $v$ such that $ADM^*_\lambda(v) = 1$, namely having an ADM operating at wavelength $\lambda$ at node $v$. We divide $V^*_\lambda$ into sets of $k$ nodes starting from an arbitrary node and going clockwise along the cycle (see Figure 5.1). Let $V_{\lambda,j}$ be the subsets of nodes obtained in this way. Let

\[ ADM^*_\lambda = |V^*_\lambda| = k q_\lambda + r_\lambda \quad (5.4) \]

where $r_\lambda = |V^*_\lambda|$ mod $k$ and $0 \leq r_\lambda < k$.

Clearly $\forall 1 \leq j \leq q_\lambda, |V_{\lambda,j}| = k$, and in case $r_\lambda > 0$ we have $|V_{\lambda, q_\lambda+1}| < k$. In both cases $|V_{\lambda,j}| \leq k$. Therefore, each $V_{\lambda,j}$ is considered in the outer loop of phase 1 of the algorithm, and hence, is added to $S$.

For $V_{\lambda,j}$ we define $\overline{S}_{\lambda,j}$ to be the set of paths in $P^w_\lambda$ having their counterclockwise endpoint in $V_{\lambda,j}$. As $V_{\lambda,j}$ has at most $k$ nodes, and every node may be the clockwise endpoint of at most $g$ paths from a 1-colorable set, we have $|\overline{S}_{\lambda,j}| \leq g \cdot k$. Therefore, $\overline{S}_{\lambda,j}$ is considered by the algorithm in the inner loop of phase 1. Being 1-colorable it should be added to $S$, thus $\overline{S}_{\lambda,j} \in S$.

Every $p \in P^w_\lambda$ has its both endpoints in the sets $V_{\lambda,j}$. In particular, it has its clockwise endpoint in $V_{\lambda,j}$ for a certain $j$, thus it is an element of some $\overline{S}_{\lambda,j}$. Therefore $\overline{SC}_\lambda \overset{def}{=} \bigcup_j \{\overline{S}_{\lambda,j}\}$ is a cover of $P^w_\lambda$. Considering all colors $1 \leq \lambda \leq W^*$ we conclude that $\overline{SC} \overset{def}{=} \bigcup_{\lambda=1}^{W^*} \overline{SC}_\lambda$ is a cover of $P$.

Therefore $\overline{SC}$ is a cover of $P$ with sets from $S$. It remains to show that its weight has the claimed property.
Figure 5.1: The sets $V_{\lambda,j}$ and $\overline{S}_{\lambda,j} (k = 4)$

Summing up equation (5.4) over all possible values of $\lambda$ we obtain $ADM^* = k \sum \lambda q_\lambda + \sum \lambda r_\lambda$, which implies:

$$\sum \lambda q_\lambda \leq \frac{ADM^*}{k}$$  \hfill (5.5)

We claim that $\forall j \leq q_\lambda, weight(\overline{S}_{\lambda,j}) = endpoints(\overline{S}_{\lambda,j}) \leq k + g$. This is because:

- The endpoints of the paths with both endpoints in $\overline{S}_{\lambda,j}$ are in $V_{\lambda,j}$ and $|V_{\lambda,j}| = k$.
- The number of paths having only the clockwise endpoint in set $V_{\lambda,j}$ is at most $g$. This follows from the observation that these paths should use the unique edge in the clockwise cut of $V_{\lambda,j}$. As the set $\overline{S}_{\lambda,j}$ is 1-colorable, the number of these paths is at most $g$.

For the set $j = q_\lambda + 1$ (which exists only if $r_\lambda > 0$) the above bound becomes $weight(\overline{S}_{\lambda,q_\lambda+1}) \leq r_\lambda + g \cdot q_\lambda$. This is because:

- The endpoints of the paths with both endpoints in $\overline{S}_{\lambda,q_\lambda+1}$ are in $V_{\lambda,q_\lambda + 1}$ and $|V_{\lambda,q_\lambda + 1}| = r_\lambda$. 

111
• By the same argument as before, the paths having only the clockwise endpoint in \( V_{\lambda q_\lambda + 1} \) are at most \( g \) in number. When \( q_\lambda \geq 1 \), \( g \leq g \cdot q_\lambda \) and we are done. Otherwise \( q_\lambda = 0 \) meaning that \( V_{\lambda 1} \) is the unique set. Then the number of paths having exactly one endpoint in the set is zero.

Summing up for all \( 1 \leq j \leq q_\lambda + 1 \) we get:

\[
weight(\overline{SC}_\lambda) \leq \sum_{j=1}^{q_\lambda} (k + g) + r_\lambda + g \cdot q_\lambda = (k + g)q_\lambda + r_\lambda + g \cdot q_\lambda = kq_\lambda + r_\lambda + 2g \cdot q_\lambda
\]

Summing up for all \( \lambda \) and recalling (5.4) and (5.5) we get:

\[
weight(\overline{SC}) = \sum_\lambda weight(\overline{SC}_\lambda) \leq \sum_\lambda (kq_\lambda + r_\lambda + 2g \cdot q_\lambda) = ADM^* + 2g \sum_\lambda q_\lambda
\]

\[
\leq ADM^* + 2g \frac{ADM^*}{k} = \left(1 + \frac{2g}{k}\right) ADM^*.
\]

\[\square\]

**Theorem 5.4.1** There is a \( 2 \ln g + o(\ln g) \)-approximation algorithm for the traffic grooming problem in ring networks.

**Proof:** The approximation ratio \( \rho \) of GROOMBYSC\((k)\) is at most \( H_{g-k} \left(1 + \frac{2a}{k}\right)\).

We substitute \( k = g \ln g \) and get:

\[
\rho \leq H_{g \ln g} \left(1 + \frac{2}{\ln g}\right) \leq (1 + \ln(g^2 \ln g)) \left(1 + \frac{2}{\ln g}\right)
\]

\[
= (1 + 2 \ln g + \ln \ln g) \left(1 + \frac{2}{\ln g}\right) = 2 \ln g + o(\ln g)
\]

\[\square\]

### 5.5 Conclusion and Possible Improvements

We presented an approximation algorithm for ring networks, whose approximation ratio is \( 2 \ln g + o(\ln g) \). Note that the approximation ratio of any algorithm for this problem is between 1 and 2\( g \).
Our algorithm can be used in arbitrary networks. In some topologies the analysis will yield a similar result. For this, note that the only point in the analysis that used the fact that the topology is a ring is where we considered the unique edge between the blocks of an optimal solution. Therefore a similar analysis follows for any topology and set of demands in which any solution can be partitioned in a similar way. This clearly includes all graphs which consists of blocks $B_0, B_1, \ldots, B_{b-1}$ whose sizes are bounded by $\alpha \leq k$ ($k$ is the parameter used in our analysis) and at most $\beta = O(1)$ edges connecting consecutive blocks $B_i$ and $B_{i+1 \mod b}$.

We mention few open problems which arise from this study.

- Improve the analysis of algorithm $GROOMBYSC(k)$.
- Find an algorithm with a better performance guarantee.
- Analyze algorithm $GROOMBYSC(k)$ for general topology and set of requests.
Chapter 6

Further Research

Each of the preceding chapters is concluded with possible improvements of our results. In Section 1.5 we presented various settings for problems that have been or could be investigated. In this chapter we mention further research directions for several of these problems.

6.1 The Chord Version of the minimum ADM Problem in rings

The minimum ADM problem studied in the Chapters 2 and 3 assumes the communication requests are already routed along the ring by another algorithm. In the first papers which defined the problem, this fact was justified by the fact that the routing problem and the WLA problem are each in NP-complete and it is logical to analyze each problem separately. On the other hand the analysis of both problems in terms of lower bounds and approximation algorithms have no immediate implications on the approximation ratio for the composite problem, namely the WRA problem which is called the chord version of the minimum ADM problem in ring networks.

The problem is formulated in [CW02b] as follows:

Instance: a set of chords $C$ along a ring.

Solution: a proper partition of $C$, $\Pi = \{C_1, C_2, \ldots, C_w\}$ such that for any $1 \leq i \leq w$ all chords in each $C_i$ can be routed as non-intersecting arcs over the ring.

Cost: the cost of each $C_i$ is the number of different nodes of the ring that are the endpoints of the chords on $C_i$, and the cost of the partition $\Pi$ is the
sum of the costs of $C_i$ for all $1 \leq i \leq w$. The minimum cost over all proper solutions is called the minimum ADM cost of $C$.

The $PAF$ algorithm presented in Chapter 2 and the $PIM$ algorithm [CW02b] are $3/2$ - approximation algorithms for the arc version of the problem. We have shown that the $IPAF$ and $IEMZ$ algorithms perform better. A $3/2$ - approximation algorithm is given for the chord version of the problem in the same work [CW02b]. It may be interesting to analyze the problem using the techniques used in Chapter 2, namely to study the relationship of the problem to the maximum disjoint cycles problem and trying to apply improved algorithms for this problem.

To the best of our knowledge no algorithms with proven approximation guarantees are known about the WRA problem in graphs other than rings.

### 6.2 Splitable Requests - Multi-Hop Communication

It was first observed and and argued in [GLS98] that the number of ADMs can be reduced by allowing a traffic stream to be transferred from one ADM in a wavelength to another ADM in a different wavelength at any intermediate node; in other words, the traffic streams are splitable. In the same work the Cut First algorithm is proposed and analyzed.

This has two variants, namely the arc version and the chord version of the problem. The solution offered in [GLS98] addresses only the arc version.

Both versions of the problem are addressed in [CW02a]. We present here the formal definition of the arc version of the problem as given in this work:

**Input:** a set of circular-arcs $A$ along a ring.

**Solution:** a choice of splitting each arc of $A$, thus obtaining $A'$, and then a proper partition of $A'$, $\Pi = \{A'_1, A'_2, ..., A'_w\}$, such that for any $1 \leq i \leq w$ all arcs in each $A'_i$ are non-intersecting.

**Cost:** the cost of each $A'_i$, id the number of different nodes of the ring that are the endpoints of the arcs in $A'_i$ for all $1 \leq i \leq w$. The minimum cost over all proper solutions is called the minimum ADM cost of $A$ with splittings.

Approximation algorithms for both version of the problem are presented in [CW02a] and for the arc version an approximation ratio of $5/4$ is shown. This result is further improved to a $16/13$-approximation algorithm, in [EL05] by using techniques similar to ones used in [EL04].
Our techniques can be used in this problem too and may lead to an improvement in these results.

The problem is not investigated much, if at all under non-ring networks.

6.3 Tradeoffs

A natural conjecture is that wavelength minimization algorithms may also simultaneously minimize the number of ADM’s. This makes sense because packing the lightpaths into a smaller number of wavelengths results in potentially more ADM sharing. For the arc version of the problem in Ring Networks it is shown in [GLS98] that the two problems are intrinsically different. This is shown in several examples in which the minima for the two objective functions do not meet.

It is not known whether or not there are any relationships between these minima. It is natural to study the tradeoffs between this objective functions. While it is natural and interesting to study these tradeoffs from a pure theoretical point of view, these tradeoffs will have a very practical aspect, once found. The cost of a system is a (possibly linear) combination of the wavelength cost and the ADM cost. The wavelength cost is a function of the cost of the ADM’s, optical routers, optical fibers and the physical topology. Tradeoffs between the minimum ADM costs and minimum wavelength cost will help us to study more easily the problem of minimizing the total cost of the system, which is as already stated, a combination of these parameters.

6.4 Online Algorithms and Competitive Ratio

As approximation algorithms usually require that the whole input is given to the algorithm in advance, they can be classified as off-line algorithms. In some applications, it is realistic to assume that data comprising the input arrives dynamically, one item at a time and an algorithm should serve the input as it arrives. Such algorithms are called on-line algorithms. In our problems these items will be connection requests or lightpaths.

The competitive ratio measures the worst case asymptotic approximation ratio of an on-line algorithm. More formally the input of an (deterministic) on-line algorithm ALG is a sequence $\sigma = \sigma_1\sigma_2...$ with unbounded length.
Each $\sigma_i$ is an input item. The goal of the on-line algorithm $ALG$ is to serve one input item $\sigma_i$ at a time without knowledge of the subsequent input items $\sigma_j$. Each service given by $ALG$ to an individual input has an associated cost (or profit in maximization problems). The sum of the costs (or profits) of all the inputs in a sequence $\sigma$ is denoted by $ALG(\sigma)$. $OPT(\sigma)$ denotes the optimum cost (or profit) possible on input sequence $\sigma$ when the whole sequence $\sigma$ is known in advance. For a minimization problem, a (deterministic) on-line algorithm $ALG$ is $c$-competitive if there is a constant $\alpha$ such that for all $\sigma$,

$$ALG(\sigma) \leq c \cdot OPT(\sigma) + \alpha.$$ 

For a maximization problem the condition is

$$ALG(\sigma) \geq \frac{1}{c} \cdot OPT(\sigma) - \alpha.$$ 

The WRA and WLA problems have been heavily studied in their wavelength minimization version. The on-line problem, in which the input (communication requests or lightpaths) arrive one at a time is studied in [AAF+96] and [BL97] for general graphs and tree networks and in [GSKR97] for rings. However, these works consider the minimization of the number of wavelengths.

The on-line problem is not studied as heavily as the off-line problem for several reasons. The problem is first seen and attacked as a network design problem, this in turn stemmed for the fact that optical switches’ setup times were very high and once a connection is set up it is not deleted for a relatively long period of time. The input to the problem is not the actual connection requests, but telecom carrier’s prediction of the communication demands between any two points of the network. With the reduction of the cost of the optical components lightpath communication will be available to a broad range of users. The reduction of the setup time of the lightpaths will make the establishment of short term connections feasible. All these will make the WRA and WLA problems on-line problems in their nature.

As far as the minimum ADM problem is concerned, the on-line problem is not too much studied if at all.

The on-line version of the problem also called Dynamic Routing and Wavelength Allocation can be defined in several flavors, depending on

- Whether or not rejection (deferral) of a request is allowed.
- Whether or not requests may be deleted.
6.5 Randomized Algorithms

It is well known fact that randomization, leads often to better approximation ratios in almost every kind of optimization problems. Randomized algorithms are used sometimes even to solve problems for which exact algorithms exist. This is done sometimes in order to achieve better time complexity or even better numeric performance, in terms of errors.

There were few attempts to use the randomization technics to solve the minimum ADM problem in particular and the WRA and WLA problems in general. To the best of our knowledge these few attempts did not lead yet to improved results.

118
Appendix A
A.1 Linear Program Used in Theorem 2.3.1
A.2 Linear Program Used in Theorem 2.4.1
A.3 Linear Program Used in Theorem 2.4.2
Bibliography


