UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING
SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Nina Skorin-Kapov

HEURISTIC ALGORITHMS FOR VIRTUAL TOPOLOGY DESIGN AND ROUTING AND WAVELENGTH ASSIGNMENT IN WDM NETWORKS

HEURISTIČKI ALGORITMI ZA PLANIRANJE VIRTUALNIH TOPOLOGIJA, USMJERAVANJE I DODJELJIVANJE VALNIH DULJINA U WDM MREŽAMA

DOCTORAL THESIS
DOKTORSKA DISERTACIJA

Zagreb, 2006.

The doctoral thesis was completed at the Department of Telecommunications of the Faculty of Electrical Engineering and Computing, University of Zagreb.

Advisor: Mladen Kos, Ph. D., Professor, Faculty of Electrical Engineering and Computing, University of Zagreb

The thesis has 145 pages.

Thesis number:

The dissertation evaluation committee:

1. Professor Vedran Mornar
2. Professor Mladen Kos
3. Associate Professor Nicholas Puech, Ecole Nationale Supérieur Des Télécommunications, Paris, France

The dissertation defense committee:

1. Professor Vedran Mornar
2. Professor Mladen Kos
3. Associate Professor Nicholas Puech, Ecole Nationale Supérieur Des Télécommunications, Paris, France
4. Professor Branko Mikac
5. Professor Ignac Lovrek

Date of dissertation defense: Math 4^{th}, 2006

Contents

List of Figures

List of Tables

Introduction

The tremendous growth of data traffic, primarily Internet traffic, in the past several years has created an ever increasing need for high-speed communication networks. As a result of the huge potential bandwidth of optical fibers, optical networks have been established as the enabling technology for future long-haul high-speed backbone networks. *Wavelength Division Multiplexing* (WDM) makes the utilization of the tremendous bandwidth of optical fibers possible by multiplexing data onto different wavelengths.

The development of future backbone optical networks is aimed towards a fully transparent all-optical network based on optical burst and packet switching. However, this goal is far from being realized. The high-speed backbone networks currently deployed are based on optical circuit switching and are commonly referred to as wavelength routed WDM networks. Nodes in these network can be configured to set up all-optical connections, called *lightpaths*, between pairs of nodes. These connections can traverse multiple links in the physical topology and yet transmission via a lightpath is entirely in the optical domain, i.e. there is no opto-electronic conversion at intermediate nodes. Establishing a set of lightpaths creates a virtual topology over the physical topology. Packet switched traffic is then routed over the virtual topology, completely independent of the underlying interconnection of optical fibers.

If nodes in the network are equipped with mechanisms which support multicasting (point-to-multipoint communication) on the WDM layer, a set of *light-trees* can be established. A light-tree is a generalization of a lightpath which optically connects a subset of nodes in the network, i.e. forms an all-optical tree which enables one-to-many communication entirely in the optical domain. A virtual topology composed of a set of light-trees is better suited to support multicast and broadcast traffic than a set of lightpaths.

In order to establish a set of lightpaths/light-trees, it is necessary to find corresponding routes in the physical topology and assign wavelengths to them. This problem is known as the Routing and Wavelength Assignment (RWA) problem. The most common objective is to minimize the number of wavelength used. The unicast RWA problem deals with establishing of a set of *lightpaths*, while the multicast RWA problem deals with establishing of a set of

light-trees. Demands to set up lightpaths/light-trees can be static, scheduled or dynamic. Static demands are known *a priori* and the virtual topology is established semi-permanently. For scheduled demands, the set-up and tear-down times of each lightpath/light-tree are known *a priori*. Dynamic demands are the case when requests for lightpaths/light-trees arrive unexpectedly with random holding times. Successful solvability of the RWA problem is crucial to efficiently utilizing resources in optical networks. Since this problem is NP-complete, efficient heuristic algorithms are needed to help solve it. This thesis is concerned with the problems of routing and assigning wavelengths to static and scheduled lightpath demands and static light-tree demands.

The successful *design* of virtual topologies is also crucial to utilizing the potential of wavelength routed optical networks. Designing and establishing a virtual topology composed of a set of lightpaths and/or light-trees is a complex problem. This problem, known as the Virtual Topology Design problem, consists of determining the set of lightpaths/light-trees which are to be established, solving the RWA problem for that set and, finally, routing packet switched traffic over the established virtual topology. Upon solving the Virtual Topology Design problem, several objective criteria can be considered. Objectives include minimizing the number of wavelengths used, minimizing the congestion and average hop distance of packet switched traffic through the virtual topology, minimizing the number of optical devices (e.g. transmitters and receivers) needed to establish a virtual topology with good performance measures, etc. This thesis investigates the design of virtual topologies consisting of a set of lightpaths considering various objective criteria.

The specific contributions of the thesis are the following.

- Greedy and meta-heuristic algorithms for the Routing and Wavelength Assignment of static and scheduled lightpath demands;
- A meta-heuristic algorithm for multicast routing, and greedy algorithms for the problem of static Multicast Routing and Wavelength Assignment;
- An additional objective criterion for the Virtual Topology Design problem and heuristic algorithms for virtual topology design considering various objectives;
- New analytical lower bounds to help asses the quality of the proposed heuristic algorithms.

The outline of the thesis is as follows. In Chapter 1, we provide a general introduction to optical networks to ease understanding of the problems and issues discussed in the thesis. Chapter 2 discusses optimization problems that arise in wavelength routed optical networks.

In Chapter 3, we investigate the problem of routing and assigning wavelengths to static lightpath demands. We propose greedy heuristic algorithms developed by applying the concepts of bin packing, and discuss lower bounds. Bin packing is a classical NP-complete optimization problem which, to the best of our knowledge, has not yet been used in the context of the RWA problem. In Chapter 4, we consider the Routing and Wavelength Assignment of scheduled lightpath demands. Two approaches are proposed. The first applies a tabu search meta-heuristic while the second is based on greedy algorithms. An analytical lower bound on the number of wavelengths needed for successful solvability is presented. In Chapter 5 we investigate the problem of Multicast Routing and Wavelength Assignment. The chapter is divided into two parts. The first considers the problem of multicast routing and a GRASP meta-heuristic is proposed for the delay-constrained multicast routing problem. The second part of the chapter deals with the the Routing and Wavelength Assignment of static multicast (light-tree) demands. Greedy algorithms based on bin packing, which incorporate the GRASP algorithm proposed for multicast routing, are presented. In Chapter 6, virtual topology design is investigated and several objectives are considered. A new objective criterion which aims at improving the connectivity of the virtual topology to help postpone the need for reconfiguration is proposed. Presented are various algorithms for virtual topology design. First, an approach based on solving the LP-relaxation of the formally defined optimization problem and rounding is presented. Greedy algorithms aimed at optimizing various objective criteria are then proposed, along with lower bounds. A discussion of conclusions and avenues of future research conclude the thesis.

Chapter 1

Introduction to Optical Networks

1.1 Optical Transmission

The basic building blocks of an optical transmission system are a transmitter, a transmission medium and a receiver. The transmitter converts data into a sequence of on/off light pulses which are transmitted over the transmission medium and converted back to the original data at the receiver. An optical transmitter is essentially a light source. The first generation systems in the 1970s used LEDs (*Light Emitting Diodes*) which are based on spontaneous emission. Today, almost all optical networks use lasers which use stimulated emission to produce high-powered beams of light. Regarding transmission media in optical communication, the most commonly used medium is optical fiber. Optical fiber is made primarily of silica (SiO_2) and acts as a waveguide, i.e. it serves as a path which allows the propagation of electromagnetic waves (e.g. light). Receivers or photodetectors are most commonly photodiodes which convert a stream of photons (the optical signal) into a stream of electrons (a photocurrent). This current is then amplified for further electronic processing.

Fiber optics is made possible as a result of the phenomenon of *total internal reflection* which depends on the refractive index of a material. The refractive index of a material is the ratio of the speed of light in vacuum to its speed in the respective material. When light travels from one material to another, the angle at which it is transmitted in the second material depends on the angle at which it approaches the boundary between the two and their corresponding refractive indices (see Fig 1.1). The relationship between these angles and the refractive indices is given by Snell's law which says that $n_1 \cdot \sin\theta_{inc} = n_2 \cdot \sin\theta_{ref}$, where θ_{inc} is the incidence angle of the ray of light as it approaches the boundary, θ_{ref} is the angle of refraction, i.e. the angle at which the light is transmitted in the second material, and n_1 and n_2 are the corresponding refractive indices of the materials. Supposing the second material

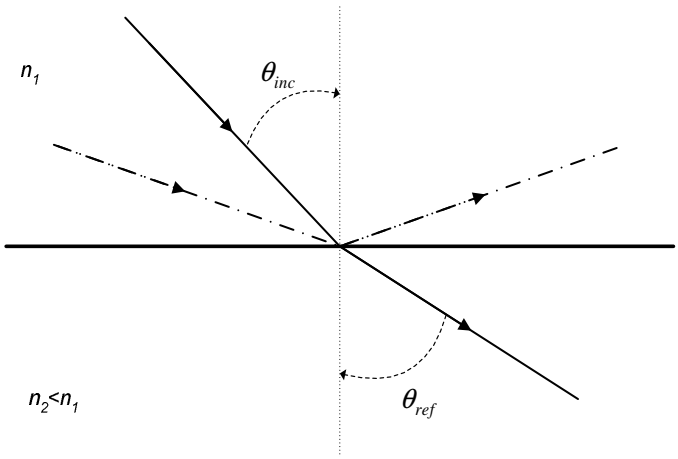


Figure 1.1: Snell's Law of Refraction.

has a smaller refractive index than the first (i.e. $n_2 < n_1$), if a ray of light approaches the boundary at an angle greater than a ceratin angle, called the *critical angle*, the ray of light is totaly reflected back into the first material. The critical angle is the incidence angle at which the ray of light is refracted at an angle of 90 degrees, i.e. right along the boundary, which according to Snell's law is equal to $\sin^{-1} \frac{n_2}{n_1}$.

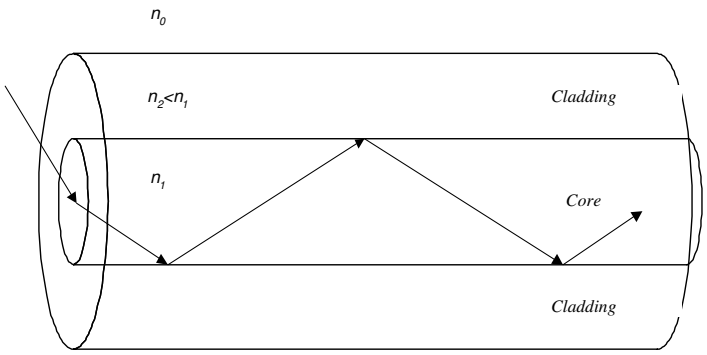


Figure 1.2: The propagation of a ray of light through an optical fiber.

An optical fiber consists of a cylindrical core made of glass which is completely encased in cladding with a smaller refractive index than the core. As a result, if light is transmitted in the core at an angle greater than the critical angle, the ray of light is totaly reflected within the core and thus the light is guided and can propagate through the fiber (see Fig. 1.2). Unfortunately, several transmission impairments affect the propagation of light. These include attenuation, dispersion and various non-linear effects.

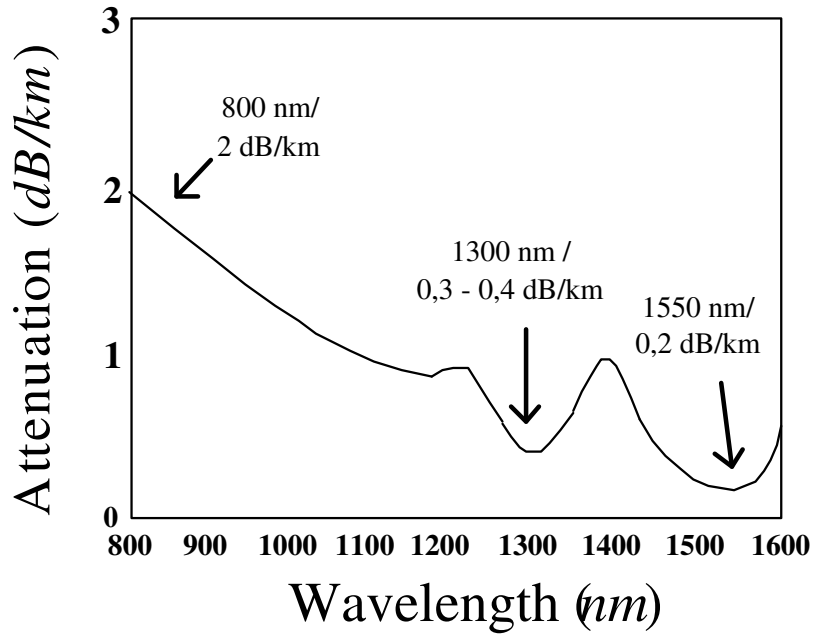


Figure 1.3: Attenuation in a standard single mode fiber (SMF) as a function of the wavelength used.

Attenuation Attenuation refers to the loss of signal power as it propagates through the fiber. This loss of power is primarily due to material absorption, Rayleigh scattering and waveguide imperfections [91]. Material absorption primarily occurs because of the impurities in the fiber glass which are mainly OH^- ions (i.e. water vapor). Raleigh scattering is caused by fluctuations in the refractive index of the fiber since it is not absolutely uniform and thus causes light to scatter. Waveguide imperfections occur since the fiber is not geometrically ideal.

Attenuation, denoted as α_{dB}, is usually expressed in terms of decibels per kilometer according to the following expression $\alpha_{dB} = \frac{-10}{l} \log_{10} \frac{P_r}{P_t}$, where the optical signal is transmitted with power P_t and arrives at the receiving end of a fiber span of length l with power P_r. Attenuation increases with the length of the fiber span l, but also depends on the wavelength of the optical signal. There are 3 low-attenuation regions in the attenuation graphic shown in Fig. 1.3. These low-loss wavelength bands, also called windows, are centered at 850 nm, 1300 nm and 1550 nm with power losses of approximately 2 dB, 0.4 dB and 0.2 dB, respectively. Transmission in the first window ranging from 600 nm to 900 nm was used in early optical networks in the 1970s with speeds of tens of megabits per second. The second window ranging from 1240 nm to 1340 nm is used in local, high speed applications, while the third window is used

for long-haul transmission networks. These networks operate at speeds of gigabits per second. A fourth window with low attenuation also exists ranging from 2500 to 2600 nm, however this window has not yet been exploited. To enable transmission using a wider wavelength band, ZWPFs (*Zero Water Peak Fiber*) have been developed which flatten the peak in attenuation between the second and third low-loss windows. However, a large amount of existing fiber from networks deployed years ago is still being used. Thus, instead of laying down new fiber, regenerators are used between fiber spans to restore the signal and minimize attenuation.

Dispersion As an optical pulse travels through fiber, the pulse broadens. This phenomenon is known as dispersion. Dispersion becomes a problem when the pulse spreads out to the extent that it overlaps with the preceding and/or successive pulse (see Fig. 1.4). This can lead to the misinterpretation of certain bits in the data sequence and thus increase the BER (*Bit Error Rate*). This problem, referred to as *inter-symbol interference*, limits the signal transmission rate, i.e. limits the minimum time interval between two consecutive pulses of light.

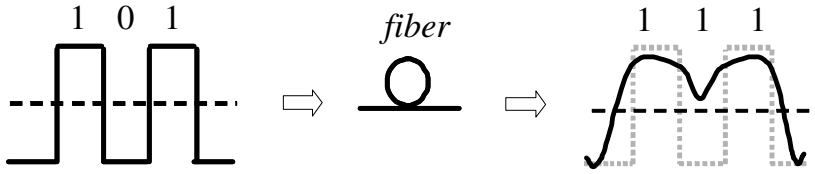


Figure 1.4: An example of the effects of dispersion on the optical signal.

The three main types of dispersion are *intermodal dispersion*, *chromatic dispersion* and *polarization mode dispersion*. Intermodal dispersion occurs in multi-mode fiber, where the core of the fiber has a diameter of approximately $50\mu m$, allowing light to propagate in different modes. Since these modes can be reflected internally at different angles of incidence at the core-cladding boundary, they propagate along different trajectories. As a result, modes arrive at the receiving end with different propagation delays broadening the pulse. One way to overcome this type of dispersion is to is to use fibers with a graded refractive index. In such fibers, the refractive index of the core in the region closer to the core-cladding boundary is such that the signal travels faster in that area. As a result, modes transmitted with a smaller incidence angle which travel along a longer trajectory (i.e. spend more time close to the core-cladding boundary) travel faster and thus arrive at the receiving end at the same time as modes travelling

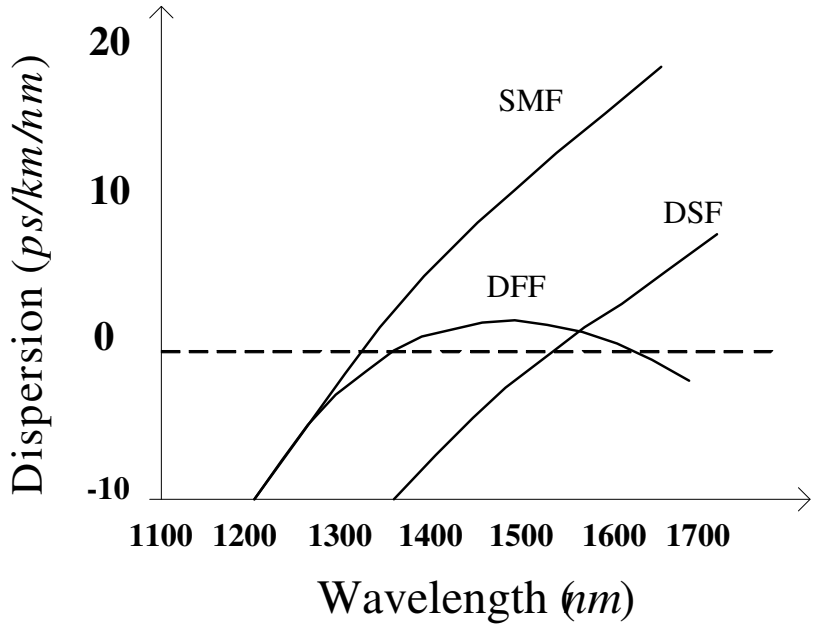


Figure 1.5: Dispersion as a function of wavelength for SMF, DFF and DSF fibers.

on shorter trajectories at slower speeds.

In single-mode fibers which are most commonly used in modern long-haul optical networks, the core has a diameter of only $8 - 10\mu m$ and thus light travels in only a single fundamental mode. As a result, there is no intermodal dispersion. However, chromatic dispersion occurs due to the fact that a transmitter cannot send all the photons of an optical signal on exactly one wavelength. Since the refractive index of a material is a function of the wavelength, some wavelengths will propagate faster than others. This type of chromatic dispersion is referred to as material dispersion. Another source of chromatic dispersion, called waveguide dispersion, refers to the differences in propagation of different wavelengths depending on the differences in the refractive indices and shape of the core and cladding.

The third type of dispersion is called polarization mode dispersion which also occurs in single-mode fibers. Namely, the fundamental mode of a transmitted signal is actually composed of two orthogonal polarization modes. As a result of the asymmetry of the fiber, these modes can propagate at different speeds. Consequently, they arrive at different times at the receiving end and thus broaden the signal.

Dispersion is dependant on the wavelength used. For Standard Single-mode Fibers (SMF), dispersion is negative for lower wavelengths, zero at around 1300 nm and

positive for higher wavelengths (see Fig. 1.5). In Dispersion Shifted Fibers (DSF), the fiber geometry is such that the zero dispersion point is shifted to 1550 nm and thus overlaps with the lowest attenuation wavelength band. In Dispersion Flattened Fibers (DFF), the dispersion is close to zero in the range from the second to the third low-attenuation window. Fig. 1.5 shows the dispersion profile of SMF, DSF and DFF fibers. Since very low dispersion can lead to non-linear effects as the optical power increases, Non-Zero Dispersion Fibers (NZ-DSF) have been developed which have low, but not zero, dispersion profiles in the third low-attenuation window. In addition to developing new fiber technology, a special pulse shape was discovered, called a *soliton*, which retains its shape as it propagates through the fiber and thus greatly reduces shape distortion.

Non-linear effects Non-linear effects become significant transmission impairments on long distance fiber spans and as bit rate or signal power increase. These effects include Self-Phase Modulation (SPM), Cross-Phase Modulation (XPM), Stimulated Raman Scattering (SRS), Stimulated Brillouin Scattering (SBS) and Four-Wave Mixing (FVM). These effects are out of the scope of this thesis, but a description of these phenomena can be found in [91].

As a result of these transmission impairments, fiber spans are limited to a few hundred kilometers for bit rates of gigabits per second. 3R regeneration (regenerate, reshape and retime) using repeaters requires optoelectronic conversion. In order to overcome this problem, optical amplifiers were developed. In the 1980's, Semiconducter Optical Amplifiers (SOA) were used but had a number of drawbacks. In 1987, Erbium-Doped Fiber Amplifiers (EDFA) first appeared and are still most commonly used today. They are made of silica fiber glass doped with ions of erbium which is a rare-earth metal. For further information on optical transmission refer to [64], [91], and [93].

1.2 WDM Optical Networks

1.2.1 WDM Technology

As a result of the growth of the Internet, the World Wide Web and several emerging multimedia network applications, there is an ever increasing demand for bandwidth in today's communication networks. Optical fiber is the preferred medium for backbone high-speed networks due to their huge bandwidth, low BER (*Bit Error Rate*) of 10^{-12} and low noise

and interference characteristics. However, although many high speed networks use optical fibers, some do not fully exploit their tremendous potential bandwidth (up to 50 THz) since at the end of each link, the optical signal is converted back to the electronic domain. As a result, data rates are limited to a few Gigabits per second due to end user equipment which is limited to peak electronic speed.

WDM (*Wavelength Division Multiplexing*) is a technology which efficiently overcomes this optical-electronic mismatch, referred to as the 'electronic bottleneck', by partitioning the fiber bandwidth into a set of disjoint wavelength bands. These non-overlapping wavelength bands (referred to as wavelengths) most commonly divide up the third low-attenuation loss region of the optical transmission spectrum (the 1550 nm band). Each wavelength supports one communication channel corresponding to an end user which can operate at an arbitrary speed, e.g. peak electronic speed. Since lightbeams on different wavelengths do not interfere with each other, multiple wavelengths (i.e. channels) can be combined (multiplexed) and simultaneously transmitted over an optical fiber. The individual channels are then separated (demultiplexed) at the receiving end to restore the individual data streams.

WDM is attractive since it is a cost-effective method of utilizing the huge potential bandwidth of fiber already deployed and thus eliminating the huge investment incurred by laying down additional fiber. Another key advantage of WDM technology is its transparency. Namely, each wavelength can carry data flows at different bit rates using different protocol formats.

Components used in optical WDM networks include transmitters, receivers, optical amplifiers and switching devices. As already mentioned, a transmitter is usually a laser, while a receiver is a filter or photodiode. Transmitters and receivers are commonly referred to as transceivers. Transceivers can be made dynamically tunable to operate on different wavelengths. In such cases, a transmitter is tuned to transmit on a certain wavelength, while a receiver can be dynamically tuned to receive the appropriate wavelength. A device integrating the functionality of a transmitter, receiver and electronic regenerator is referred to as a transponder [49]. To realize signal transmission, transponders modulate signals onto distinct wavelengths which are multiplexed, amplified and transmitted along the fiber. The signal is optically amplified about every 100 kilometers. Upon reaching the receiving end, the signal is pre-amplified, demultiplexed and restored to individual data streams.

Fiber interconnecting devices are critical to optical networking. In point-to-point transmission systems, such as SDH and SONET, the switching equipment used converts the signal from the optical into the electronic domain and back at every network node. Examples of electronic switches include Wavelength Add-Drop Multiplexers (WADM) and Digital Cross-

Connects (DXC). As optical networks are evolving, some routing, switching and intelligence is handled on the optical layer. Optical switches are those which can switch an optical signal without performing opto-electronic conversion, although the switch itself may be controlled by electronic signals. An Optical Add-Drop Mulitplexer (OADM), can forward certain wavelengths with no opto-electronic conversion, while letting data streams on other wavelengths which originate or terminate at the node be 'added' or 'dropped', respectively. Wavelength Routing Switches (WRS), also referred to as Wavelength Cross-Connects (WXC), can switch any wavelength on an incoming port to *any* output port in the optical domain. This means that the internal connections can be dynamically reconfigured. If the WRS has N input ports, N output ports and can support W wavelengths, it acts as W individual $N \times N$ switches. Wavelength routing nodes can also be equipped with Wavelength Converters (WC) which can convert the wavelength of a data stream at the router entirely in the optical domain. Further information regarding components of optical networks and enabling technologies can be found in [64], [91] and [93].

1.2.2 WDM Network Architectures

Broadcast-and-Select WDM Networks

In broadcast-and-select WDM networks, nodes are interconnected by a device called a passive star coupler [67]. Nodes are equipped with fixed or tunable transmitters which can transmit signals on different wavelengths. These signals are combined into a single signal by the passive star coupler. This device then transmits the combined signal to all the nodes in the network, where the power of transmitted signal is split equally among all the output ports leading to all nodes in the network. Each node can receive information sent on a certain wavelength by tuning their receiver to the desired wavelength. An example of a broadcast-and-select network is shown in Fig 1.6. Here, Nodes 1, 3, and 4 are transmitting data using wavelengths W_1, W_2 and W_3, respectively, while Node 2 is not transmitting any information. Node 4 is tuned to receive information carried on wavelength W_1, Nodes 1 and 2 are both tuned to receive information carried on wavelength W_2, while Node 3 is receiving information carried on wavelength W_3.

Broadcast-and-select networks can be single-hop [62] or multi-hop [63] networks. In single hop networks, transmitted messages travel from source to destination entirely in the optical domain. These networks require rapid tuning of transceivers and require synchronization protocols to coordinate the transmissions between various nodes. Multihop networks avoid rapid tuning by using fixed tuned transceivers which are semi-permanently configured

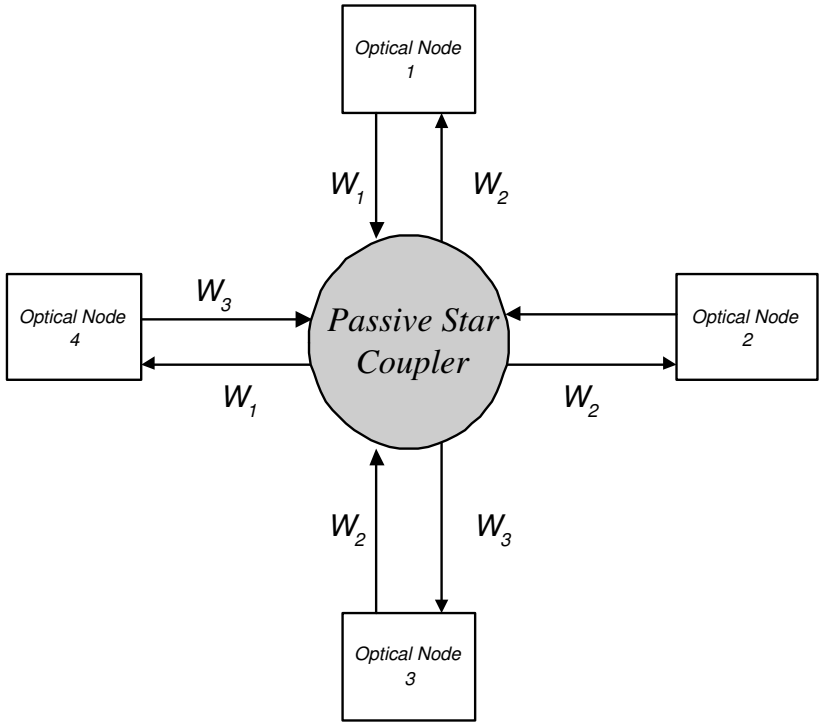


Figure 1.6: An example of a broadcast-and-select network.

to transmit or receive messages on a certain wavelength and thus form a virtual topology over the physical one.

An advantage of broadcast-and-select networks is that they can easily support multicast traffic since multiple receivers at different nodes can be tuned to receive the same wavelength channel. Drawbacks of broadcast-and-select networks include the need for synchronization and rapid tuning in single hop networks, the lack of wavelength reuse which creates the need for a large number of wavelengths, and they cannot span long distances since the signal power is split among various nodes. These networks are therefore mostly used in high-speed local area networks and metropolitan area networks.

Wavelength Routed Networks

Wavelength routed networks consist of nodes composed of wavelength routing switches (WRS) and end users. These nodes are interconnected by optical fiber links, forming an arbitrary physical topology. Configurable WDM nodes enable all-optical channels, called *lightpaths*, to be set up and torn down between pairs of nodes. Lightpaths can traverse multiple physical links and essentially create a virtual topology on top of the physical topology.

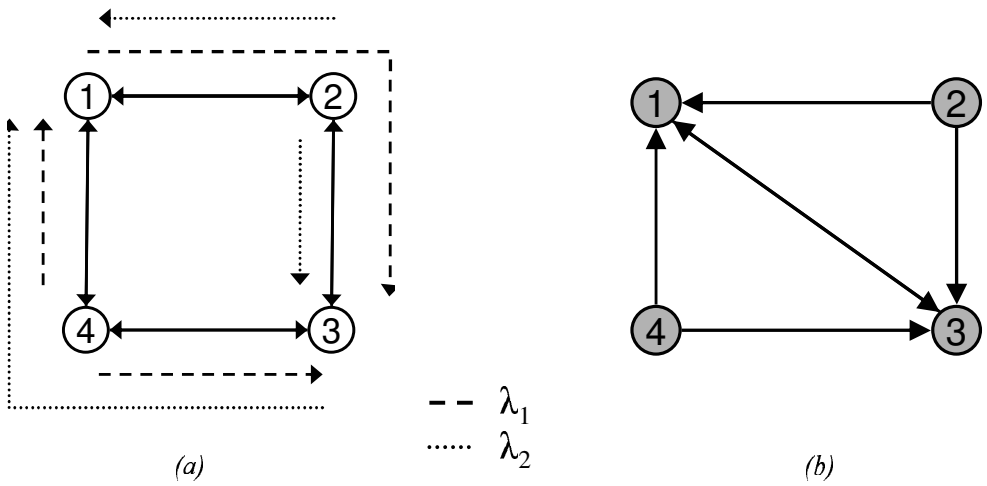


Figure 1.7: An example of (a) a wavelength routed network with a possible lightpath assignment and (b) its corresponding virtual topology .

Information sent via a lightpath does not require any opto-electronic conversion at intermediate nodes. End nodes of a lightpath tune their transmitters or receivers to the wavelength on which the lightpath operates in order to access the transmitted signal. Intermediate nodes forward the message in the optical domain using their WRSs. An example of a wavelength routed network and its corresponding virtual topology are shown in Fig. 1.7. Problems and issues regarding the design of wavelength routed networks will be discussed in detail in Chapter 2.

Wavelength routed networks have several advantages over broadcast-and-select networks. Since they do not split the power of an optical signal among multiple users, they have better scalability properties which are only limited by the physical characteristics of the optical medium. Furthermore, such networks take advantage of *wavelength reusability* which means that lightpaths which do not traverse the same physical links can operate on the same wavelength. This way, wavelength routed networks exploit both the optical capacity of the fiber using WDM technology *and* the capacity due to wavelength reuse. In addition, the management cost of wavelength routed networks is lower than that of broadcast-and-select networks since optical devices have lower maintenance costs. Another key advantage of wavelength routed networks is that they can be reconfigured on demand. Namely, lightpaths can be set up and torn down dynamically and thus enable the network to support changing traffic trends. Wavelength routing further enables the construction of transparent all-optical networks in which data can be transmitted over the same infrastructure regardless of the bit rate, traffic

characteristics or protocol used.

Linear Lightwave Networks

In linear lightwave networks, instead of directly multiplexing multiple wavelengths on an optical fiber, waveband partitioning is used [67]. This means that a set of wavelengths is grouped together into a waveband and then multiple wavebands are multiplexed onto the fiber. This type of two-tier partitioning, in which several wavelengths are treated as a single unit, lowers the network requirements. Namely, a waveband is transmitted and switched optically using a single port so the number of optical switches needed corresponds to the number of wavebands used and not individual wavelengths. Two important constraints in these types of networks are referred to as *inseparability* and *distinct source combining*. The first constraint says that once wavelengths of a single waveband are combined on the fiber, they cannot be separated in the network. The latter constraint forbids splitting the signal at one node and recombining it at another node in the network, i.e. only signals from distinct sources can be combined.

1.3 Next Generation Optical Networks

1.3.1 Optical Circuit, Burst, and Packet Switching

Currently, WDM technology is mainly being used to exploit the large bandwidth of optical fibers. Point-to-point technology, which converts the signal back into the electronic domain at the end of each link, is quite mature. However, optical switching is rapidly evolving. The evolution of optical networks is aimed towards the realization of a fully transparent *all-optical* network [80]. In such networks, a single infrastructure can carry data flows using different protocols, modulation schemes and/or bit rates. Another key development in optical networking is the design of reconfigurable optical networks in which bandwidth can be allocated dynamically between users according to changing traffic trends.

Photonic switching technology thus plays the main role in future optical networks. WDM technology is evolving from circuit to burst and packet switching technology. The main difference is in the granularity of the switching elements. The evolution of optical communications [61] is shown in Fig 1.8 and will take place in several phases. While point-to-point and ring networks employ electronic switching, wavelength routed networks are the first step in the gradual migration to all-optical switching. This phase is currently underway in high-speed transport networks. These networks are based on *optical circuit switching* (OCS) at

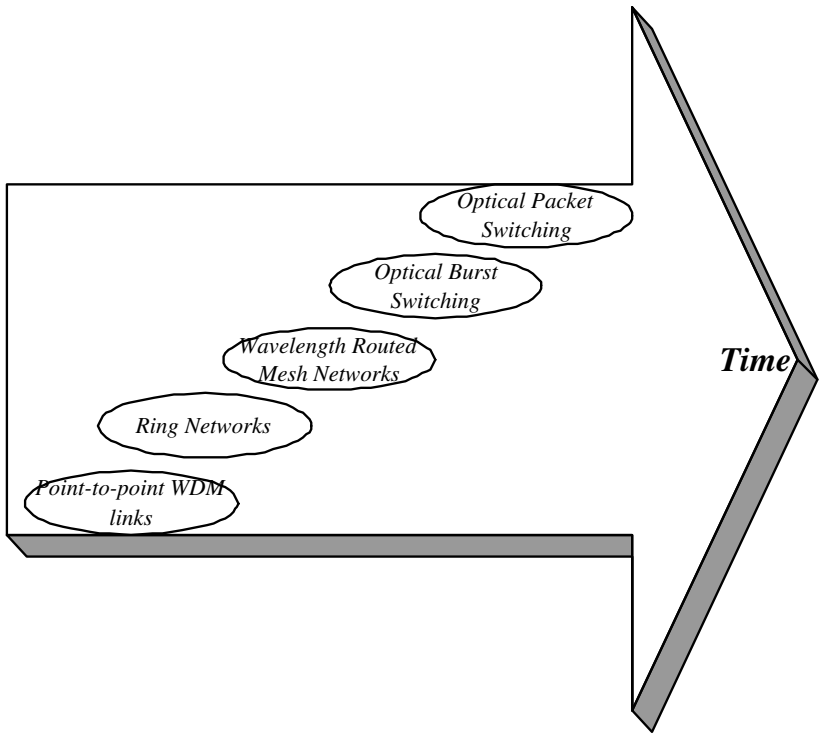


Figure 1.8: The evolution of optical communication systems.

the granularity of a wavelength. Lightpaths are established between pairs of nodes and data is transmitted entirely in the optical domain via lightpaths. Although this approach is a big step towards a transparent and configurable optical network, it has certain drawbacks [80]. One drawback is that channels are reserved regardless of whether data is being transmitted. Furthermore, to be efficient, traffic must be groomed or statistically multiplexed.

The future optical Internet will rely on photonic switches to transmit IP packets directly over WDM links eliminating intermediate layers between IP and WDM (e.g. ATM, SONET, SDH) [61], [23]. This avoids avoids excessive control and management overhead. However, a control plane is still necessary to ensure that IP packets arrive at their desired destinations. Photonic inspection or optical processing is currently technologically and economically infeasible, so the control of optical switches is processed electronically while the payload is switched optically with some optical buffering. Optical buffering is usually performed using Fiber Delay Lines (FDL) which delay the signal for a short time. In order to reduce header inspection duration, the concept of *optical burst switching* (OBS) was developed. OBS is aimed at improving the drawbacks of circuit switched networks and is meant to ease the transition from optical circuit to optical packet switching [61].

The basic switching entity in OBS networks is a variably sized *burst*. A burst is an aggregate flow of data packets with a single burst header. This is usually a group of IP packets heading for the same destination with the same QoS (*Quality of Service*) requirements. OBS is aimed to provide ultra-high throughput and real-time data provisioning. Grouping packets into bursts with a single burst header reduces header inspection duration and buffering at intermediate nodes. Control packets are transmitted on a separate wavelength with an offset time. Bandwidth is used only for the duration of the burst. Optical burst switching technology is still in the experimental stage and there are several challenges which need to be faced. The main challenges include designing fast and cost-efficient switches, burst switching protocols and signalling mechanisms, and wavelength channel scheduling [67]. For an overview of OBS refer to [97].

The final stage in the transition from circuit to packet switching leading to a ultra-high bandwidth, data-centric, flexible and transparent all-optical Internet is *Optical Packet Switching* (OPS) ([80], [96]). This technology is still in the experimental stage but is predicted to be the core of the next generation optical Internet. The basic switching entity is a *packet* promising arbitrarily fine switching granularity. Each packet is composed of a header and payload. At the optical switch, the payload is switched or buffered depending on the information in the header. The challenges involved in realizing OPS include the design of fast and cost-effective switches, packet synchronization, content resolution, and scalable buffering and packet level parsing.

It is important to note that optical circuit, burst and packet switching technologies all have different application domains. As a result, it is more likely that all three will coexist in the next generation optical network instead of replacing one another [80]. Since the control and transport planes will most likely be separated, label switching will probably be used for control in the optical domain for all three switching frameworks [61]. This technology is migrated from the existing IP/MPLS networks and will be discussed in the next section.

1.3.2 Control and Management: GMPLS

In wavelength routed (circuit switched) optical networks, control mechanisms are required to dynamically set up and tear down lightpaths [94]. Research in this field is primarily focused on developing control mechanisms which minimize the blocking probability of lightpath requests, set-up delay and control message overhead [65]. Several approaches available in literature include both centralized and distributed schemes [95]. In traditional telecommunication networks, network control is implemented as part of a layered management system. However, the approach adopted in IP networks separates control from management focusing

on the automation of provisioning and control [94]. The future optical Internet is focused on developing such an automated optical control plane.

One of the motivation factors for optical burst/packet switching is the convergence of electronic and optical technologies, i.e. of IP and OPS. Running IP directly over WDM is more efficient than using intermediate layers, but requires an optical control plane. As optical networking is evolving from SONET/SDH networks to high-speed mesh transport networks, a control plane is necessary which can support *both* legacy and new traffic types and which can operate across network boundaries [30]. In other words, flexible control supporting both circuit and packet switched traffic is needed.

The Internet Engineering Task Force (IETF) and other standardization bodies have been working on the development of protocols for the control plane of transport networks. The IETF introduced the first Internet Draft of a collection of protocols, called Generalized Multi Protocol Label Switching (GMPLS), in the year 2000 and has been updating it ever since. GMPLS [55] is a framework aimed at satisfying the need for an IP-oriented control plane in optical networks and can support a variety of services and incorporate traffic engineering capabilities.

GMPLS is a generalized version of MPLS, while adding features that enable it to work in optical circuit/burst/packet switched networks. Multi Protocol Label Switching (MPLS) [79] is a control framework developed to enable fast switching in IP networks. The idea behind MPLS is that labels are added to IP packets which are used to forward packets along Label Switched Paths (LSP) traversing multiple Label Switched Routers (LSR). The IP header of the packet is not examined at intermediate nodes along the LSP enabling fast switching. Packets with with common parameters (e.g. the same destination) are assigned to the same LSP.

This idea of tunnelling IP packets along Label Switched Paths is similar to the idea of tunnelling IP traffic via lightpaths [30]. The wavelength assigned to a particular lightpath is analogous to the labels used in electronic MPLS networks. Optical cross-connects in a wavelength routed network could maintain port and wavelength mappings similar to label forwarding tables at Label Switched Routers. Thus, the concepts of MPLS were extended giving rise to GMPLS which is aimed at providing a unified control layer for multi-layer transport networks and thus seamlessly interconnecting new and legacy networks. This frameworks should allow end-to-end provisioning and control and traffic engineering regardless as to whether the edge nodes belong to different networks. Such a multi-service framework would provide higher flexibility, lower operational cost and easier management of different services [17]. For more information on GMPLS refer to [30], [61] and [99], and

refer to [36] for the current IETF drafts and RFCs regarding the GMPLS architecture.

Chapter 2

Optimization Problems in Wavelength Routed WDM Networks

In wavelength routed optical networks, complex and diverse virtual topologies can be created over the same physical optical network by establishing various all-optical communication channels between nodes. Several optimization problems and issues arise in the design of such networks. This thesis focuses on the problems of Routing and Wavelength Assignment and Virtual Topology Design in wavelength routed WDM networks. A description of these problems and other issues related to wavelength routed networks follows.

2.1 Routing and Wavelength Assignment

Wavelength routed WDM networks are equipped with configurable WDM nodes which enable all-optical connections, called *lightpaths*, to be set up and torn down between pairs of nodes. Although these all-optical connections can traverse multiple physical links, information sent via a lightpath does not require any opto-electronic conversion at intermediate nodes. Establishing a set of lightpaths creates a virtual topology on top of the physical topology. The physical topology represents the physical interconnection of WDM nodes by actual fiber links in the WDM optical network. The links in the virtual topology represent all-optical connections or lightpaths established between pairs of nodes. Demands to set up lightpaths between certain nodes can be static, scheduled or dynamic. In the case of *static lightpath demands*, a desired static virtual topology (i.e. the set of lightpaths we wish to establish) is known *a priori*. Such a virtual topology is set up 'semi-permanently' which implies that even when a certain connection is not being used its resources remain reserved. When we refer to *scheduled lightpath demands*, we refer to connection requests for which

we know the set-up and tear-down times *a priori*. In other words, we know in advance when a connection will be needed. For *dynamic lightpath demands*, lightpath requests arrive unexpectedly with random holding times. These lightpaths are set up dynamically at request arrival time and are released when the connection is terminated.

To set up a lightpath, nodes on its corresponding physical path must be configured to do so. If the network lacks wavelength converters, the lightpath must use the same wavelength along its entire physical path. This is called the *wavelength continuity constraint*. Two lightpaths that share a common physical link cannot be assigned the same wavelength. This is called the *wavelength clash constraint*. The source and destination nodes of the lightpath must also have available transmitters and receivers respectively in order for the lightpath to be successfully set up.

Determining routes for a set of lightpath demands and assigning wavelengths to these routes subject to a subset of the above mentioned constraints is known as the Routing and Wavelength Assignment (RWA) problem. This problem has been proven to be NP-complete [11] and several heuristic algorithms have been developed to solve it suboptimally [4] [12] [57] [71]. Several variations of the RWA problem have been studied [39] [64] [67] such as the routing and wavelength assignment of static, scheduled or dynamic lightpath demands with a limited or unlimited number of wavelengths in networks with wavelength converters at each node, at a subset of nodes, or in networks with no wavelength converters. Algorithms to solve these problems can be centralized or distributed. The objective is often to minimize the number of wavelengths used or to maximize the number of lightpaths set up subject to a limited number of wavelengths. A second objective can be to minimize the physical lengths of the corresponding physical paths. Examples of heuristic algorithms which solve the RWA problem for static, scheduled, and dynamic lightpath demands can be found in [86], [85] and [13], respectively. In this thesis, we will consider the problems of Static and Scheduled Routing and Wavelength Assignment in Chapters 3 and 4, respectively.

2.2 Multicast Routing and Wavelength Assignment

Due to the rapid development of advanced network services and applications, network traffic has been growing exponentially in the past several years. As already mentioned, WDM technology is a promising solution for satisfying the ever increasing capacity requirements in telecommunication networks. In addition to high capacity requirements, the development of several multimedia applications has created an increasing need for point-to-multipoint and multipoint-to-multipoint communication. This type of communication is referred to as

multicasting. WDM optical networks can efficiently support multicasting since splitting light is inherently easier than copying data into an electronic buffer. Applications of multicasting include multimedia conferencing, distance education, video distribution, distributed games and many others. Many of these applications require packets of information to be sent with a certain Quality of Service (QoS). In this thesis, we will consider the QoS demand of a bounded end-to-end delay. This constraint is particularly important for real-time applications such as video-conferencing or distance education.

Recall that in wavelength routed WDM networks, a virtual topology is established over the physical optical network by setting up all-optical connections, called *lightpaths*, between pairs of nodes. A generalization of a lightpath, called a *light-tree*, was proposed in [81] to facilitate multicasting in wavelength routed WDM optical networks. Light-trees logically connect a source node to a group of destination nodes. Thus, these trees enable all-optical point-to-multipoint communication, i.e. entirely in the optical domain. Light-tree demands, like lightpath demands, can be static, scheduled or dynamic. A survey of optical multicasting in WDM networks is given in [22]. To establish a light-tree, nodes in the network must be equipped with multicast capable switches [47] which increases network cost. However, it has been shown that setting up a virtual topology composed of a set of light-trees, as opposed to lightpaths, substantially reduces the average packet hop distance and the number of transceivers required in a network for unicast, multicast *and* broadcast traffic [81].

To establish a virtual topology composed of a set of *light-trees*, we must solve the *Multicast Routing and Wavelength Assignment* (MC_RWA) problem. In this thesis, we consider the *static* MC_RWA problem, i.e. all the the requests are known *a priori* and the virtual topology is established 'semi-permanently'. Given is a network and a set of multicast requests. For each multicast request, it is necessary to find a multicast tree, i.e. a light-tree, which connects the source node to all the destination nodes. Optimization problems in multicast tree construction are discussed in [70]. Multicast routing is often reduced to the minimum Steiner tree problem in graphs. Generally, for a given graph $G = (V, E)$, where V is a set of nodes and E is a set of edges, and a given subset of nodes, $D \subseteq V$, a Steiner tree is one which connects all the nodes in D using a subset of edges in E. This tree may or may not include nodes in $V \setminus D$. A *minimum* Steiner tree is such a tree which is of minimum weight in a weighted graph. Several applications of Steiner Trees can be found in [10].

In addition to finding a feasible multicast tree, in order to solve the Multicast Routing and Wavelength Assignment problem it is necessary to assign wavelengths to these trees subject to the following constraints. If no wavelength converters are available, the same wavelength must be assigned along the entire tree. This is called the *wavelength continuity*

constraint. In addition, light-trees that share a common physical link cannot be assigned the same wavelength. This is called the *wavelength clash constraint*. Depending on the power splitters available at each node, the optical signal may only be split into a bounded number of signals [2]. This imposes a constraint on the degree of the established multicast trees. Delay constrained multicasting, where each multicast request has an end-to-end delay bound associated with it, can also be considered. The objective of the MC_RWA problem is often to minimize the number of wavelengths used, or to maximize the number of light-trees successfully set up subject to a limited number of wavelengths. A second objective which can be considered is the minimization of the costs of the established light-trees. The cost of a light-tree can represent various values such as the actual cost, the total number of hops, the total length or the maximum transmission delay in the tree.

The problem of Routing and Wavelength Assignment of unicast (lightpath) demands has been shown to be NP-complete [11]. Multicast (light-tree) demands make the problem even harder. In fact, multicast routing, i.e. the minimum Steiner tree problem, itself is NP-hard [27]. Several variations of the MC_RWA problem and their solutions have been proposed in [8], [33], [34], [40], and [84].

2.3 Virtual Topology Design

The Virtual Topology Design problem in wavelength routed WDM networks is as follows. Given is a physical topology, the number of available wavelengths on each link, the number of available transmitters and receivers at each node, and a traffic matrix representing the long-term average traffic flows between nodes. To create a virtual topology, a set of lightpaths (and/or light-trees) which forms the virtual topology over the physical one must be determined. In this thesis, we will consider the design of virtual topologies consisting only of lightpaths. In order for these lightpaths to be established, each lightpath must be routed over the physical topology and assigned a particular wavelength (i.e. the RWA problem). We will refer to a combination of these subproblems as the *V*irtual topology and *R*outing and *W*avelength *A*ssignment problem ($VRWA$). Lastly, packet switched traffic must be routed over the established virtual topology. This will be referred to as *T*raffic *R*outing (TR). The Virtual Topology Design problem refers to a combination of the *VRWA* and *TR* problems.

To establish a virtual topology, a number of constraints must be satisfied. These include the wavelength continuity and clash constraints included in the RWA problem. Furthermore, the source and destination nodes of a lightpath must have available transmitters and receivers, respectively, in order for the lightpath to be successfully established. Due to a limited number

of transmitters and receivers at each node and a limited number of available wavelengths on each link, it is usually not possible to set up a lightpath between every pair of nodes. There are several objectives which can be considered in virtual topology design which include minimizing congestion, average packet hop distance, the number of wavelengths used, etc. A detailed description of these objectives will be discussed in Chapter 6.

Determining a good virtual topology with respect to various optimization criteria given a limited amount of resources is a complex problem. Several variations of the virtual topology problem have been studied [39], [24], [64], [67]. These include designing regular [68] and arbitrary ([5], [48], [66], [76], [100], [88], [73], [50]) virtual topologies.

2.4 Other Issues

2.4.1 Virtual Topology Reconfiguration

The virtual topology is most often designed to perform well for a given traffic matrix which represents the estimated long-term average traffic flows between pairs of nodes. However, traffic is prone to change and thus the established virtual topology may not preform well for changing traffic patterns. The advantage of wavelength routed networks lies in the fact that wavelength routed switches are reconfigurable and thus the virtual topology can be changed to meet the changing traffic demands. This is done by establishing new lightpaths and/or tearing down existing lightpaths. However, reconfiguration is expensive since it incurs service disruption and control overhead. As a result, reconfiguration is often kept to a minimum. Research in this field is focused on finding new virtual topologies which are as close to the current virtual topology as possible and yet improve performance measures. Related work can be found in [6] and [75]. An approach to virtual topology reconfiguration without assuming the future traffic pattern is known is given in [28].

2.4.2 Survivability

Survivability and fault management are also important issues in wavelength routed networks. Mechanisms must be developed which deal with component failures. It is crucial in wavelength routed networks that failure recovery be fast in order to minimize service disruption since there is a huge amount of traffic being carried. Handling failures at the optical layer, as opposed to the client layer, has the following advantages. Namely, recovery is faster at the lightpath level and the number of failed lightpaths is usually much smaller than the number of failed user connections allowing restoration to be performed with less overhead

[67]. Two common approaches in literature dealing with component failures in wavelength routed networks are as follows. The first approach is to define secondary paths and preassign wavelengths to them. This enables almost instant recovery but at the cost of reserving excessive resources. The second approach is to deal with failures as they arrive by dynamically searching for backup paths. This, however, incurs a longer recovery delay and provides no guarantee that there is a backup path available. The objective criteria for designing survivable networks and efficient fault management mechanisms include resource requirements, connection acceptance rates and failure recovery times [67]. Related work can be found in [60], [101], [61], and [99].

2.4.3 Traffic Grooming

Connection requests in wavelength routed optical networks often require less capacity than is available on a single lightpath. As a result, efficient traffic grooming which combines low-rate (sub-wavelength) traffic onto a single wavelength, i.e. a higher capacity lightpath, can drastically improve network throughput and reduce network cost. Several approaches have been studied, but the majority of research is focused on traffic grooming in ring networks. Traffic grooming in WDM mesh networks with the objective to maximize network throughput is studied in [106]. An approach which solves traffic grooming together with routing and wavelength assignment is proposed in [98].

2.4.4 Control and Management

As already mention in Chapter 1.3.2, a control mechanism is needed in wavelength routed networks which can dynamically establish and tear down lightpaths. In order to establish a lightpath, a dynamic routing and wavelength assignment algorithm is used to determine the physical path and wavelength assigned to the lightpath. The network control mechanism must have information about lightpaths and wavelengths currently being used in the network in order to perform dynamic RWA. Once RWA is solved, the corresponding wavelength routed switches must be appropriately configured to establish the lightpath. Several approaches have been studied in literature which fall into two categories: centralized and distributed ([95], [65]). Common objectives when developing an efficient control mechanism include maximizing the number of connections established (throughput), minimizing connection set-up times and minimizing control overhead . Besides the mentioned control functions, a network management system is needed for performance management, fault management, security management and accounting management [67]. For more on control and

management refer to [99] and [61].

2.4.5 Transmission Impairments

Transmission impairments of the physical layer are often ignored when designing virtual topologies and performing RWA. However, taking these impairments into consideration in the planning process can reduce network cost and improve the performance characteristics. Since transmission distance is fairly long in wide-area wavelength routed optical networks, physical limitations should be taken into consideration to ensure a low BER (*Bit Error Rate*). The BER of a connection depends on a large number of transmission impairments including attenuation, dispersion and non-linear effects. Developing routing and/or wavelength assignment schemes with target BER levels (i.e. a BER lower than a certain threshold value) or minimizing the BER level could help improve network performance [21]. An approach to virtual topology design considering physical limitations is given in [98]. Another optimization problem regarding transmission impairments in wavelength routed networks is minimizing the number of optical amplifiers needed and determining their placement in the network subject to device limitations. These limitations include the maximum output power and the maximum gain of amplifiers, the maximum power of transmitters, the minimum signal power needed for wavelength detection, and attenuation in the fiber [67]. Amplifier placement with the objective of minimizing the total amplifier cost is investigated in [104].

2.4.6 Wavelength Convertible Networks

In wavelength convertible networks, nodes in the network are equipped with wavelength convertible switches which enable a signal on an particular input port to be switched to an output port on a different wavelength without performing opto-electronic conversion [74]. The wavelength continuity constraint is dropped when solving the virtual topology design and routing and wavelength assignment problems in such networks. Since wavelength converters are fairly expensive, sparse wavelength conversion, i.e. where converters are only placed at a subset of the nodes in the network, is often considered. This gives rise to the optimization problem of converter placement. For more on RWA with wavelength conversion and converter placement refer to [16], [15], [92] and [14].

Chapter 3

Static Routing and Wavelength Assignment

In this chapter we consider the problem of *Static Routing and Wavelength Assignment* in wavelength routed networks with no wavelength conversion. We improve upon solutions proposed for the routing and wavelength assignment of static lightpath requests by efficiently applying bin packing algorithms. Bin packing is a classical NP-hard optimization problem [27] which finds its application in many real world problems, such as truck loading, stock-cutting problems, storage allocation for computer networks, the problem of packing commercials into breaks and many others. However, the potential of this model has not yet been systematically explored in the context of the routing and wavelength assignment problem.

We apply bin packing to develop very efficient - yet simple - heuristic algorithms for the RWA problem with the objective to minimize the number of wavelengths used. We also consider a second objective, which is to minimize the physical lengths of the established lightpaths. The motivation for these objectives is as follows. Minimizing the number of wavelengths is desirable in order to leave more room for future expansion of the virtual topology. Minimizing the physical length of a lightpath, not only in terms of hops, but also in terms of actual distance, is desirable in all WDM networks due to signal degradation and propagation delay.

The algorithms were tested on large random networks and compared with an efficient RWA algorithm presented in [57]. Results indicate that the proposed algorithms obtain solutions which, not only use significantly fewer wavelengths, but which also establish shorter lightpaths. The obtained solutions were also compared with analytical lower bounds. For denser networks, the proposed algorithms obtained optimal or near optimal solutions with

respect to both wavelengths and lightpath lengths in many cases. Furthermore, the speed and simplicity of the algorithms make them highly tractable for large networks with many lightpath requests.

In the next section, we informally define the RWA problem, and discuss related work in Section 3.2. In Section 3.3 we introduce classical bin packing and suggest heuristic algorithms for the RWA problem. Lower bounds are briefly discussed in Section 3.4. Numerical results and a chapter summary are given in Sections 3.5 and 3.6, respectively.

3.1 Problem Definition

The physical optical network is modelled as a graph $G = (V, E_p)$, where V is the set of nodes and E_p is the set of physical edges. Edges are assumed to be bidirectional, each representing a pair of optical fibers (i.e. one fiber per direction). Given is a set of lightpath requests $\tau = \{(s_1, d_1), \ldots, (s_n, d_n)\}$, where $s_i, d_i \in V, i = 1, \ldots, n$. Each lightpath request (s_i, d_i) in G is defined by its source node s_i and destination node d_i. The static Routing and Wavelength Assignment problem searches for a set of directed paths $P = \{P_1, \ldots, P_n\}$ in G, each corresponding to one lightpath request, and assigns wavelengths to these paths. Paths P_i and P_j where $i \neq j$, $i, j = 1, \ldots, n$, cannot be assigned the same wavelength if they share a common directed edge. The length[1] of path P_i, $i = 1, \ldots, n$, denoted as $l(P_i)$, can be upper bounded by a value H. The objective is to minimize the number of wavelengths required to successfully route and assign wavelengths to all the lightpath requests in τ. We also consider a second objective which is to minimize the average physical length of the established lightpaths, i.e. $min \frac{\sum_{j=1}^{n} l(P_j)}{n}$.

3.2 Related Work

Most approaches used to solve the RWA problem in WDM optical networks decompose the problem into two subproblems, routing and wavelength assignment, solved subsequently. A classification of such RWA algorithms can be found in [12]. In [4], the authors use a multicommodity flow formulation and randomized rounding to solve the routing subproblem. Wavelength assignment is solved using graph coloring heuristics. In [35], the authors use local random search for route selection. For each routing scheme, the wavelength assignment problem is solved using a greedy graph coloring algorithm. A generalization of the graph coloring problem, called the partition coloring problem, and its application to routing

[1] Length can be considered in terms of the number of hops or actual distance.

and wavelength assignment is studied in [54]. In [69], wavelength assignment of previously calculated alternative paths is solved using a tabu search algorithm suggested for partition coloring.

An algorithm which solves the routing and wavelength assignment subproblems simultaneously is suggested in [53]. Here, the authors present an integer formulation and a column generation technique to help solve it. This approach may not be practical for larger problems. A fast yet effective greedy algorithm based on edge disjoint path (EDP) algorithms is presented in [57]. The algorithm, called *Greedy_EDP_RWA*, creates a partition $\tau_1, \ldots, \tau_k$ of the set of lightpath requests τ. Each element of the partition is composed of a subset of lightpath requests which can be routed on mutually edge disjoint paths in G and, hence, can be assigned the same wavelength. The number of distinct wavelengths needed to successfully perform RWA corresponds to the number of elements in the partition. This algorithm is very simple and fast and yet was shown to outperform the algorithm presented in [4]. The algorithm in [69] was shown to perform the same or slightly better than the multistart *Greedy_EDP_RWA* algorithm with respect to the number of wavelengths used, after 10 minutes of computational time, for networks with the number of nodes ranging from 14 to 32.

3.3 Heuristic Algorithms for RWA Using a Bin Packing Approach

3.3.1 Bin Packing

The bin packing problem is a classical combinatorial optimization problem that has been widely studied in literature. Given is a list of n items of various sizes and identical bins of limited capacity. To solve the bin packing problem, it is necessary to pack these items into the minimum number of bins, without violating the capacity constraints, so that all items are packed. Since this problem is NP-hard [27], a vast array of approximation algorithms have been proposed and studied. Surveys of bin packing algorithms can be found in [19] and [18]. A more recent heuristic algorithm is suggested in [3].

Four well-known classical bin packing algorithms are the First Fit (FF), Best Fit (BF), First Fit Decreasing (FFD) and Best Fit Decreasing (BFD) algorithms. The FF and BF algorithms are so-called on-line bin packing algorithms, which means that they pack items into bins in random order with no information of subsequent items. Both algorithms label bins in sequential order as new bins are used. The FF algorithm packs each item into the first

bin (i.e. the bin with the lowest index) into which it fits. The BF algorithm packs each item into the bin which leaves the least room left over after packing the item.

The FFD and BFD algorithms are two very fast and well known off-line bin packing algorithms. This means that they have information of all the items to be packed *a priori*. Having this information, it seems logical to first place larger items into bins and then fill up the remaining space with smaller items. On the contrary, if all the small items are neatly packed into one bin, there is a great chance that none of the large items will fit into that bin. Moreover, each larger item may need an extra bin of its own leaving a lot of unused space around it and ultimately leading to a larger number of bins used. The FFD and BFD algorithms apply this concept by sorting the given items in non-increasing order of their corresponding sizes, and then perform packing in the same manner as the FF and BF algorithms, respectively. These algorithms perform significantly better than FF and BF.

3.3.2 Algorithms for the RWA Problem

To apply bin packing to the Routing and Wavelength Assignment problem we must define items, bins, and their corresponding sizes in terms of optical networks. We consider lightpath requests to represent items, while copies of graph G represent bins. Each copy of G, referred to as bin $G_i, i = 1, 2, 3, \ldots$, corresponds to one wavelength. We consider the size of each lightpath $(s_j, d_j) \in \tau$ to be the length of its shortest path SP_j in graph G. However, it is important to note that lightpaths are not necessarily routed on their shortest paths. This measure is used only by the FFD and BFD algorithms in order to sort the 'items' or lightpaths in non-increasing order of their corresponding sizes.

The capacity of each bin is limited by the edges in G. Namely, two lightpaths routed on the same copy of G cannot traverse any of the same edges due to the *wavelength clash constraint*. To solve the RWA problem, we wish to pack as many items (lightpaths) into a minimum number of bins (copies of G), and therefore minimize the number of wavelengths used. In doing so, we must also take care to satisfy the wavelength continuity and clash constraints. Herein, we propose RWA algorithms, to be referred to as FF_RWA, BF_RWA, FFD_RWA and BFD_RWA, which are respectively based on classical bin packing algorithms FF, BF, FFD and BFD. The FF_RWA algorithm obtains solutions equivalent to those obtained by the *Greedy_EDP_RWA* algorithm suggested in [57], while the remaining algorithms perform significantly better.

FF_RWA (***Greedy_EDP_RWA*** [57])

The First Fit bin packing algorithm modified to solve the Routing and Wavelength Assignment problem, referred to as FF_RWA, runs as follows. First, only one copy of G, bin G_1, is created. Higher indexed bins are created as needed. Lightpath requests (s_j, d_j) are selected at random and routed on the lowest indexed copy of G in which there is room. Bin G_i is considered to have room for lightpath (s_j, d_j) if the length of the shortest path from s_j to d_j in G_i, denoted as P_j^i, is less than H. If a lightpath is routed in bin G_i, the lightpath is assigned wavelength i and the edges along path P_j^i are deleted from G_i. If all the edges from bin G_i are deleted, the bin no longer needs to be considered. If no existing bin can accommodate lightpath request (s_j, d_j), a new bin is created.

The FF_RWA algorithm is similar to the *Greedy_EDP_RWA* algorithm suggested in [57]. The difference is in the order in which some steps are executed. Namely, the FF_RWA algorithm routes each lightpath on the first copy of G it fits in. If all the existing bins are full, a new bin is created. The *Greedy_EDP_RWA* algorithm, on the other hand, creates only one copy of G at a time, and then tries to route as many lightpaths as possible on that copy. Due to its basic equivalency with FF_RWA, we will compare the *Greedy_EDP_RWA* from [57] with the rest of the bin packing algorithms proposed in this thesis.

BF_RWA

The Best Fit Routing and Wavelength Assignment algorithm, BF_RWA, routes lightpaths in the bin into which they fit 'best'. The BF_RWA algorithm defines the 'best fit' quite differently than the BF bin packing algorithm. Namely, in classical bin packing, the 'best fitting' bin is considered to be the one in which there remains the least empty space after packing an item. The BF_RWA algorithm, on the other hand, considers the best bin to be the one in which the lightpath can be routed on the shortest path. In other words, if at some point in running the algorithm, there are B bins created, bin G_i, $1 \leq i \leq B$, is considered to be the best bin for lightpath (s_j, d_j) if $l(P_j^i) \leq l(P_j^k)$, for all $k = 1, \ldots, B$, and $k \neq i$. This is not necessarily the overall shortest path, SP_j, since it is possible that none of the existing bins have this path available. If there is no satisfactory path available in any of the B bins (i.e. $l(P_j^i) > H$, for $i = 1, \ldots, B$), a new bin is created.

The motivation for the 'best fit' approach described above, is not only to use less wavelengths, but also to minimize the physical length of the established lightpaths. Of course, we could route each lightpath (s_j, d_j) strictly on its shortest path SP_j, but this would in most cases lead to a larger number of bins, which in turn means using a larger number of

FF_RWA (FFD_RWA)
Input:
$G = (V, E_p)$;*//physical network*
$\tau = \{(s_1, d_1), \ldots, (s_n, d_n)\}$; *//lightpath requests*
H; *//max physical length of lightpath*
Begin:
(ONLY FOR **FFD_RWA**: Sort and renumerate demands τ in non-increasing order of their shortest paths, SP_j, in G)
$P = \{\}$; *//The final paths*
Create 1 copy (bin) of $G : G_1$;
$BINS := \{G_1\}$;
while τ is not empty **do**
  for $j = 1$ to $|\tau|$ **do**
    $P_j = \emptyset$;
    for $i = 1$ to $|BINS|$ **do**
      Find shortest path, P_j^i, for lightpath (s_j, d_j) in G_i;
      if $l(P_j^i) \leq H$ **then**
        $P_j = P_j^i$;
        Assign wavelength i to path P_j;
        Delete edges in P_j^i from G_i;
        $i = |BINS|$;
      end if;
    end for;
    if $P_j = \emptyset$ **then**
      $New := |BINS| + 1$;
      Create copy of G: G_{New};
      $BINS := BINS \cup \{G_{New}\}$;
      Find shortest path, P_j^{New}, for lightpath (s_j, d_j) in G_{New};
      $P_j = P_j^{New}$;
      Assign wavelength New to path P_j;
      Delete edges in P_j^{New} from G_{New}
    end if;
    $P = P \cup P_j$;
    $\tau = \tau \setminus (s_j, d_j)$;
  end for;
end while;
return P;
End

BF_RWA (BFD_RWA)
Input:
$G = (V, E_p)$;*//physical network*
$\tau = \{(s_1, d_1), \ldots, (s_n, d_n)\}$; *//lightpath requests*
H; *//max physical length of lightpath*
Begin:
(ONLY FOR **BFD_RWA**: Sort and renumerate demands τ in non-increasing order of their shortest paths, SP_j, in G)
$P = \{\}$; *//The final paths*
Create 1 copy (bin) of $G : G_1$;
$BINS := \{G_1\}$;
while τ is not empty **do**
  for $j = 1$ to $|\tau|$ **do**
    $P_j = \emptyset$, $l(P_j) = \infty$;
    $BestBin := 0$;
    for $i = 1$ to $|BINS|$ **do**
      Find shortest path, P_j^i, for lightpath (s_j, d_j) in G_i;
      if $l(P_j^i) \leq H$ and $l(P_j^i) < l(P_j)$ **then**
        $BestBin = i$;
        $P_j = P_j^i$;
        Assign wavelength i to path P_j;
      end if;
    end for;
    if $P_j \neq \emptyset$ **then**
      Delete edges in $P_j^{BestBin}$ from $G_{BestBin}$;
    else
      $New := |BINS| + 1$;
      Create copy of G: G_{New};
      $BINS := BINS \cup \{G_{New}\}$;
      Find shortest path, P_j^{New}, for lightpath (s_j, d_j) in G_{New};
      $P_j = P_j^{New}$;
      Assign wavelength New to path P_j;
      Delete edges in P_j^{New} from G_{New}
    end if;
    $P = P \cup P_j$;
    $\tau = \tau \setminus (s_j, d_j)$;
  end for;
end while;
return P;
End

Figure 3.1: Pseudocodes of the FF_RWA, BF_RWA, FFD_RWA, and BFD_RWA algorithms.

wavelengths.

FFD_RWA

The First Fit Decreasing Routing and Wavelength Assignment algorithm sorts the lightpath requests in non-increasing order of the lengths of their shortest paths, SP_j, in G. Lightpaths with shortest paths of the same length are placed in random order. The algorithm then proceeds as FF_RWA.

The motivation for such an approach is as follows. If the connection request with the longest shortest path is considered first, it will be routed in 'empty' bin G_1, i.e. $G_1 = G$. This means the lightpath will not only successfully be routed in G_1, but will be routed on its overall shortest path. After deleting the corresponding edges from bin G_1, the remaining edges can be used to route 'shorter' lightpath requests which are easier to route on alternative routes that are satisfactory (i.e. shorter than H). In other words, the FFD_RWA algorithm first routes 'longer' lightpaths which are harder to route, and then fills up the remaining space

in each bin with 'shorter' lightpaths. This may lead to fewer wavelengths used.

BFD_RWA

The Best Fit Decreasing Routing and Wavelength Assignment algorithm sorts the lightpath requests in non-increasing order of the lengths of their shortest paths SP_j in G, and then proceeds as BF_RWA.

Pseudocodes of the FF_RWA, BF_RWA, FFD_RWA and BFD_RWA algorithms are shown in Fig. 3.1. Some 'first fit' and 'longest path first' approaches have been used by wavelength assignment algorithms [12], but to the best of our knowledge have not been used to solve the routing subproblem, or for simultaneous routing and wavelength assignment. All four algorithms efficiently solve the static RWA problem, while the FF_RWA and BF_RWA algorithms can also be used for dynamic RWA. This makes sense since they are analogous to the 'on-line' FF and BF bin packing algorithms. Namely, in the *dynamic* RWA problem, lightpath requests in τ are not known *a priori*, but arrive unexpectedly. This means that lightpaths in τ are established in a specific order, i.e. in the order in which they arrive. If such is the case, the FF_RWA and BF_RWA algorithms simply establish lightpaths in the specified order according to their corresponding 'first fit' or 'best fit' strategies.

3.4 Lower Bounds

Since the algorithms considered in this chapter are heuristics which obtain upper bounds on the minimal objective function values, it is useful to have good lower bounds in order to assess the quality of the sub-optimal solutions. We use a lower bound for the number of wavelengths required which is similar to a lower bound developed for the virtual topology design problem presented in [76]. Stronger lower bounds may be found using more sophisticated methods, such as that in [83], but we use the lower bound presented here for its simplicity. Namely, the algorithms were tested for fairly large test problems, so using complex algorithms for finding better lower bounds was not practical. Furthermore, computational results demonstrate the efficiency of the suggested lower bound, particularly in denser networks. This bound on the number of wavelengths needed to establish a given set τ of n lightpath requests in a network with $|V|$ nodes and $|E_p|$ edges is

$$LB_W = max\{ \max_{i \in V} \lceil \frac{\Delta_{l_{out}}(i)}{\Delta_p(i)} \rceil, \max_{i \in V} \lceil \frac{\Delta_{l_{in}}(i)}{\Delta_p(i)} \rceil, \lceil \frac{\sum_{j=1}^{n} l(SP_j)}{2 * |E_p|} \rceil \}. \quad (3.1)$$

$\Delta_{l_{out}}(i)$ represents the logical out-degree of node i, i.e. the number of lightpaths for which node i is the source node. $\Delta_{l_{in}}(i)$ represents the logical in-degree of node i, i.e. the number of lightpaths for which node i is the destination node. Since all edges in graph G are bidirectional, the physical in-degree is equal to the physical out-degree for each node i and is denoted as $\Delta_p(i)$. $l(SP_j)$ is the length of the shortest path in G of lightpath request (s_j, d_j). The first element in (5.1) represents the maximum ratio of logical to physical out-degree of any node in G, rounded up to the first higher integer. If some node i has $\Delta_p(i)$ adjacent physical links and is the source node for $\Delta_{l_{out}}(i)$ lightpaths, at least one physical link will have $\lceil \frac{\Delta_{l_{out}}(i)}{\Delta_p(i)} \rceil$ lightpaths routed over it. Since lightpaths routed on the same physical links cannot be assigned the same wavelength, at least $\lceil \frac{\Delta_{l_{out}}(i)}{\Delta_p(i)} \rceil$ wavelengths are needed to route the corresponding lightpaths. The highest such ratio among all the nodes in the network is a lower bound on the number of wavelengths needed to perform RWA for set τ. The second element in (5.1) is analogous to the first element, but represents the maximum ratio of logical to physical *in*-degree of any node in G, rounded up to the first higher integer. In some cases, the third element in (5.1) may give a better lower bound. This element represents the minimum total physical hop length of all the lightpaths divided by the total number of links in the network. The minimum total physical hop length of the established lightpaths is the sum of the lengths of the shortest paths (in terms of hops) of all the lightpath requests. Since each edge in E_p represents 2 links, one in each direction, the total physical hop length is divided by $2 * |E_p|$.

A simple lower bound on the average physical length of the established lightpaths is equal to the average length of the shortest paths in G of all the lightpath requests in τ. We consider the lengths of lightpaths in terms of hops and refer to this lower bound as the Physical Hops lower bound, LB_{PH}. The bound is as follows.

$$LB_{PH} = \frac{\sum_{j=1}^{n} l(SP_j)}{n}. \quad (3.2)$$

Table 3.1: 100-node test networks with an average degree of 3: Lower bound and the average (*lowest, highest*) number of wavelengths used in the solutions obtained by the *Greedy_EDP_RWA* algorithm (from [57]), and the BF_RWA, FFD_RWA and BFD_RWA algorithms proposed in this thesis.

Test Netw.	P_l	Lightpath requests	LB_W	*Greedy_EDP _RWA (FF_RWA)*	*BF_RWA*	*FFD_RWA*	*BFD_RWA*
1	0.2	2054	29	48.8 (*47,52*)	45.2 (*45,46*)	**45** (*45,45*)	**45** (*45,45*)
2		2034	27	49.6 (*49,51*)	49.1 (*49,50*)	**49** (*49,49*)	**49** (*49,49*)
3		2006	29	34.4 (*33,35*)	31.4 (*31,32*)	**29.9** (*29,30*)	30.5 (*30,31*)
4		2044	22	31.4 (*30,33*)	27.4 (*27,29*)	28 (*28,28*)	**26** (*25,27*)
5		2109	25	35.7 (*35,38*)	30.6 (*30,32*)	31.1 (*31,32*)	**29** (*28,30*)
1	0.4	4063	50	91.3 (*89,95*)	87.1 (*87,88*)	**87** (*87,87*)	**87** (*87,87*)
2		4038	47	96.9 (*96,98*)	96.1 (*96,97*)	**96** (*96,96*)	**96** (*96,96*)
3		3982	52	68.1 (*67,69*)	62.7 (*62,65*)	**60.6** (*60,61*)	61.2 (*61,62*)
4		4045	44	60 (*58,64*)	51 (*50,54*)	52.3 (*52,53*)	**48.7** (*48,49*)
5		4122	51	68.6 (*67,71*)	57.6 (*56,59*)	58.3 (*58,59*)	**56** (*54,58*)
1	0.6	6054	71	130 (*128,134*)	122.6 (*122,124*)	**122** (*122,122*)	**122** (*122,122*)
2		6020	66	127.9 (*127,129*)	**126** (*126,126*)	**126** (*126,126*)	**126** (*126,126*)
3		5999	66	100.3 (*98,104*)	92.5 (*92,95*)	**91** (*91,91*)	91.8 (*91,92*)
4		6048	62	86.1 (*85,88*)	73 (*72,74*)	77. 2 (*77,78*)	**71.4** (*71,72*)
5		6095	71	98 (*96,99*)	**82.3** (*79,86*)	84.4 (*84,86*)	85.7 (*83,88*)
1	0.8	8014	88	167.4 (*165,170*)	**161** (*161,161*)	**161** (*161,161*)	**161** (*161,161*)
2		8006	84	159.7 (*159,161*)	**158** (*158,158*)	**158** (*158,158*)	**158** (*158,158*)
3		7985	89	133.4 (*130,136*)	121.5 (*121,123*)	**120** (*120,120*)	**120** (*120,120*)
4		8008	86	115.5 (*114,119*)	97 (*95,99*)	100.8 (*100,101*)	**94.5** (*94,95*)
5		8046	88	127.2 (*123,131*)	**103.9** (*101,107*)	109 (*109,109*)	106.7 (*104,109*)
1	1.0	9900	99	207.5 (*205,210*)	**196** (*196, 196*)	**196** (*196,196*)	**196** (*196,196*)
2		9900	99	198.4 (*196,203*)	**196** (*196,196*)	**196** (*196,196*)	**196** (*196,196*)
3		9900	99	166.1 (*164,170*)	150.4 (*149,152*)	147.1 (*146,149*)	**146.1** (*146,147*)
4		9900	99	140.9 (*138,143*)	117.1 (*115,120*)	123.5 (*123,124*)	**114.6** (*114,115*)
5		9900	99	154.5 (*151,158*)	128.2 (*125,131*)	132.7 (*132,133*)	**129.6** (*127,132*)

3.5 Numerical Results

In order to determine the performance measures of the proposed algorithms, the *Greedy_EDP_RWA* [57] and the BF_RWA, FFD_RWA and BFD_RWA algorithms were implemented in C++ and run on a PC powered by a P4 2.8GHz processor. The FF_RWA algorithm was not implemented since it yields solutions equivalent to those obtained by the *Greedy_EDP_RWA* algorithm. A series of random 100-node networks with average degrees of 3, 4, and 5 were created (5 networks per average degree). Random sets of lightpath requests were created for each test network with the probability P_l of there being a lightpath request between two nodes. The value of P_l ranged from 0.2 to 1.0, in 0.2 increments, for up to 9900 lightpath requests. The upper bound on the physical hop length of the established lightpaths, H, is set here to $\max(diam(G), \sqrt{|E_p|})$ as suggested in [57].

Table 3.2: 100-node test networks with an average degree of 3: Lower bound and the average lightpath length in the solutions obtained by the *Greedy_EDP_RWA* algorithm (from [57]), and the BF_RWA, FFD_RWA and BFD_RWA algorithms proposed in this thesis.

Test Network	P_l	Lightpath requests	LB_{PH}	*Greedy_EDP _RWA (FF_RWA)*	*BF_RWA*	*FFD_RWA*	*BFD_RWA*
1	0.2	2054	3.48	4.55	3.51	4.58	**3.48***
2		2034	3.40	4.44	3.43	4.46	**3.40***
3		2006	3.43	4.48	3.49	4.50	**3.47**
4		2044	3.39	4.50	3.65	4.52	**3.58**
5		2109	3.48	4.60	3.67	4.64	**3.65**
1	0.4	4063	3.48	4.54	3.49	4.56	**3.48***
2		4038	3.38	4.39	3.39	4.39	**3.38***
3		3982	3.44	4.46	3.48	4.49	**3.47**
4		4045	3.40	4.51	3.58	4.51	**3.53**
5		4122	3.51	4.62	3.63	4.64	**3.62**
1	0.6	6054	3.48	4.51	3.49	4.55	**3.48***
2		6020	3.38	4.37	3.39	4.40	**3.38***
3		5999	3.43	4.45	3.47	4.48	**3.46**
4		6048	3.42	4.50	3.57	4.52	**3.54**
5		6095	3.51	4.58	**3.59**	4.62	**3.59**
1	0.8	8014	3.48	4.53	3.49	4.54	**3.48***
2		8006	3.38	4.37	3.39	4.40	**3.38***
3		7985	3.45	4.45	**3.48**	4.49	**3.48**
4		8008	3.43	4.51	3.57	4.53	**3.54**
5		8046	3.51	4.59	**3.60**	4.61	**3.60**
1	1	9900	3.48	4.52	**3.49**	4.55	**3.49**
2		9900	3.39	4.37	3.40	4.41	**3.39***
3		9900	3.45	4.44	**3.48**	4.48	**3.48**
4		9900	3.43	4.49	3.56	4.53	**3.55**
5		9900	3.51	4.59	**3.61**	4.60	**3.61**

All four algorithms were run with 10 different seeds (i.e. 10 different permutations of τ) for each test case. The average, lowest and highest number of wavelengths of the solutions obtained by each algorithm were recorded. The average physical hop lengths of the established lightpaths were also found. The average number of wavelengths needed to successfully perform Routing and Wavelength Assignment by each of the algorithms for the test networks with an average degree of 3 are shown in Table 3.1. The lowest and highest solution values found are shown in parenthesis. The lower bound, LB_W, is also shown. The corresponding average lightpath lengths and the lower bound PH_{LB} are shown in Table 3.2. For test networks with an average degree of 4, the wavelengths and average lightpath lengths of the obtained solutions are shown in Tables 3.3 and 3.4, respectively. Tables 3.5 and 3.6 show the results obtained for test networks with an average degree of 5. The best obtained

Table 3.3: 100-node test networks with an average degree of 4: Lower bound and the average (*lowest, highest*) number of wavelengths used in the solutions obtained by the *Greedy_EDP_RWA* algorithm (from [57]), and the BF_RWA, FFD_RWA and BFD_RWA algorithms proposed in this thesis.

Test Netw.	P_l	Lightpath requests	LB_W	*Greedy_EDP_RWA (FF_RWA)*	*BF_RWA*	*FFD_RWA*	*BFD_RWA*
1		2116	21	24.3 (*23,25*)	21.7 (*21,24*)	**21*** (*21,21*)	**21*** (*21,21*)
2		2081	25	28.1 (*26,30*)	25.4 (*25,27*)	**25*** (*25,25*)	**25*** (*25,25*)
3	0.2	2067	24	25.8 (*24,27*)	24.4 (*24,26*)	**24*** (*24,24*)	**24*** (*24,24*)
4		2054	29	29.9 (*29,31*)	29.4 (*29,31*)	**29*** (*29,29*)	**29*** (*29,29*)
5		2125	32	33.8 (*32,36*)	**32*** (*32,32*)	**32*** (*32,32*)	**32*** (*32,32*)
1		4063	39	44.3 (*43,46*)	39.9 (*39,43*)	**39*** (*39,39*)	**39*** (*39,39*)
2		4047	46	50.9 (*49,53*)	47.2 (*47,48*)	**47** (*47,47*)	**47** (*47,47*)
3	0.4	4064	50	53.4 (*52,55*)	50.1 (*50,51*)	**50*** (*50,50*)	**50*** (*50,50*)
4		4063	47	51.4 (*50,52*)	48.3 (*47,50*)	**47*** (*47,47*)	**47*** (*47,47*)
5		4099	50	55.8 (*53,59*)	50.3 (*50,51*)	**50*** (*50,50*)	**50*** (*50,50*)
1		6017	61	66.9 (*63,69*)	61.1 (*61,62*)	**61*** (*61,61*)	**61*** (*61,61*)
2		5995	69	74.6 (*72,78*)	69.1 (*69,70*)	**69*** (*69,69*)	**69*** (*69,69*)
3	0.6	6054	67	75.4 (*73,78*)	67.8 (*67,71*)	**67*** (*67,67*)	**67*** (*67,67*)
4		6054	71	77.5 (*75,81*)	71.2 (*71,72*)	**71*** (*71,71*)	**71*** (*71,71*)
5		6113	66	77.9 (*76,83*)	67.6 (*66,70*)	**66*** (*66,66*)	**66*** (*66,66*)
1		7960	80	86.7 (*84,89*)	80.1 (*80,81*)	**80*** (*80,80*)	**80*** (*80,80*)
2		7988	81	89.8 (*88,93*)	81.8 (*81,83*)	**81*** (*81,81*)	**81*** (*81,81*)
3	0.8	8052	88	99.4 (*96,103*)	89.8 (*88,93*)	**88*** (*88,88*)	**88*** (*88,88*)
4		8014	86	94.4 (*91,99*)	86.6 (*86,89*)	**86*** (*86,86*)	**86*** (*86,86*)
5		8017	88	101.4 (*97,106*)	88.9 (*88,90*)	**88*** (*88,88*)	**88*** (*88,88*)
1		9900	99	108.3 (*106,110*)	99.9 (*99,102*)	**99*** (*99,99*)	**99*** (*99,99*)
2		9900	99	110.7 (*108,113*)	100.4 (*99,102*)	**99*** (*99,99*)	**99*** (*99,99*)
3	1.0	9900	99	120 (*118,123*)	105 (*103,109*)	**99*** (*99,99*)	**99*** (*99,99*)
4		9900	99	110.8 (*108,112*)	99.1 (*99,100*)	**99*** (*99,99*)	**99*** (*99,99*)
5		9900	99	122.7 (*119,125*)	106.7 (*105,109*)	**103.6** (*103,104*)	104.5 (*104,105*)

solution for each test case is marked in bold. If the obtained solution is equal to the lower bound, i.e. the obtained solution is surely optimal, it is marked as '*'.

The BF_RWA, FFD_RWA and BFD_RWA algorithms all perform significantly better than the *Greedy_EDP_RWA* algorithm for all cases with respect to the number of wavelengths used. The FFD_RWA and BFD_RWA algorithms perform best for this optimization criterion. In fact, the worst solution obtained by the FFD_RWA and BFD_RWA algorithms is better or equal to the best solution obtained by the *Greedy_EDP_RWA* algorithm in all cases. The worst solution obtained by the BF_RWA algorithm is better or equal to the best solution obtained by the *Greedy_EDP_RWA* algorithm in all but 2 cases for networks with an average degree of 3, 4 cases for networks with an average degree of 4, and 7 cases for networks with an average degree of 5.

Since only those lightpaths whose shortest paths are equal in length are sorted randomly,

Table 3.4: 100-node test networks with an average degree of 4: Lower bound and the average lightpath length in the solutions obtained by the *Greedy_EDP_RWA* algorithm (from [57]), and the BF_RWA, FFD_RWA and BFD_RWA algorithms proposed in this thesis.

Test Network	P_l	Lightpath requests	LB_{PH}	*Greedy_EDP_RWA (FF_RWA)*	*BF_RWA*	*FFD_RWA*	*BFD_RWA*
1		2116	2.92	3.86	2.97	3.87	**2.93**
2		2081	2.95	3.88	2.98	3.87	**2.96**
3	0.2	2067	2.96	3.89	2.99	3.90	**2.96***
4		2054	3.05	4.03	3.11	4.03	**3.05***
5		2125	3.23	4.25	3.28	4.26	**3.24**
1		4063	2.91	3.84	2.93	3.85	**2.91***
2		4047	2.97	3.87	2.98	3.88	**2.97***
3	0.4	4064	2.97	3.87	2.98	3.88	**2.97***
4		4063	3.05	4.02	3.10	4.00	**3.05***
5		4099	3.24	4.23	3.29	4.25	**3.26**
1		6017	2.92	3.82	2.93	3.83	**2.92***
2		5995	2.96	3.85	2.97	3.87	**2.96***
3	0.6	6054	2.98	3.87	**2.98***	3.88	**2.98***
4		6054	3.05	4.00	3.07	4.00	**3.05***
5		6113	3.24	4.23	3.29	4.23	**3.27**
1		7960	2.93	3.83	2.94	3.84	**2.93***
2		7988	2.97	3.84	2.98	3.85	**2.97***
3	0.8	8052	2.98	3.86	2.99	3.87	**2.98***
4		8014	3.05	4.00	3.08	4.00	**3.05***
5		8017	3.24	4.22	3.27	4.23	**3.26**
1		9900	2.94	3.83	**2.94***	3.83	**2.94***
2		9900	2.97	3.85	**2.98**	3.85	**2.98**
3	1	9900	2.98	3.86	2.99	3.86	**2.98***
4		9900	3.05	4.00	3.07	4.00	**3.05***
5		9900	3.24	4.20	3.28	4.22	**3.27**

the FFD_RWA and BFD_RWA algorithms usually perform the same for various permutations of τ. As a result, the worst solutions obtained by the FFD_RWA and BFD_RWA algorithms were in most cases their best solutions. These solutions were also better or equal to those obtained by the *Greedy_EDP_RWA* and BF_RWA algorithms. This seems to indicate that the FFD_RWA and BFD_RWA algorithms could be run only once and still obtain high quality solutions. The *Greedy_EDP_RWA* and BF_RWA algorithms, on the other hand, need to be run as multistart algorithms and even then they obtain inferior solutions.

As can be seen in Table 3.3, for networks with average degree 4, the *average* solutions obtained by the FFD_RWA and BFD_RWA algorithms were optimal in at least 23 out of the 25 test cases, in one case for the BF_RWA algorithm, and in zero cases for the *Greedy_EDP_RWA* algorithm. To obtain better results with the *Greedy_EDP_RWA* and

Table 3.5: 100-node test networks with an average degree of 5: Lower bound and the average (*lowest, highest*) number of wavelengths used in the solutions obtained by the *Greedy_EDP_RWA* algorithm (from [57]), and the BF_RWA, FFD_RWA and BFD_RWA algorithms proposed in this thesis.

Test Netw.	P_l	Lightpath requests	LB_W	*Greedy_EDP _RWA (FF_RWA)*	*BF_RWA*	*FFD_RWA*	*BFD_RWA*
1		2116	20	21.7 (*21,23*)	20.3 (*20,22*)	**20*** (*20,20*)	**20*** (*20,20*)
2		2029	27	27.3 (*27,28*)	27.1 (*27,28*)	**27*** (*27,27*)	**27*** (*27,27*)
3	0.2	2081	24	24.8 (*24,27*)	**24*** (*24,24*)	**24*** (*24,24*)	**24*** (*24,24*)
4		2067	19	20.1 (*19,22*)	19.1 (*19,20*)	**19*** (*19,19*)	**19*** (*19,19*)
5		2098	23	24.6 (*23,25*)	23.1 (*23,24*)	**23*** (*23,23*)	**23*** (*23,23*)
1		4063	37	39.4 (*38,41*)	37.2 (*37,38*)	**37*** (*37,37*)	**37*** (*37,37*)
2		3988	46	48.8 (*48,50*)	46.1 (*46,47*)	**46*** (*46,46*)	**46*** (*46,46*)
3	0.4	4047	46	47.2 (*46,48*)	46.1 (*46,47*)	**46*** (*46,46*)	**46*** (*46,46*)
4		4064	43	45.6 (*44,47*)	43.6 (*43,44*)	**43*** (*43,43*)	**43** *(*43,43*)
5		4100	47	47.7 (*47,49*)	**47*** (*47,47*)	**47*** (*47,47*)	**47*** (*47,47*)
1		6017	60	61.8 (*61,64*)	**60*** (*60,60*)	**60*** (*60,60*)	**60*** (*60,60*)
2		5963	68	70.3 (*69,72*)	68.1 (*68,69*)	**68*** (*68,68*)	**68*** (*68,68*)
3	0.6	5995	69	71.1 (*70,73*)	69 (*69,69*)	**69*** (*69,69*)	**69*** (*69,69*)
4		6054	63	65.7 (*63,69*)	63.1 (*63,64*)	**63*** (*63,63*)	**63*** (*63,63*)
5		6088	62	64.1 (*62,66*)	62.2 (*62,63*)	**62*** (*62,62*)	**62*** (*62,62*)
1		7960	79	82.2 (*81,85*)	**79*** (*79,79*)	**79*** (*79,79*)	**79*** (*79,79*)
2		7984	88	91.3 (*89,94*)	88.1 (*88,89*)	**88*** (*88,88*)	**88*** (*88,88*)
3	0.8	7988	80	83.5 (*81,85*)	80.1 (*80,81*)	**80*** (*80,80*)	**80*** (*80,80*)
4		8052	86	89.6 (*87,94*)	86.1 (*86,87*)	**86*** (*86,86*)	**86*** (*86,86*)
5		7994	81	83.7 (*82,87*)	81.1 (*81,82*)	**81*** (*81,81*)	**81*** (*81,81*)
1		9900	99	102.4 (*100,104*)	**99*** (*99,99*)	**99*** (*99,99*)	**99*** (*99,99*)
2		9900	99	109.9 (*107,113*)	100.1 (*99,102*)	**99*** (*99,99*)	**99*** (*99,99*)
3	1.0	9900	99	103.5 (*101,106*)	99.1 (*99,100*)	**99*** (*99,99*)	**99*** (*99,99*)
4		9900	99	105.2 (*103,107*)	99.1 (*99,100*)	**99*** (*99,99*)	**99*** (*99,99*)
5		9900	99	104.4 (*102,108*)	99.1 (*99,100*)	**99*** (*99,99*)	**99*** (*99,99*)

BF_RWA algorithms, they could be run as multistart algorithms and then select the best found solution. The *best* solution obtained by the BF_RWA algorithm was optimal in 22 cases while the *best* solution obtained by the *Greedy_EDP_RWA* was optimal in only 3 cases. Table 3.5 indicates that for test networks with average degree 5, the *average* solutions obtained by the FFD_RWA and BFD_RWA algorithms were optimal in all of the 25 test cases, while the BF_RWA and *Greedy_EDP_RWA* algorithms obtained optimal average solutions in five and zero cases, respectively. For these networks, the *best* solution obtained by the BF_RWA algorithm was optimal in all cases while the *best* solution obtained by the *Greedy_EDP_RWA* was optimal in only 8 cases. It is evident that sorting lightpath requests in non-increasing order of their shortest paths helps obtain solutions using fewer wavelengths than establishing lightpaths at random.

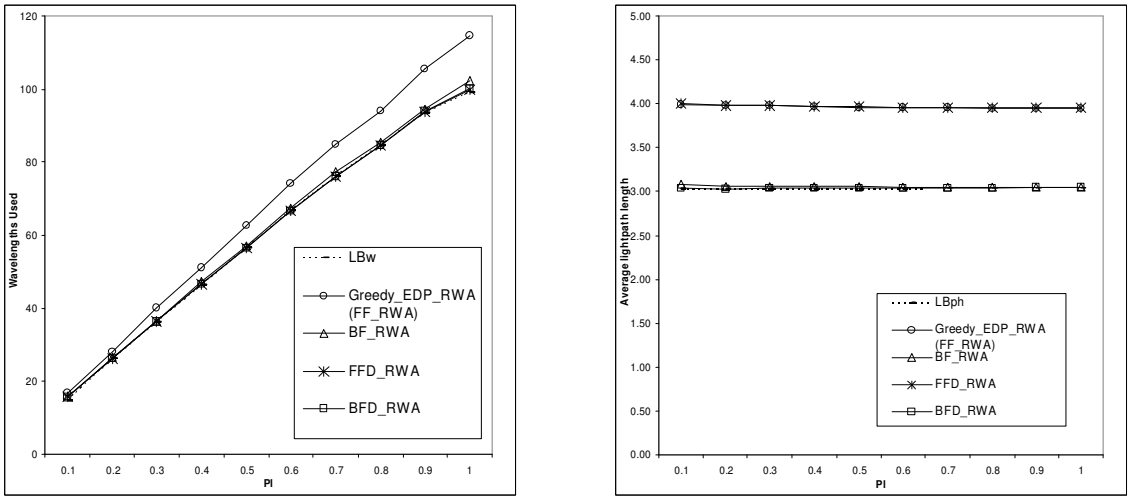
The average length of the established lightpaths are compared in Tables 3.2, 3.4 and 3.6. Both the BF_RWA and BFD_RWA algorithms perform significantly better than the FF_RWA

Table 3.6: 100-node test networks with an average degree of 5: Lower bound and the average lightpath length in the solutions obtained by the *Greedy_EDP_RWA* algorithm (from [57]), and the BF_RWA, FFD_RWA and BFD_RWA algorithms proposed in this thesis.

Test Network	P_l	Lightpath requests	LB_{PH}	*Greedy_EDP_RWA (FF_RWA)*	*BF_RWA*	*FFD_RWA*	*BFD_RWA*
1	0.2	2116	2.71	3.58	2.75	3.58	**2.71***
2		2029	2.83	3.67	2.84	3.68	**2.83***
3		2081	2.70	3.55	2.72	3.54	**2.70***
4		2067	2.67	3.50	2.71	3.51	**2.67***
5		2098	2.77	3.62	2.79	3.62	**2.77***
1	0.4	4063	2.70	3.55	2.73	3.55	**2.70***
2		3988	2.86	3.69	**2.86***	3.71	**2.86***
3		4047	2.71	3.55	2.73	3.54	**2.71***
4		4064	2.67	3.48	2.68	3.47	**2.67***
5		4100	2.74	3.58	2.75	3.58	**2.74***
1	0.6	6017	2.71	3.55	2.72	3.54	**2.71***
2		5963	2.84	3.67	2.85	3.69	**2.84***
3		5995	2.70	3.52	2.71	3.52	**2.70***
4		6054	2.68	3.48	2.70	3.48	**2.68***
5		6088	2.74	3.56	2.75	3.56	**2.74***
1	0.8	7960	2.71	3.55	2.73	3.54	**2.71***
2		7984	2.84	3.67	2.85	3.69	**2.84***
3		7988	2.71	3.53	2.72	3.52	**2.71***
4		8052	2.68	3.48	2.69	3.47	**2.68***
5		7994	2.74	3.56	**2.74***	3.55	**2.74***
1	1	9900	2.72	3.55	2.73	3.54	**2.72***
2		9900	2.83	3.65	**2.83***	3.67	**2.83***
3		9900	2.71	3.52	2.72	3.52	**2.71***
4		9900	2.69	3.47	2.70	3.47	**2.69***
5		9900	2.74	3.55	**2.74***	3.54	**2.74***

and FFD_RWA algorithms, although the BFD_RWA algorithm performs best in all cases. In fact, the BFD_RWA algorithm obtains the optimal solution in at least 9, 17 and 25 cases for networks with average degrees 3, 4 and 5, respectively. Routing the lightpaths according to the 'best fit' strategy evidently leads to shorter lightpaths than using the 'first fit' strategy. For easier visualization of the obtained results, the average wavelengths and lightpath lengths of the solutions found for the test networks with an average degree of 4 are shown graphically in Fig. 3.2. Here the values for P_l ranged from 0.1 to 1.0 in 0.1 increments.

Furthermore, the algorithms were tested on a reference European core network topology shown in Fig. 3.3 which was designed as part of the COST Action 266 project [37]. P_l from 0.1 to 1.0 in 0.1 increments. The results are shown in Fig. 3.4. The lower bound LB_W on the required number of wavelengths is not efficient for this network topology, so we assess the



(a) (b)

Figure 3.2: 100-node test networks with an average degree of 4: Comparison of the (a) average number of wavelengths used and the (b) average lightpath length in the solutions obtained by the *Greedy_EDP_RWA* algorithm (from [57]), and the BF_RWA, FFD_RWA and BFD_RWA algorithms proposed in this thesis.

quality of the algorithms by comparing them to each other. Since the network is small and not many alternative paths are available, the algorithms performed fairly similar with respect to the number for wavelengths used (Fig. 3.4.(a)). However, the BF_RWA, FFD_RWA and BFD_RWA algorithms performed the same or better than the *Greedy_EDP_RWA* algorithm in all cases. Fig. 3.4.(b) indicates that the 'best fit' algorithms again perform significantly better than the 'first fit' algorithms with respect to the average hop length. The BFD_RWA algorithm established the shortest lightpaths in all cases.

All four algorithms are very fast and highly tractable. The FFD_RWA and BFD_RWA algorithms are slower than the *Greedy_EDP_RWA*(FF_RWA) and BF_RWA algorithms by the time it takes to sort the lightpath demands. On the other hand, these algorithms are more robust and often give their best solutions in every run. As a result, these algorithms only need to be run once. The best and worst solution values obtained by the *Greedy_EDP_RWA* algorithm, on the other hand, vary significantly so this algorithm needs to be run as a multistart algorithm in order to obtain good results. This, of course, leads to much larger execution times. It should also be noted that the 'best fit' algorithms are somewhat slower with respect to the 'first fit' algorithms since they search among all the existing bins to find the 'best fit', while the first fit algorithms establishes the first found satisfactory route. When run on

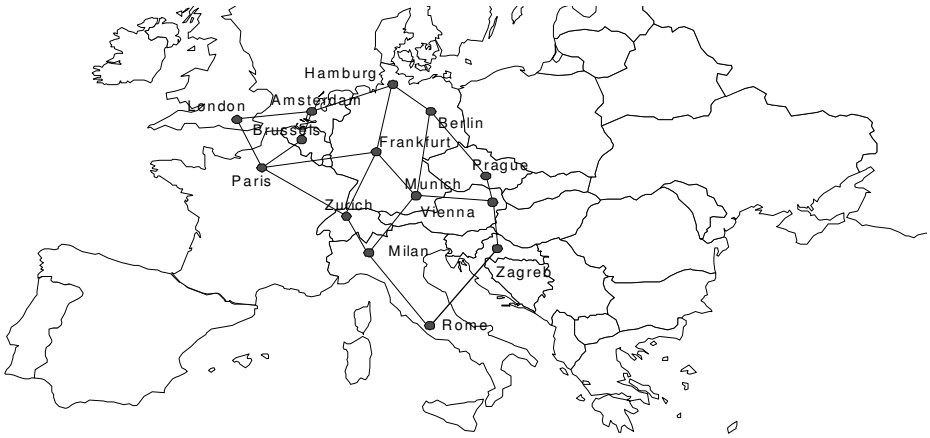
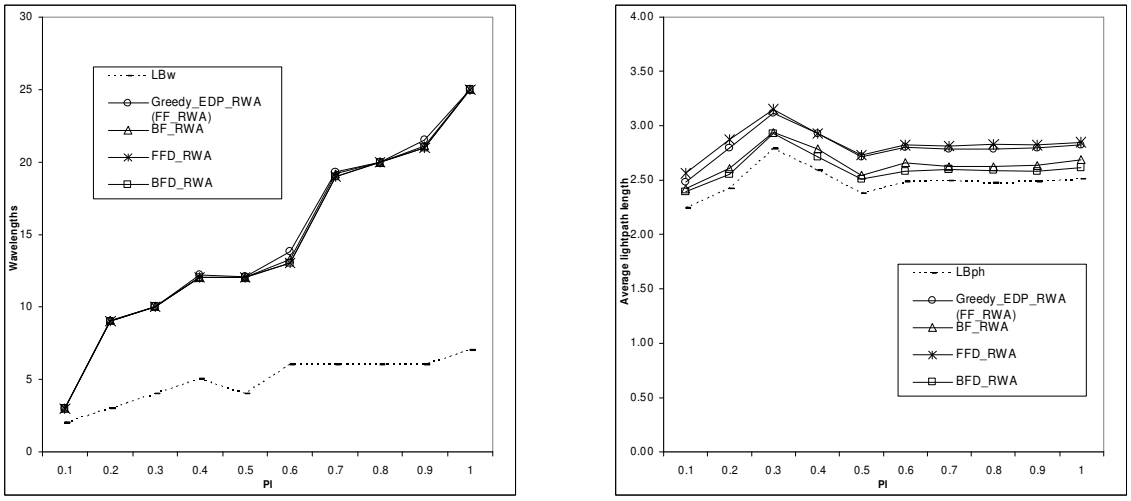


Figure 3.3: The hypothetical European core network [37].

a PC powered by a P4 2.8GHz processor, the maximum execution time for the FFD_RWA and BFD_RWA algorithms for the 100 node networks with 9900 lightpath requests was less than 8 minutes. The maximum execution time for the FF_RWA and BF_RWA algorithms was under 6 minutes. For the European core network, all algorithms performed under half a second.

The following conclusions can be drawn from the obtained results. Sorting lightpaths in non-increasing order of their shortest paths helps to obtain solutions using significantly fewer wavelengths. We can see from Tables 3.1, 3.3 and 3.5 that the advantage of sorting lightpaths becomes increasingly evident as the number of lightpath requests increases ($P_l \nearrow$). These are the cases where RWA is more challenging since we wish to establish a larger number of lightpaths. Routing lightpaths according to the 'best fit' strategy helps to consistently reduce lightpath hop length. The BFD_RWA algorithm, which both sorts lightpaths and uses the 'best fit' strategy, clearly performs best for all test cases.

Furthermore, recall that the *Greedy_EDP_RWA* and BF_RWA algorithms can be run for the dynamic Routing and Wavelength Assignment problem. For this problem, the mentioned algorithms are not run in multistart mode, but run once for each permutation of τ. As a result, we compare the *average* solution values obtained for the various permutations of τ. The BF_RWA algorithm performed significantly and consistently better than the *Greedy_EDP_RWA* algorithm with respect to both wavelengths and lightpath lengths. Using less wavelengths leaves more room for future lightpath requests. This decreases the chances that a lightpath request will be blocked due to the lack of available resources, which is a common objective criterion used to solve the dynamic RWA problem.



(a) (b)

Figure 3.4: The hypothetical European core network [37]: Comparison of the (a) average number of wavelengths used and the (b) average lightpath length in the solutions obtained by the *Greedy_EDP_RWA* algorithm (from [57]), and the BF_RWA, FFD_RWA and BFD_RWA algorithms proposed in this thesis.

3.6 Summary and Future Work

Successful solvability of the static Routing and Wavelength Assignment (RWA) problem is mandatory for making efficient use of resources in wavelength routed optical networks. In this chapter, the bin packing problem is applied to optical networks to help develop highly efficient heuristic algorithms for the RWA problem. Suggested are methods of sorting and routing lightpaths which not only reduce the required number of wavelengths, but reduce the average physical length of established lightpaths as well. Numerical results indicate that the proposed methods obtain optimal or near optimal solutions in many cases, and significantly outperform an efficient existing algorithm from [57] for the same problem. Furthermore, the heuristics are robust and highly tractable and can thus be used to solve large problem instances in reasonable time. Further avenues of research will include developing similar algorithms for routing and wavelength assignment in networks with full or limited wavelength conversion. Networks equipped with a limited number of transceivers and/or a limited number of wavelengths will also be considered.

Chapter 4

Scheduled Routing and Wavelength Assignment

In this chapter we consider the Routing and Wavelength Assignment of Scheduled Lightpath Demands (RWA SLD) in networks with no wavelength converters. We assume no limit on the number of transmitters and receivers at each node and the number of available wavelengths on each link. The objective is to minimize the number of wavelengths needed to successfully establish a desired set of scheduled lightpath demands. Considering scheduled lightpath demands seems relevant due to the periodic nature of traffic [52]. We know that traffic between some nodes (e.g. office headquarters) is heavier during office hours than in the middle of the night, and *vice versa* for the other nodes (e.g. networked data bases). We could utilize this information by setting up multiple lightpaths between nodes at times when their traffic is heavy and tearing down some or all of these lightpaths at times when their traffic is low. By tearing down lightpaths between nodes at times of low traffic, their resources are freed and can thus be used to establish alternative connections. If we have lightpath demands which do not overlap in time and if we take this information into consideration when performing routing and wavelength assignment, we can route both demands on the same path using the same wavelength without them clashing. This can significantly reduce the amount of network resources required to successfully route a set of lightpath demands.

Since the Routing and Wavelength Assignment of Scheduled Lightpath Demands is solved *a priori* using given scheduling information, the lightpaths can be set up and torn down quickly at the specified times. This is an advantage over the routing and wavelength assignment of dynamic lightpath demands where RWA is performed dynamically as lightpath requests arrive, resulting in longer set-up delays. The RWA SLD problem has not been studied as widely as the static and dynamic cases. A branch and bound algorithm along with

a tabu search heuristic algorithm are given in [52]. In this chapter, we suggest a faster and more efficient tabu search algorithm for the RWA SLD problem along with two very fast and simple greedy algorithms based on edge and time disjoint paths. Furthermore, we derive an efficient lower bound on the number of wavelengths needed for the RWA SLD problem.

The rest of this chapter is organized as follows. In Section 4.1, we define the RWA SLD problem. Related work is briefly described in Section 4.2. In Section 4.3 we suggest a tabu search algorithm for the RWA problem, followed by two simple yet very efficient greedy algorithms in Section 4.4. In 4.5 we discuss lower bounds. Numerical results and a chapter summary are given in Sections 4.6 and 4.7, respectively.

4.1 Problem Definition

The physical optical network is modelled as a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. Edges are assumed to be bidirectional (each representing a pair of optical fibers, i.e. one fiber per direction) and have assigned weights representing their length or cost. Given is a set of scheduled lightpath demands $\tau = \{SLD_1, \ldots, SLD_M\}$. Each scheduled lightpath demand SLD_i, where $i = 1, \ldots, M$, is represented by a tuple $(s_i, d_i, n_i, \alpha_i, \omega_i)$ as suggested in [52]. Here $s_i, d_i \in V$, are the source and destination nodes of SLD_i, n_i is the the number of requested lightpaths between these nodes, and α_i and ω_i are the set-up and tear-down times respectively. The Routing and Wavelength Assignment problem consists of finding a set of paths $P = \{P(SLD_1), \ldots, P(SLD_M)\}$ in G, each corresponding to one scheduled lightpath demand, and assigning a set of wavelengths to each of these paths. As in [52], we assume that all the lightpaths of a particular SLD must be routed on the same path and must therefore be assigned different wavelengths. We will refer to this as the *group lightpath constraint*. As a result, each path $P(SLD_i)$, where $i = 1, \ldots, M$, must be assigned a set of n_i wavelengths, one for each individual lightpath of SLD_i. The set of wavelengths assigned to paths $P(SLD_i)$ and $P(SLD_j)$, where $i \neq j$ and $i, j = 1, \ldots, M$, must be disjoint if these paths share a common edge *and* if SLD_i and SLD_j overlap in time. The objective is to minimize the number of wavelengths assigned and required to successfully route and assign wavelengths to all the scheduled lightpath demands in τ.

4.2 Related Work

In [52], the authors solve the Routing and Wavelength Assignment problem of Scheduled Lightpath Demands by decoupling it into two separate subproblems: routing and wavelength assignment. They suggest a branch and bound algorithm for the routing subproblem which provides optimal solutions but has an exponential complexity. To solve the routing subproblem for larger problems, the authors propose a tabu search algorithm which obtains suboptimal solutions. Two different optimization criteria are considered giving rise to two versions of the tabu search algorithm: TS_{ch} and TS_{cg}. The TS_{ch} algorithm minimizes the number of WDM channels[1] which is particularly important in opaque WDM networks and will not be discussed here. The TS_{cg} algorithm minimizes congestion, i.e. the number of lightpaths on the most heavily loaded link. This optimization criterion is important in networks with a limited number of wavelengths since congestion is essentially a lower bound on the number of wavelengths required. Wavelength assignment is performed subsequently using a greedy graph coloring algorithm (referred to as GGC) suggested in [43]. The objective of the GGC algorithm is to minimize the number of wavelengths used. The quality of the solutions for the RWA SLD problem obtained by TS_{cg}/GGC are measured in terms of the number of wavelengths needed. A complete description is provided in [52].

Fault tolerant routing and wavelength assignment of scheduled lightpath demands was studied in [51] and [82]. In [51], the authors formulate the problem of fault tolerant RWA SLD with the objective being to minimize the number of WDM channels. They propose a Simulated Annealing algorithm using channel reuse and back-up multiplexing. In [82], fault tolerant RWA SLD under single component failure is considered. The authors develop ILP formulations for the problem with dedicated and shared protection. Two objectives are considered: minimizing the capacity needed to guarantee protection for all connection requests and maximizing the number of requests accepted subject to a limited capacity.

[1] WDM channels refer to the use of a particular wavelength on a directed physical link and are specified by the wavelength used and the head and tail nodes of the directed link.

4.3 An Alternative Tabu Search Algorithm for the Routing Subproblem

4.3.1 Tabu Search

Tabu search is an iterative meta-heuristic which guides simpler heuristics in such a way that they explore various areas of the solution space and prevents them from remaining in local optima. In every iterative step of the tabu search method, we begin with some current solution and explore its neighboring solutions. Neighboring solutions with respect to the current one are all those obtained by applying some elementary transformation to the current solution. The best neighboring solution according to some evaluation function is selected as the new current solution in the next iteration. After executing a desired number of iterations, the best found solution overall, called the incumbent solution, is deemed the final result.

To prevent the search technique from getting stuck in a local optimum or cycling between already seen solutions, a memory structure called a tabu list is introduced. The tabu list 'memorizes' a certain number of previously visited solutions which are then forbidden for as long as they remain in the list. The tabu list is updated circularly after every iteration by adding the current solution to the list and removing the oldest element if the list is full. The length of the tabu list can vary depending on different problems and is often determined experimentally. The key to developing a good tabu search algorithm is to define a good initial solution, neighborhood structure and evaluation function. Sometimes the neighborhood of a solution can be very large so various neighborhood reduction techniques are applied. As a result, only a subset of the neighboring solutions are evaluated. A detailed description of the tabu search method can be found in [29].

4.3.2 The Proposed Tabu Search Algorithm: TS_{cn}

The tabu search algorithm TS_{cg} for the routing subproblem suggested by the authors of [52], although faster than their branch and bound algorithm, still has a fairly long execution time if run for a large enough neighborhood size and the number of iterations needed to obtain solutions of good quality. This is due to their randomized neighborhood search technique. Namely, the TS_{cg} algorithm begins by computing the K-shortest paths between the source and destination nodes of each of the M SLDs in τ. Potential routing solutions are represented by a vector of M integers (initially all set to 1) ranging from 1 to K, each representing the path used by a particular SLD. Neighboring solutions are all those in which one and only one SLD is routed on a different route. A neighborhood defined as such can be very large so

a desired number of SLDs are chosen at random, randomly rerouted and evaluated according to their corresponding congestion. To obtain good solutions, a large number of iterations and a fairly large number of neighbors need to be evaluated.

Since our final objective for the RWA SLD problem is to minimize the number of wavelengths used, we will compare our results with that of algorithm TS_{cg}/GGC. The evaluation function used by TS_{cg} (i.e. minimization of congestion) does not necessarily lower the number of wavelengths needed even though this is the final goal of the TS_{cg}/GGC algorithm. Recall that by minimizing congestion, which is the number of lightpaths on the most loaded link, we are essentially trying to minimize the lower bound on the number of wavelengths needed. Minimizing the *lower* bound does not necessarily mean that the number of wavelengths needed will be lower. On the other hand, if we tried to minimize the *upper* bound on the number of wavelengths needed, this guarantees that there exists a wavelength assignment with at most the upper bound number of wavelengths.

The tabu search algorithm for the routing subproblem suggested in this thesis attempts to improve the drawbacks mentioned above. A new evaluation function is suggested along with a directed neighborhood search technique which drastically reduces the neighborhood size. As a result, this algorithm, in combination with the same graph coloring algorithm (GGC) used in [52] for wavelength assignment, performs faster and obtains solutions of better or equal quality than those obtained by TS_{cg}/GGC. We will refer to the proposed algorithm as TS_{cn} where cn stands for *chromatic number*. The relevance of this name will be described below.

Preliminaries

Recall that the objective of the wavelength assignment algorithm as well as our final objective for the RWA SLD problem is to minimize the total number of wavelengths used. Since we perform wavelength assignment using the GGC algorithm *after* solving the routing subproblem, it seems that having a routing algorithm aware of the objective and behavior of the wavelength assignment algorithm could help to obtain better solutions for the RWA SLD problem. In other words, the optimization criteria of the routing algorithm should be such that it gives routing schemes on which wavelength assignment can be performed using a smaller number of wavelengths. To formulate such an optimization criteria we must first analyze the behavior of the wavelength assignment algorithm which is here essentially a graph coloring algorithm.

Namely, the problem of wavelength assignment can be reduced to the *graph coloring problem* which consists of assigning colors to the nodes of a graph such that no two neigh-

boring nodes are assigned the same color. The objective is to minimize the total number of colors used. This classical graph theory problem has been proven to be NP-complete so several heuristic algorithms have been developed [41]. One such algorithm is the GGC algorithm proposed in [43]. The routing solution obtained by solving the routing subproblem is used as input for the GGC algorithm in the following manner. A conflict graph corresponding to the routing solution is created where each established lightpath is represented by one node in the conflict graph[2] and there is an edge between two nodes if their respective paths share a common physical link in G *and* overlap in time. This means that the lightpaths corresponding to neighboring nodes in the conflict graph cannot be assigned the same wavelength. The graph coloring algorithm GGC is executed on this conflict graph where each color represents a different wavelength.

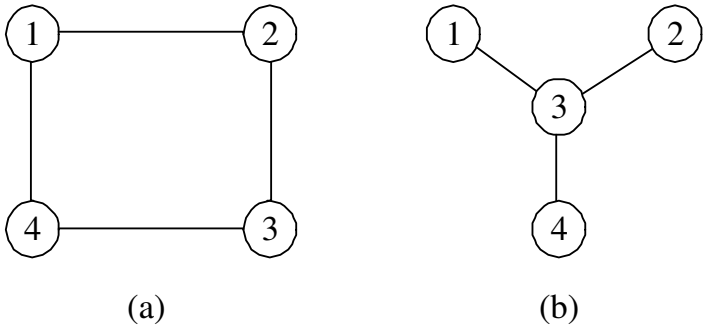


Figure 4.1: Two simple 4 node networks.

The minimum number of colors needed to color a graph is called the *chromatic number*. In 1941, Brooks [7] showed the upper bound on the chromatic number to be $\Delta(G) + 1$, where $\Delta(G)$ is the maximum degree in G. This bound was used for a long time. A more recent result by L. Stacho in 2001 [90] gives a tighter upper bound. The author showed that the chromatic number is always less than or equal to $\Delta_2(G) + 1$ where $\Delta_2(G)$ is the largest degree of any node v in G, such that v is adjacent to a node whose degree is at least as big as its own.

Let us consider an example. In Fig. 4.1, two simple 4 node networks are shown. For the network shown in Fig 4.1.(a), both Brooks' and Stacho's upper bounds give a value of 3. However, for the network shown in Fig. 4.1.(b), using Brooks' upper bound on the chromatic number, we get a value of $\Delta(G) + 1 = 3 + 1 = 4$, while using Stacho's we get a value of

[2]Note that one node in the conflict graph represents one particular *lightpath* of an SLD and not the SLD itself. In other words, each scheduled lightpath demand SLD_i is represented by n_i nodes in the conflict graph which are all adjacent to each other.

$\Delta_2(G) + 1 = 1 + 1 = 2$. We can easily see that nodes 1, 2, and 4 can be colored with one color and node 3 with a second color.

According to Stacho's upper bound, it is evident that graphs with smaller values of $\Delta_2(G)$ give smaller upper bounds for the chromatic number. Note that a routing solution X obtained by solving the routing subproblem corresponds to exactly one conflict graph $CG(X)$ on which we solve the graph coloring problem. If we take into consideration upon constructing routing solution X that we wish to minimize its corresponding value for $\Delta_2(CG(X))$, we may attain a routing scheme whose corresponding conflict graph will need fewer colors to perform graph coloring successfully. This also means that we need fewer wavelengths to perform a successful wavelength assignment. Accordingly, the optimization criteria or evaluation function used by the TS_{cn} algorithm to evaluate a routing solution X is the minimization of the upper bound on the chromatic number of its corresponding conflict graph (i.e. $min\ (\Delta_2(CG(X)) + 1)$).

The TS_{cn} Algorithm

A description of the tabu search algorithm TS_{cn} proposed for the routing subproblem of scheduled lightpath demands follows. As in [52], we first compute the K-shortest paths between each source-destination pair of each SLD using Eppstein's algorithm [25]. K can be set to various values. If we set K to a larger value, the solution obtained will probably need less wavelengths but the physical paths used to route the SLDs will probably be longer. This may present a problem if delay is an issue. On the other hand, if K is set to a smaller value, the physical paths will be restricted to only a few of the shortest paths. As a result, the number of wavelengths needed to successfully route the SLDs will most likely be larger.

Recall that we have given a graph $G = (V, E)$ and a set of M Scheduled Lightpath Demands (SLDs) each represented by a tuple $(s_i, d_i, n_i, \alpha_i, \omega_i)$, where s_i is the source, d_i is the destination, n_i is the number of requested lightpaths, α_i is the set-up time, and ω_i is the tear-down time of the SLD. For simplification purposes, the authors of [52] assume that the *group lightpath constraint* applies, i.e. all the lightpaths of a particular SLD are routed on the same path. The same will be assumed here for easier comparison of the mentioned algorithms. A potential routing solution X is represented by a vector of M integers, $X = (x_1, \ldots, x_M)$, where $x_i \in \{1, \ldots, K\}, i = 1, \ldots, M$, represents the path used by SLD_i. If the integer representing the path of a specific SLD is set to 1, that means that that particular SLD is routed on the shortest path from its source to destination. If it is set to 2, then that SLD is routed on the second shortest path from source to destination, and so on up to the K^{th} shortest path. The TS_{cn} algorithm initially routes all the SLDs in τ on their shortest paths in

G.

Neighboring solutions with respect to the current one are all those where one and only one SLD is routed on a different route. Instead of selecting a large number of neighbors at random as in [52] and evaluating them, we suggest a more directed neighborhood reduction technique. This technique drastically reduces the size of the neighborhood and yet helps obtain solutions of good quality. First we construct the conflict graph $CG(X)$ of the current solution X and then find the set of nodes $L(X)$ which determine $\Delta_2(CG(X))$. That is, we find the one or more nodes which have the largest degree in $CG(X)$, subject to the fact that they are adjacent to a node whose degree is at least as big as their own. Recall that the nodes in the conflict graph represent individual lightpaths and not SLDs. Since we are routing all the lightpaths of a particular SLD on the same path (i.e. they all have the same degree[3]), either all or none of the lightpaths of a particular SLD are in $L(X)$. As a result, we can easily reduce the set of lightpaths $L(X)$ to their corresponding set of SLDs $L_{SLD}(X)$ where $|L_{SLD}(X)| \leq |L(X)|$. The number of SLDs in $L_{SLD}(X)$ is usually fairly small. Instead of evaluating a huge number of neighboring solutions, we evaluate only $|L_{SLD}(X)|$ neighbors. The $|L_{SLD}(X)|$ neighbors are obtained by randomly rerouting each SLD in $L_{SLD}(X)$.

To determine the best neighboring solution which will pass into the next iteration, we create a conflict graph for each neighboring solution and find its corresponding upper bound on the chromatic number. In other words, we find $\Delta_2(CG(X)) + 1$ for each neighbor X. The neighboring solution with the lowest upper bound is passed into the next iteration and becomes the new current solution. If this solution is better than the incumbent solution, the incumbent solution is updated. Such an evaluation function is the motivation for the neighborhood reduction technique. Namely, if we reroute the SLDs which determine $\Delta_2(CG(X))$ (i.e. the SLDs in $L_{SLD}(X)$) instead of rerouting SLDs at random, there is a greater chance that we might improve the upper bound and pass a better solution into the next iteration. Of course, this does not guarantee that a better solution cannot be found by rerouting a series of SLDs not included in set $L_{SLD}(X)$. However, this is an approximation algorithm in which a trade off between execution time and potential solution quality must be made.

A few extra features of the algorithm are as follows. For diversification purposes, if there is no improvement after a certain number of iterations, we take a random number of SLDs and randomly reroute them. If at some point no neighbor can be rerouted (basically, they have all been rerouted and are on the tabu list), we reroute all the SLDs with the maximum

[3]For example, if SLD_1 with $n_1 = 3$ lightpaths is adjacent to SLD_2 and SLD_3 with $n_2 = 7$ and $n_3 = 5$ lightpaths respectively, all three nodes representing lightpaths of SLD_1 have a degree of $n_2 + n_3 + (n_1 - 1) = 7 + 5 + (3 - 1) = 14$ in the conflict graph. $n_1 - 1$ is added because each lightpath of SLD_1 is adjacent to all the other lightpaths of SLD_1 except itself.

degree in the conflict graph, (i.e. $\Delta(CG(X))$, not $\Delta_2(CG(X))$). If this solution is not on the tabu list, it becomes the new current solution. If it is on the tabu list, we take a random number of SLDs and randomly reroute them. In addition to the tabu list which records the last change made in the form of $(SLD_i, P(SLD_i))$, where SLD_i is a number ranging from 1 to M and $P(SLD_i)$ is a number ranging from 1 to K, we separately record the SLD which was last changed. Rerouting this SLD on *any* path is forbidden in the following iteration.

The pseudocode of TS_{cn} is shown in Fig. 4.2.

TScn
Input and initialization:
$G = (V, E)$;
$\tau = \{SLD_1, \ldots, SLD_M\}$, where $SLD_i = (s_i, d_i, n_i, \alpha_i, \omega_i), i = 1, \ldots, M$; *// the set of SLDs*
K; *// the number of K-shortest paths*
//initial routing solution with all paths set to 1
$X_0 = (x_1^0, \ldots, x_M^0), x_i^0 := 1, i = 1. \ldots, M$;
Find $\Delta_2(CG(X_0))$ and the corresponding SLDs $L_{SLD}(X_0) = \{SLD_{r_1}, \ldots, SLD_{r_s}\}, r_i \in \{1, \ldots, M\}, i = 1, \ldots, s$;
$X := X_0$;*//incumbent solution*
$\Delta_2 := \Delta_2(CG(X_0))$;*//fitness of incumbent solution*
$Tabulist := \{\}, i := 0, itWOImprovement := 0$;

Begin:
//iterations
while $i <$ desired number of iterations **do**
  $X_{it} := \{\}$, $\Delta_2(CG(X_{it})) := \infty$, $L_{SLD}(X_{it}) := \{\}$;
  for j in $1, \ldots, |L_{SLD}(X_i)|$ **do**
    $x^i_{r_j}{}' :=$ random number in $\{1, \ldots, K\} \backslash x^i_{r_j}$ except for that forbidden by tabu list;
    $X_i' := (x_1^i, \ldots, x^i_{r_j-1}, x^i_{r_j}{}', x^i_{r_j+1}, \ldots, x_M^i)$;
    Find $\Delta_2(CG(X_i'))$ and $L_{SLD}(X_i')$;
    if $\Delta_2(CG(X_i')) < \Delta_2(CG(X_{it}))$ **then**
      $X_{it} := X_i'$, $\Delta_2(CG(X_{it})) := \Delta_2(CG(X_i'))$, $L_{SLD}(X_{it}) := L_{SLD}(X_i')$;
    end if
  end for
  if $\Delta_2(CG(X_{it})) == \infty$ **then**
    //all neighbors are on the tabu list
    Find all nodes with max degree in conflict graph of solution X_i (i.e. $\Delta(CG(X_i))$) and randomly reroute them. If this is on tabu list, choose a random number of SLDs and randomly reroute them;
  else
    $X_i := X_{it}$, $\Delta_2(CG(X_i)) := \Delta_2(CG(X_{it}))$, $L_{SLD}(X_i) := L_{SLD}(X_{it})$;
  end if
  Update tabu list;
  if $\Delta_2(CG(X_i)) < \Delta_2$ **then**
    $X := X_i$, $\Delta_2 := \Delta_2(CG(X_i))$;
  else
    $itWOImprovement := itWOImprovement + 1$;
  end if
  if $itWOImprovement \geq$ allowed no. of iterations without improvement **then**
    Select a random number of SLDs and randomly reroute them;
  end if
  $i := i + 1$;
end while
End

Figure 4.2: Pseudocode of the TS_{cn} algorithm.

After solving the routing subproblem with the TS_{cn} algorithm, we use the GGC graph coloring algorithm [43] for wavelength assignment. The computational results are presented in Section 4.6.

4.3.3 Complexity Analysis

For better insight, we examine the computational complexity of the TS_{cg}/GGC and TS_{cn}/GGC algorithms. Both tabu search algorithms use Eppstein's algorithm for computing the k-shortest paths, run the desired number of iterations of their respective tabu search algorithms, and then use the GGC algorithm for wavelength assignment. As a result, the computational complexity of the TS_{cg}/GGC and TS_{cn}/GGC algorithms differ only with respect to the operations performed in each iteration of the tabu search algorithms. Eppstein's algorithm for the K-shortest paths with time complexity $O(|E| + |V| \log |V| + K))$ is run for each of the M SLDs. The GGC algorithm is an improvement algorithm which is run for a desired number of iterations where each iteration has a worst case time complexity of $O(|V|^2)$. The complexity analysis of the iterations of the respective tabu search algorithms follows.

In each iteration of the TS_{cg}/GGC algorithm, each neighboring solution is evaluated by finding the highest congestion on any of the $|E|$ links. The congestion on edge $e \in E$ is computed by sorting the set-up and tear-down times of the SLDs routed over e and then finding the time interval in which the maximum number of lightpaths are active. Sorting takes $O(M \log M)$ time. Finding the highest congestion takes $O(M)$ time since the number of time intervals must be $\leq 2M$. It follows that finding the highest congestion over all edges takes $O(|E|M(\log M + 1))$ time. Time complexity analysis for some of these steps was developed in [49] for their Simulated Annealing algorithm for fault-tolerant RWA SLD. If Nbr is the neighborhood size, each of the Nbr neighbors is evaluated in $O(|E|M(\log M + 1))$ time in each iteration.

The TS_{cn}/GGC algorithm, on the other hand, evaluates each neighboring solution X by constructing a conflict graph $CG(X)$ and then finding the upper bound on the chromatic number, $\Delta_2(CG(X)) + 1$, of the conflict graph. The conflict graph can be constructed in $O(M^2)$ time. $\Delta_2(CG(X))$ and the corresponding neighborhood $L_{SLD}(X)$, can be found in $O(M^2)$. It follows that the complexity of evaluating a neighboring solution is $O(M^2)$. Since the neighborhood is adaptive, the size of the neighborhood is not constant. The upper bound on the number of neighbors is M. This occurs only if the conflict graph $CG(X)$ is a complete graph. However, empirical testing indicates that the neighborhood size is often drastically smaller than M (see Section 4.6, Table 4.5), even when M is large. The neighborhood size could also be additionally upper bounded by a constant, say value Nbr used by TS_{cg}/GGC, so that the number of neighbors evaluated in each iteration is $\min\{L_{SLD}(X), Nbr\}$, where each evaluation is performed in $O(M^2)$ time. The numerical results in Section 4.6 indicate that TS_{cg}/GGC is significantly slower than TS_{cn}/GGC for the cases tested.

4.4 Edge and Time Disjoint Path Algorithms

4.4.1 DP_RWA_SLD

In order to solve the routing and wavelength assignment problem of a set of scheduled lightpath demands, we propose an algorithm motivated by a routing and wavelength assignment algorithm for *static* lightpath demands suggested in [57]. This algorithm, called *Greedy_EDP_RWA*, creates a partition $\tau_1, \ldots, \tau_k$ of a set of static lightpath demands[4] $\tau = \{(s_i, d_i), \ldots, (s_M, d_M)\}$, where $s_i, d_i \in V, i = 1, \ldots, M$. Each element of the partition is composed of a subset of lightpath demands which can be routed on mutually edge disjoint paths in G and hence can be assigned the same wavelength. The length of each path is upper bounded by a value h set in [57] to $\max(diam(G), \sqrt{|E|})$. The justification for setting h to this value is given in [44]. The number of distinct wavelengths needed to successfully perform RWA corresponds to the number of elements in the partition.

To solve the RWA SLD problem, we propose a fast algorithm using some of the ideas introduced above. Routing and wavelength assignment are solved simultaneously based on the idea of finding a partition $\tau_1, \ldots, \tau_k$ of the set of *scheduled* lightpath demands τ where each element $\tau_i, i \in 1, \ldots, k$, of the partition is composed of SLDs routed over 'disjoint' paths. Here, 'disjoint' paths include not only *edge* disjoint paths as in *Greedy_EDP_RWA*, but *time* disjoint paths as well. Two paths that are disjoint in time may be routed using the same physical edges. The lengths of the paths are upper bounded by a value h. We will refer to this algorithm as *DP_RWA_SLD*, where DP stands for *Disjoint Paths*.

The *DP_RWA_SLD* algorithm first sorts the SLDs in τ in decreasing order of the number of lightpaths each SLD requests. The reason for this will be discussed later. A partition of τ is then created in the following manner. The first SLD from the sorted set of demands is routed on its shortest path in G. This SLD and its corresponding path are placed in τ_1 and removed from τ. Subsequent attempts are made to route the remaining requests in τ as follows. For each new SLD considered, the edges of the paths of those SLDs already in τ_1 with which the new SLD overlaps in time are deleted from G. The resulting graph is referred to as G'. The new SLD is now routed on its shortest path in G'. If this routing is successful (i.e. there exists such a path in G' whose length is $\leq h$), the new SLD is added to τ_1 and removed from τ. Otherwise, it remains in τ. After attempting to route all the SLDs in τ, we are left with a set of demands routed on mutually disjoint paths in τ_1 and a set of unrouted demands in τ. This entire procedure is iteratively repeated on the SLDs remaining

[4]Here, each static lightpath demand represents a single lightpath which is to be set up permanently. As a result, each demand is defined only by its source and destination nodes.

DP_RWA_SLD
Input and initialization:
$G = (V, E)$;
$\tau = \{SLD_1, \ldots, SLD_M\}$, where $SLD_i = (s_i, d_i, n_i, \alpha_i, \omega_i), i = 1, \ldots, M$;*//the set of SLDs*
$h = \max(diam(G), \lfloor \frac{1}{2}|E| \rfloor)$;
$\lambda = 0$;*//the number of wavelengths*
$i := 0$;*//the number of elements in the partition*
Begin:
Sort the SLDs in τ in decreasing order of their corresponding values of n_i. Sort requests with the equal values of n_i in decreasing order of the lengths of their shortest paths in G (if more than one request has the same length place them in random order);
while τ is not empty **do**
  $i := i + 1$;
  $\tau_i = \{\}$;
  $P_{\tau_i} = \{\}$; *//paths of SLDs in* τ_i
  for each $SLD_j \in \tau$ in the sorted order **do**
    $G' = G$;
    for each $SLD_k \in \tau_i$ **do**
      if $(\alpha_j \leq \alpha_k \leq \omega_j)$ or $(\alpha_k \leq \alpha_j \leq \omega_k)$ **then**
        //$SLD_j \in \tau$ *and* $SLD_k \in \tau_i$ *overlap in time*
        Remove from G' all edges in $P(SLD_k)$;
      end if
    end for
    Find shortest path $P(SLD_j)$ for SLD_j in G';
    if the length of $P(SLD_j)$ is $\leq h$ **then**
      Add $P(SLD_j)$ to P_{τ_i} and SLD_j to τ_i;
    end if
  end for
  W_i= max value of n_j of any $SLD_j \in \tau_i$;
  For each SLD in τ_i, assign to their corresponding lightpaths wavelengths $(\lambda + 1), \ldots$ up to $(\lambda + W_i)$ if needed;
  $\lambda := \lambda + W_i$;
  $\tau := \tau \backslash \tau_i$;
end while
End

Figure 4.3: Pseudocode of the DP_RWA_SLD algorithm.

in τ to create the other elements of the partition, $\tau_2, \ldots, \tau_k$, until all the demands in τ are successfully routed.

Since we are creating a partition of SLDs (not individual lightpaths) we cannot assume that only one wavelength is needed for each element of the partition. Since all the SLDs in τ_i are mutually disjoint, their respective lightpaths can be assigned the same set of wavelengths. On the other hand, each individual lightpath of a particular SLD must be assigned a different wavelength since they are all routed on the same path. It follows that the number of wavelengths W_i which must be assigned to τ_i is the maximum number of lightpaths any SLD included in τ_i requests. Wavelength assignment is performed in the following manner. For each SLD in τ_1, its corresponding lightpaths are assigned wavelengths 1, 2, ...up to W_1 if necessary. The lightpaths routed in τ_2 are assigned wavelengths $\{(W_1+1), \ldots, (W_1+W_2)\}$, the lightpaths in τ_3 are assigned wavelengths $\{((W_1 + W_2) + 1), \ldots, ((W_1 + W_2) + W_3)\}$, and so on. Generally speaking, each element $\tau_i, i = 1, \ldots, k$ is assigned wavelengths $\{(\sum_{t=0}^{i-1} W_t + 1), \ldots, (\sum_{t=0}^{i-1} W_t + W_i)\}$, where $W_0 = 0$.

This method of wavelength assignment is the motivation for sorting the SLDs in τ in decreasing order of the number of lightpaths each SLD requests. Recall that the number of wavelengths W_i which must be assigned to τ_i is the maximum number of lightpaths any SLD included in τ_i requests. If such is the case, it is evident that it is more desirable to route SLDs

Table 4.1: An example of a set of scheduled lightpath demands

SLD_i	s_i	d_i	n_i	α_i	ω_i
SLD_1	4	3	5	1:00	6:00
SLD_2	4	2	10	2:00	6:00
SLD_3	4	1	9	2:00	7:00
SLD_4	1	3	7	1:00	2:00

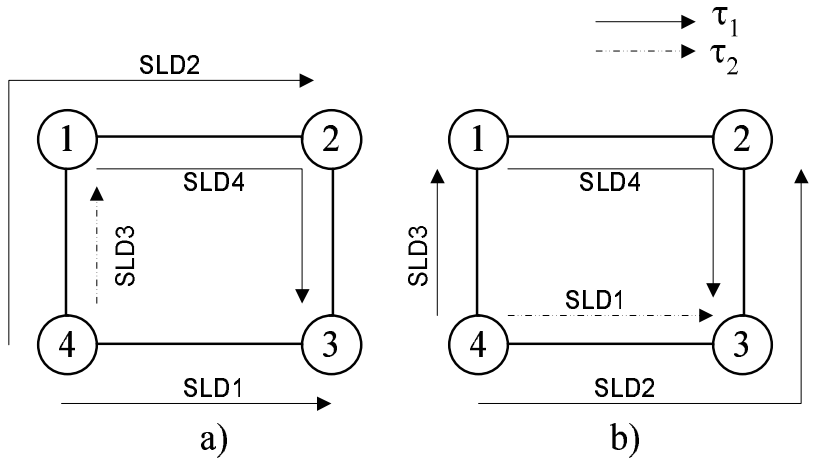


Figure 4.4: An example of a partition of a set of SLDs τ obtained using the DP_RWA_SLD algorithm (a) without sorting the SLDs and (b) with sorting the SLDs.

which request a large number of lightpaths (i.e. the requests with high traffic demands) in the same element of the partition. In most cases, this will lead to a smaller number of total wavelengths assigned, as will be demonstrated on an example. This also means that high traffic demands are routed on mutually edge/time disjoint paths. We can intuitively see that this will reduce congestion as opposed to routing high traffic on the same path at the same time.

Furthermore, the SLDs with the same number of lightpaths are sorted in decreasing order of the lengths of their corresponding shortest paths in G. This is done since SLDs which have longer shortest paths are generally harder to route and should therefore be routed when more edges are available. Related work is given in [86]. If there are multiple SLDs with the same number of lightpaths and the same shortest path length, they are placed in random order.

To demonstrate the benefit of sorting the SLDs before creating a partition of τ, a short example is considered. Suppose the set of SLDs in Table 4.1 and the physical network shown in Fig. 4.1.(a). Let the upper bound h on the length of a lightpath to be set to

2. The lightpaths of SLD_1, SLD_2, and SLD_3 all overlap in time, while the lightpaths of SLD_4 are only in time conflict with those of SLD_1. Suppose we create a partition of τ in the order in which the SLDs are shown in Table 4.1. In that case, SLD_1, SLD_2 and SLD_4 could be routed in the first element of the partition τ_1, while SLD_3 would require a second element as shown in Fig. 4.4.(a). Such a partition would partition would require $W_1 + W_2 = \max(n_1, n_2, n_4) + \max(n_3) = 10 + 9 = 19$ wavelengths to perform wavelength assignment. Now consider routing the SLDs in descending order of their requested lightpaths, i.e. $\{SLD_2, SLD_3, SLD_4, SLD_1\}$. This could result in a partition as follows: $\tau_1 = \{SLD_2, SLD_3, SLD_4\}$ and $\tau_2 = \{SLD_1\}$ shown in Fig. 4.4.(b). Such a partition would require $W_1 + W_2 = \max(n_2, n_3, n_4) + \max(n_1) = 10 + 5 = 15$ wavelengths.

The pseudocode of DP_RWA_SLD is shown in Fig. 4.3.

4.4.2 DP_RWA_SLD*

A related version of the DP_RWA_SLD algorithm is also proposed, referred to as $DP\_RWA\_SLD^*$. After creating an element of the partition τ_i, a second attempt at routing into τ_i the SLDs remaining in τ is executed. The basic idea is the following. After creating each element of the partition τ_i and assigning up to W_i wavelengths to each of the lightpaths of the SLDs included in τ_i, we can see that there may be several SLDs that require less than W_i wavelengths. The edges on paths used by these SLDs could be utilized by routing other SLDs using the wavelengths assigned to τ_i but not used on these particular edges. In other words, we want to "fill up" τ_i by fully utilizing the set of wavelengths already assigned to it.

This is best shown on an example. Suppose we created an element τ_i which is assigned $W_i = 10$ wavelengths. Now suppose demand SLD_j routed in τ_i requests 4 lightpaths (i.e. $n_j = 4$). These lightpaths are assigned wavelengths $(\sum_{t=0}^{i-1} W_t + 1)$, $(\sum_{t=0}^{i-1} W_t + 2)$, $(\sum_{t=0}^{i-1} W_t + 3)$ and $(\sum_{t=0}^{i-1} W_t + 4)$. Each edge on path $P(SLD_j)$ could be used to route any SLD which demands $(W_i - n_j) = 10 - 4 = 6$ lightpaths or less even if it overlaps in time with SLD_j. These lightpaths would simply be assigned wavelengths $(\sum_{t=0}^{i-1} W_t + 5)$, $(\sum_{t=0}^{i-1} W_t + 6), \ldots$ up to $(\sum_{t=0}^{i-1} W_t + 10)$ if necessary.

To successfully execute this modification, the following steps are added to algorithm DP_RWA_SLD giving rise to $DP\_RWA\_SLD^*$. After creating an element τ_i and assigning W_i wavelengths in the same way as DP_RWA_SLD (i.e. one run of the **while** loop), we try and route the SLDs remaining in τ a second time. As before, to route $SLD_j \in \tau$ in τ_i we start with graph G and check to see if it is in time conflict with any of the SLDs already routed in τ_i. For the SLDs which are in time conflict with SLD_j *and* request more than $(W_i - n_j)$ lightpaths, we delete the edges of their corresponding paths from G creating

DP_RWA_SLD*
Input and initialization:
$G = (V, E)$;
$\tau = \{SLD_1, \ldots, SLD_M\}$, where
$SLD_i = (s_i, d_i, n_i, \alpha_i, \omega_i), i = 1, \ldots, M$;*//the set of SLDs*
$h = \max(diam(G), \lceil \frac{D}{|E|} \rceil)$;
$\lambda = 0$;*//the number of wavelengths*
$i := 0$;*//the number of elements in the partition*
$fillingUp$:= false; *//this indicates if we are starting to create a partition or "filling it up"*
Begin:
Sort the SLDs in τ in decreasing order of their corresponding values of n_i. Sort requests with the equal values of n_i in decreasing order of the lengths of their shortest paths in G (if more than one request has the same length place them in random order);
while τ is not empty **do**
  if $fillingUp ==$ false **then**
    Run one while loop of the DP_RWA_SLD algorithm;
    $fillingUp :=$ true;
  else
    for each $SLD_j \in \tau$ in the sorted order **do**
      $G' = G$
      for each $SLD_k \in \tau_i$ **do**
        if $(\alpha_j \leq \alpha_k \leq \omega_j)$ or $(\alpha_k \leq \alpha_j \leq \omega_k)$ **then**
          //$SLD_j \in \tau$ and $SLD_k \in \tau_i$ overlap in time
          if $n_k > W_i - n_j$ **then**
            Remove from G' all edges in $P(SLD_k)$;
          end if
        end if
      end for
      Find shortest path $P(SLD_j)$ for SLD_j in G';
      if the length of $P(SLD_j)$ is $\leq h$ **then**
        Add $P(SLD_j)$ to P_{τ_i} and SLD_j to τ_i;
        Find the max wavelength $W_{max}(P(SLD_j))$ used by any SLD in τ_i which uses any of the edges in $P(SLD_j)$ and is in time conflict with SLD_j;
        Assign to SLD_j the wavelengths $(W_{max}(P(SLD_j)) + 1), \ldots, (W_{max}(P(SLD_j)) + n_j)$;
      end if
    end for
    $fillingUp :=$ false;
    $\tau := \tau \backslash \tau_i$;
  end if
end while
End

Figure 4.5: Pseudocode of the $DP\_RWA\_SLD^*$ algorithm.

G'. The edges of those paths whose SLDs request $(W_i - n_j)$ or less lightpaths remain in G' even though they are in time conflict with SLD_j.

SLD_j is then routed on its shortest path $P(SLD_j)$ in G'. If the routing is successful (i.e. there exists such a path and its length is $\leq h$), SLD_j is added to τ_i and removed from τ. In order to assign wavelengths to the lightpaths of SLD_j, we do the following. We check all the edges in path $P(SLD_j)$ and determine the highest wavelength $W_{max}(P(SLD_j))$ used on any of these edges by an SLD in τ_i which overlaps in time with SLD_j. We then assign wavelengths $(W_{max}(P(SLD_j)) + 1), \ldots, (W_{max}(P(SLD_j)) + n_j)$ to the n_j lightpaths of SLD_j. Note that $W_{max}(P(SLD_j))$ is the highest wavelength assigned to some demand $SLD_k \in \tau_i$ whose path overlaps with $P(SLD_j)$ and can therefore be written as $(\sum_{t=0}^{i-1} W_t + n_k)$. Since prior to routing SLD_j, we deleted from G all those edges used by SLDs in time conflict with SLD_j requesting more than $(W_i - n_j)$ lightpaths, we can be certain that $n_k \leq W_i - n_j$. It follows that $W_{max}(P(SLD_j)) + n_j = \sum_{t=0}^{i-1} W_t + n_k + n_j \leq \sum_{t=0}^{i-1} W_t + W_i$. This proves that we have not assigned to SLD_j any wavelength aside from the W_i wavelengths already assigned to τ_i.

The pseudocode of $DP\_RWA\_SLD^*$ is shown in Fig. 4.5.

4.4.3 Complexity Analysis

The computational complexity of the DP_RWA_SLD and $DP\_RWA\_SLD^*$ algorithms follows. The DP_RWA_SLD algorithm first finds the all-pairs shortest paths between nodes in the physical network using Floyd's algorithm [26] in $O(|V|^3)$ time. The M SLDs are then sorted in $O(M \log M)$ time. The while loop runs $O(M^2|V|^2)$ time giving us a final complexity of $O(|V|^3 + M \log M + M^2|V|^2)$. In the $DP\_RWA\_SLD^*$ algorithm, the while loop is run twice as many times as in DP_RWA_SLD which still yields the same complexity. The complexity of the DP_RWA_SLD and algorithms is not comparable to that of the TS_{cg}/GGC and TS_{cn}/GGC algorithms since the former are constructive heuristics which end deterministically, while the later are improvement heuristics which can be terminated at any time and still obtain a feasible solution. However, numerical results (see Section 4.6) indicate that in order to obtain good solutions using the tabu search algorithms, a fair number of iterations need to be run resulting in execution times drastically longer than those of the greedy algorithms.

4.5 Lower Bounds

Since the algorithms considered in this thesis are heuristics which obtain upper bounds on the minimal objective function values, it is useful to have good lower bounds in order to assess the quality of the sub-optimal solutions. A simplistic lower bound on the number of wavelengths needed to perform successful routing and wavelength assignment on a set of scheduled lightpath demands such that the *group lightpath constraint* is satisfied is

$$W_{n_{max}}^{LB} = \max_{i=1,\ldots,M} \{n_i\}. \tag{4.1}$$

This represents the maximum number of lightpaths requested by any SLD in τ. However, this lower bound is not necessarily efficient for a set of lightpath requests highly correlated in time. In [76], a simple lower bound on the number of wavelengths required to set up a regular virtual topology in wavelength routed optical networks is obtained by comparing the fixed logical degree to the maximum physical degree in the network. We further develop this idea of the logical to physical degree ratio to derive a tighter lower bound for the RWA SLD problem as follows.

Let

$$S_s = \{SLD_i | s_i = s, i = 1, \ldots, M\}, \qquad \forall s \in V \tag{4.2}$$

be the set of SLDs whose source node is node s.

Let

$$T_s^S = \{\alpha_i \cup \omega_i | SLD_i \in S_s, i = 1, \ldots, M\}, \qquad \forall s \in V \tag{4.3}$$

be an ordered set of moments in time when some SLD in S_s is either set up and/or some SLD in S_s is torn down. If $T_s^S = \{t_{s_1}, \ldots, t_{s_{|T_s^S|}}\}$, then $t_{s_1} < t_{s_1} \ldots < t_{s_{|T_s^S|}}$ and $|T_s^S| \leq 2|S_s|$.

Let

$$\begin{aligned} TO_{s_j}^S = \{SLD_k \in S_s | [t_{s_j}, t_{s_{j+1}}] \subseteq [\alpha_k, \omega_k]\}, \\ \forall s \in V, \forall j = 1, \ldots, |T_s^S| - 1, \end{aligned} \tag{4.4}$$

be the set of SLDs whose source node is s and are active in time interval $[t_{s_j}, t_{s_{j+1}}]$. This means that all the SLDs in $TO_{s_j}^S$ overlap in time. Furthermore, let $TO_{s_j}^S$ be an *ordered* set with respect to the number of lightpaths requested by each SLD. In other words, if $TO_{s_j}^S = \{SLD_{to_1^{s_j}}, \ldots, SLD_{to_{|TO_{s_j}^S|}^{s_j}}\}$, then $n_{to_1^{s_j}} \leq n_{to_2^{s_j}} \leq \ldots \leq n_{to_{|TO_{s_j}^S|}^{s_j}}$.

Lastly, let Δ_{phy_s} be the out-degree[5] of node s in the physical topology. All the lightpaths of the SLDs in S_s will surely be routed over one of the Δ_{phy_s} outgoing edges adjacent to node s. If the individual lightpaths of a single SLD do not necessarily need to be routed on the same path (i.e. if we relax the *group lightpath constraint*), each individual lightpath can be routed over any one of the Δ_{phy_s} outgoing edges. Lightpaths in S_s which overlap in time, i.e. their respective SLDs are both in at least one set $TO_{s_j}^S$, $j = 1, \ldots, |T_s^S| - 1$, and which are routed over the same physical edge must be assigned different wavelengths. To route and assign wavelengths to the lightpaths of the SLDs in some set $TO_{s_j}^S$, at least one physical link will have

$$\begin{aligned} W_{TO_{s_j}^S}^{LB} = \left\lceil \frac{\sum_{i|SLD_i \in TO_{s_j}^S} n_i}{\Delta_{phy_s}} \right\rceil, \\ \forall s \in V, \forall j \in \{1, \ldots, |T_s^S| - 1\}, \end{aligned} \tag{4.5}$$

lightpaths routed over it and therefore require at least as many wavelengths.

If we consider the lightpaths of the SLDs in set $TO_{s_j}^S$ to represent a logical topology over the physical topology which is constant in the corresponding time interval, $W_{TO_{s_j}^S}^{LB}$ represents the ratio of logical to physical degree of node s in time interval $[t_{s_j}, t_{s_{j+1}}]$. The highest such ratio

$$W_S^{LB} = \max_{s \in V} \max_{1 \leq j \leq |T_s^S| - 1} W_{TO_{s_j}^S}^{LB} \tag{4.6}$$

for any source node in the network over all time intervals is a lower bound on the number of wavelengths needed to perform routing and wavelength assignment for a set of scheduled

[5] According to our problem definition, the physical out-degree is equal to the physical in-degree for each node in V since we assume that each link in the physical topology represents two fibers - one in each direction.

lightpath demands τ. Note that W_S^{LB} is a lower bound for the RWA SLD problem where the *group lightpath constraint* is relaxed. Since imposing such a constraint makes the problem harder, W_S^{LB} is also a lower bound for the constrained problem.

Furthermore, assuming the *group lightpath constraint* does apply, we suggest an alternative lower bound, referred to as $W_S^{LB\prime}$. Let the load of SLD_i be its corresponding number of lightpaths n_i. A lower bound $W_{TO_{s_j}^S}^{LB\prime}$ on the number of wavelengths needed to perform routing and wavelength assignment of the SLDs in set $TO_{s_j}^S$ is the maximum load on any outgoing physical link adjacent to s after performing optimal load balancing of the $|TO_{s_j}^S|$ SLDs over the Δ_{phy_s} links. If $n_i = 1$ for all SLDs in $TO_{s_j}^S$, load balancing is trivial and gives the same lower bound as (4.6). Otherwise, this problem is NP-complete. For very small cases, exhaustive search could be applied. However, for larger cases this is not practical. Since we do not actually need to perform load balancing but are solely interested in the maximum load of the optimal solution, we can use a lower bound on the maximum load, which, in turn, is a lower bound on the number of wavelengths needed. We know that at least

$$N_{s_j}^S = \left\lceil \frac{|TO_{s_j}^S|}{\Delta_{p_s}} \right\rceil, \qquad \forall s \in V, \forall j \in \{1, \ldots, |T_s^S| - 1\} \tag{4.7}$$

SLDs (not individual lightpaths) will surely be routed on at least one physical outgoing link adjacent to s in time period $[t_{s_j}, t_{s_{j+1}}]$. Defined as such, $N_{s_j}^S \leq |TO_{s_j}^S|$. By summing up the load of the $N_{s_j}^S$ SLDs in $TO_{s_j}^S$ with the lightest load, i.e. the lowest number of lightpaths n_i, we obtain a lower bound on the maximum load. Since $TO_{s_j}^S$ is a set of SLDs sorted in nondecreasing order of their corresponding number of SLDs, the lower bound on the number of lightpaths routed over at least one of the outgoing edges of s in time interval $[t_{s_j}, t_{s_{j+1}}]$ is the sum of the number of lightpaths of the first $N_{s_j}^S$ SLDs in $TO_{s_j}^S$. In other words, if $TO_{s_j}^S = \{SLD_{to_1^{s_j}}, \ldots, SLD_{to_{|TO_{s_j}^S|}^{s_j}}\}$, then

$$W_{TO_{s_j}^S}^{LB\prime} = \sum_{i=1}^{N_{s_j}^S} n_{to_i^{s_j}}, \qquad \forall s \in V, \forall j \in \{1, \ldots, |T_s^S| - 1\}. \tag{4.8}$$

It follows that the lower bound on the number of wavelengths needed to perform RWA of a set of scheduled lightpath demands in the case that the *group lightpath constraint* applies is

$$W_S^{LB\prime} = \max_{s \in V} \max_{1 \leq j \leq |T_s^S| - 1} W_{TO_{s_j}^S}^{LB\prime}. \tag{4.9}$$

Note that for some cases, e.g. when one or a few SLDs request a very large number of lightpaths, bounds (4.1) and/or (4.6) may be tighter. As a result, we consider all the mentioned bounds.

The above discussion regarding lower bounds derived by considering SLDs with common source nodes can also be applied to SLDs with common destination nodes. Namely, if SLDs terminate at the same node $d \in V$, they will surely be routed over one of the Δ_{phy_d} in-degree edges adjacent to node d. Let

$$D_d = \{SLD_i | d_i = d, i = 1, \ldots, M\}, \qquad \forall d \in V \tag{4.10}$$

be the set of SLDs whose destination node is d. This is analogous to (4.2) for SLDs with source node s. Sets $T_{d_j}^D$ and TO_d^D representing the time intervals and time overlapping SLDs in D_d can be obtained from (4.3) and (4.4), respectively, by replacing S with D and s with d. $N_{d_j}^D$ can be obtained in the same manner from (4.7). This leads to two additional lower bounds,

$$W_D^{LB} = \max_{d \in V} \max_{1 \le j \le |T_d^D| - 1} W_{TO_{d_j}^D}^{LB} = \max_{d \in V} \max_{1 \le j \le |T_d^D| - 1} \left\{ \left\lceil \frac{\sum_{i | SLD_i \in TO_{d_j}^D} n_i}{\Delta_{phy_d}} \right\rceil \right\} \tag{4.11}$$

and

$$W_D^{LB\prime} = \max_{d \in V} \max_{1 \le j \le |T_d^D| - 1} W_{TO_{d_j}^D}^{LB\prime} = \max_{d \in V} \max_{1 \le j \le |T_d^D| - 1} \left\{ \sum_{i=1}^{N_{d_j}^D} n_{to_i^{d_j}} \right\} \tag{4.12}$$

analogous to (4.6) and (4.9).

The preceding discussion shows a lower bound on the number of wavelengths needed to solve the RWA SLD problem without the *group lightpath constraint* to be

$$W_{LB} = \max\{W_S^{LB}, W_D^{LB}\}. \tag{4.13}$$

For the problem augmented with the *group lightpath constraint*, a tighter lower bound is

$$W'_{LB} = \max\{W_{n_{max}}^{LB}, W_S^{LB}, W_S^{LB\prime}, W_D^{LB}, W_D^{LB\prime}\}. \tag{4.14}$$

In the example given in Table 4.1, supposing the physical topology shown in Fig. 4.1.(a), the lower bound W'_{LB} would be calculated as follows. In this example, τ =$\{SLD_1, SLD_2, SLD_3, SLD_4\}$, $M = 4$, and $V = \{1, 2, 3, 4\}$, while the physical in and out-degree of each node is $\Delta_{phy_i} = 2$, $\forall i \in V$. Lower bound $W_{n_{max}}^{LB} = 10$ represents the maximum number of

lightpaths requested by any SLD in τ. To calculate W_S^{LB} we must find $W_{TO_{s_j}^S}^{LB}$ for each s and j. For $s = 1$, $S_1 = \{SLD_4\}$, while for $s = 4$, $S_4 = \{SLD_1, SLD_2, SLD_3\}$. For $s = 2$ or $s = 3$, these sets are empty since nodes 2 and 3 are not source nodes for any requested SLD. T_1^S = {1:00, 2:00} and T_4^S ={1:00, 2:00, 6:00, 7:00}, while $TO_{1_1}^S = \{SLD_4\}$, $TO_{4_1}^S = \{SLD_1\}$, $TO_{4_2}^S = \{SLD_1, SLD_3, SLD_2\}$, and $TO_{4_3}^S = \{SLD_3\}$. Note that these sets are ordered in nondecreasing order of the number of lightpaths requested by the SLDs in the set. Lower bounds over the source nodes and time intervals are $W_{TO_{1_1}^S}^{LB} = \lceil 7/2 \rceil = 4$, $W_{TO_{4_1}^S}^{LB} = \lceil 5/2 \rceil = 3$, $W_{TO_{4_2}^S}^{LB} = \lceil (5 + 10 + 9)/2 \rceil = 12$ and $W_{TO_{4_3}^S}^{LB} = \lceil 9/2 \rceil = 5$. It follows that $W_S^{LB} = 12$. Furthermore, $N_{1_1}^S = 1$, $N_{4_1}^S = 1$, $N_{4_2}^S = 2$, and $N_{4_3}^S = 1$. It follows that $W_{TO_{1_1}^S}^{LB\prime} = n_4 = 7$, $W_{TO_{4_1}^S}^{LB\prime} = n_1 = 5$, $W_{TO_{4_2}^S}^{LB\prime} = n_1 + n_3 = 5 + 9 = 14$, and $W_{TO_{4_3}^S}^{LB\prime} = n_2 = 10$. This leads to lower bound $W_S^{LB\prime} = 14$. W_D^{LB} and $W_D^{LB\prime}$ are analogously found to be 6 and 10 respectively. It follows that a lower bound for the RWA SLD without the *group lightpath constraint* is $W_{LB} = \max\{12, 6\} = 12$, while $W'_{LB} = \max\{10, 12, 14, 6, 10\} = 14$ gives a lower bound for the constrained version of the problem. In the example in Fig. 4.4.(b), we can see that a routing and wavelength assignment was found with 15 wavelengths, demonstrating the efficiency of the bound for this case.

Table 4.2: Hypothetical U.S. network [52], $\delta = 0.01$, $M = 30$: Avg. no. of wavelengths, avg. iter. in which the best solution was obtained, avg. exec. time per iteration and avg. exec. time to best solution for algorithms TS_{cg}/GGC [52] and TS_{cn}/GGC; Avg. no. of wavelengths and avg. exec. time for algorithms DP_RWA_SLD and $DP\_RWA\_SLD^*$, and lower bound W'_{LB}.

	TS_{cg}/GGC [52]				TS_{cn}/GGC				*Lower bound*
K	Avg. wave-lengths	Avg. iter. found best	Avg. time/iter (ms)	Avg. time to best sol. (ms)	Avg. wave-lengths	Avg. iter. found best	Avg. time/iter (ms)	Avg. time to best sol. (ms)	Avg. W'_{LB}
2	11.12	0.81	395.37	85.74	11.12	3.53	4.48	27.57	
3	10.50	12.25	402.11	4918.09	10.50	2.72	2.73	10.44	
4	10.28	35.38	402.12	14204.34	10.28	9.92	2.69	33.71	
5	10.22	26.63	495.22	10901.78	10.22	10.52	2.59	38.65	9.90
	DP_RWA_SLD				$DP\_RWA\_SLD^*$				
	Avg. wavelengths		Avg. exec. time (ms)		Avg. wavelengths		Avg. exec. time (ms)		
	10.00		0.76		9.90		0.88		

Table 4.3: Hypothetical U.S. network [52], $\delta = 0.8$, $M = 30$: Avg. no. of wavelengths, avg. iter. in which the best solution was obtained, avg. exec. time per iteration and avg. exec. time to best solution for algorithms TS_{cg}/GGC [52] and TS_{cn}/GGC; Avg. no. of wavelengths and avg. exec. time for algorithms DP_RWA_SLD and $DP\_RWA\_SLD^*$, and lower bound W'_{LB}.

	TS_{cg}/GGC [52]				TS_{cn}/GGC				*Lower bound*
K	Avg. wave-lengths	Avg. iter. found best	Avg. time/iter (ms)	Avg. time to best sol. (ms)	Avg. wave-lengths	Avg. iter. found best	Avg. time/iter (ms)	Avg. time to best sol. (ms)	Avg. W'_{LB}
2	14.33	28.48	396.90	11329.26	13.85	77.42	7.66	595.97	
3	13.78	59.92	399.40	37730.77	12.65	13.82	4.21	59.90	
4	12.47	240.12	404.43	98217.44	11.68	36.45	4.03	147.18	
5	11.70	321.03	404.24	130825.00	11.20	96.38	3.89	368.59	10.08
	DP_RWA_SLD				$DP\_RWA\_SLD^*$				
	Avg. wavelengths		Avg. exec. time (ms)		Avg. wavelengths		Avg. exec. time (ms)		
	11.90		0.94		10.63		1.07		

4.6 Analysis of Computational Results

4.6.1 Experimental Method and Numerical Results

The TS_{cg}/GGC [52], TS_{cn}/GGC, DP_RWA_SLD, and $DP\_RWA\_SLD^*$ algorithms for the Routing and Wavelengths Assignment problem of Scheduled Lightpath Demands were all implemented in C++ and run on a PC powered by a P4 2.8GHz processor. The TS_{cg} [52] and the suggested TS_{cn} tabu search algorithms for the routing subproblem were run in combination with the GGC graph coloring algorithm from [43] for wavelength assignment. The source code for the GGC algorithm was provided by the authors. Random numbers were generated using the R250 random number generator [42].

We tested the algorithms using the hypothetical U.S. backbone given in [52]. The network consists of 29 nodes and 44 edges which are assumed to be bidirectional. The weight of an edge represents its physical length. Using a Perl script provided by the authors of [52], 60 sets of M=30 SLDs were generated with time correlation 0.01, and 60 sets with time correlation 0.8. Each SLD could request at most 10 lightpaths. Time correlation closer to 0 means that the SLDs are weakly time correlated while time correlation closer to 1 means that the SLDs generated are strongly time correlated. For exact definition of the time correlation

Table 4.4: Hypothetical U.S. network [52], $M = 30$, WORST cases: Test cases for which the best solution was found in the highest iteration, the corresponding iteration, avg. execution time per iteration and the execution time to best solution for algorithms TS_{cg}/GGC [52] and TS_{cn}/GGC

		TS_{cg}/GGC [52]				TS_{cn}/GGC			
δ	K	Test case	Iteration found best	Avg. time/iter. (s)	Time to best solution (s)	Test case	Iteration found best	Avg. time/iter. (s)	Time to best solution (s)
0.01	2	52	2	0.3921	0.784	54	86	0.0077	0.666
	3	21	158	0.4059	64.137	21	45	0.0047	0.210
	4	21	277	0.4035	111.770	21	141	0.0044	0.624
	5	24	265	0.4012	106.344	21	149	0.0045	0.671
0.8	2	39	1509	0.3965	598.240	39	1348	0.0078	10.467
	3	22	1204	0.4006	482.276	17	219	0.0051	1.112
	4	41	2419	0.4020	972.498	39	457	0.0036	1.649
	5	33	2989	0.4041	1207.822	54	1525	0.0042	6.442

parameter used, refer to [52]. In this thesis, we will refer to this parameter as δ.

As in [52], the TS_{cg}/GGC algorithm was run with a neighborhood size of 200, the length of the tabu list was set to 2 times the neighborhood size and the number of allowed iterations without improvement was set to 150. Regarding the TS_{cn}/GGC algorithm, the size of the neighborhood is not an input parameter since TS_{cn} uses an adaptive neighborhood. The remaining parameters for the TS_{cn}/GGC algorithm were determined experimentally. Since effective tabu tenures, i.e. the length of the tabu list, have been shown to depend on the size of the problem [29], we tested the algorithm with tabu tenures proportional to the number of possible neighboring solutions. Since a neighboring solution with respect to a current one is defined such that one of the M SLDs is routed on a different path, there are $M(K-1)$ possible neighbors. Experimental results indicated that a tabu list of size $M(K-1)/10$ was long enough to disable cycling and short enough so as not to restrict the search. Setting the number of iterations without improvement to a value dependant on the size of the problem also proved effective. Empirical testing also showed that applying diversification every $M(K-1)/3$ iterations helped obtain good results.

Both the TS_{cg}/GGC and TS_{cn}/GGC algorithms were run for 3000 iterations, as in [52], and K ranged from 2 to 5. Since a tabu search algorithm can reach its best incumbent solution in any iteration and then continue running without any improvement (even with di-

versification), we recorded the iteration in which the best solution was first found for each test case for both tabu search algorithms. We also measured the average execution time per iteration and the time it took each tabu search algorithm to reach its best solution. These results, averaged over the 60 test cases, and the average number of wavelengths of the solutions obtained by each of the tabu search algorithms for time correlations 0.01 and 0.8 are shown in Tables 4.2 and 4.3, respectively. The number of wavelengths and execution times for the DP_RWA_SLD and $DP\_RWA\_SLD^*$ algorithms and lower bound W'_{LB} are also shown in Tables 4.2 and 4.3. For further insight regarding execution time, in Table 4.4, the number of iterations and the time it took to reach the best solution by each of the tabu search algorithms for the test case for which they performed *worst* are shown. Note that the results shown regarding the execution times of the the tabu search algorithms do not include the time it takes to subsequently run the GGC algorithm. The average execution times of the GGC algorithm were around 12 and 18 seconds for time correlations 0.01 and 0.8, respectively.

We can see that TS_{cn}/GGC performs better than (or equal to) TS_{cg}/GGC in all cases with respect to solution quality *and* execution time. For test data with time correlation 0.01, the initial solution is often optimal since most of the SLDs do not overlap in time. These test cases, although helpful in showing the benefit of performing RWA considering scheduled lightpath demands as opposed to static lightpath demands, are less effective in comparing the results of RWA SLD algorithms. The results for time correlation 0.8 are much more interesting. The specific test cases where the number of wavelengths differed in the solutions obtained by each of the tabu search algorithms are shown in Fig. 4.6. We can see that TS_{cn}/GGC used used less wavelengths in all cases.

According to Table 4.2, the DP_RWA_SLD algorithm outperforms both tabu search algorithms in combination with the GGC algorithm for time correlation 0.01. For time correlation 0.8, DP_RWA_SLD outperforms TS_{cg}/GGC for cases where $K = 2$, 3, and 4, and outperforms TS_{cn}/GGC for cases where $K = 2$ and 3. DP_RWA_SLD has the shortest execution time among all the mentioned algorithms for all cases. The $DP\_RWA\_SLD^*$ algorithm outperforms TS_{cn}/GGC, TS_{cg}/GGC and DP_RWA_SLD for all values of K in solution quality and both tabu search algorithms in execution time. Since the TS_{cn}/GGC algorithm uses less wavelengths than TS_{cg}/GGC for all test cases, and the $DP\_RWA\_SLD^*$ algorithm uses less wavelengths than the DP_RWA_SLD algorithm in all cases, we compare the results of TS_{cn}/GGC and $DP\_RWA\_SLD^*$. The test cases where the solutions obtained by TS_{cn}/GGC and $DP\_RWA\_SLD^*$ differed for time correlation 0.8 are shown in Fig. 4.7. The TS_{cn}/GGC algorithm algorithm performed better in 4 cases, while the $DP\_RWA\_SLD^*$ algorithm performed better in 14 cases.

Table 4.5: Hypothetical U.S. network [52], $M = 30$: Average neighborhood size for algorithm TS_{cn}/GGC

	Average neighborhood size	
K	$\delta = 0.01$	$\delta = 0.8$
2	0.810	2.035
3	0.711	1.946
4	0.662	1.879
5	0.661	1.825

Since the neighborhood of the TS_{cn}/GGC algorithm is adaptive, we recorded the average neighborhood sizes for the TS_{cn}/GGC algorithm. These results are shown in Table 4.5. We can see that the proposed neighborhood reduction technique dramatically reduces the size of the neighborhood and yet obtains good results. The average neighborhood size for test cases with time correlation 0.01 is less than one since for many of the test cases, solutions can be found where none of the SLDs overlap in both time and space due to the very small time correlation. Such solutions give conflict graphs where none of the nodes representing lightpaths of different SLDs are adjacent, and thus RWA is trivial.

Table 4.6: Hypothetical U.S. network [52], $M = 30$: Avg. physical hop length of the lightpaths in the solutions obtained by algorithms TS_{cg}/GGC [52], TS_{cn}/GGC, DP_RWA_SLD and $DP\_RWA\_SLD^*$

	Time correlation $\delta = 0.01$				*Time correlation* $\delta = 0.8$			
K	$TS_{cg}/$ GGC[52]	$TS_{cn}/$ GGC	$DP\_$ RWA_SLD	$DP\_$ RWA_SLD^*	$TS_{cg}/$ GGC[52]	$TS_{cn}/$ GGC	$DP\_$ RWA_SLD	$DP\_$ RWA_SLD^*
2	3.755	3.828			3.847	3.987		
3	3.888	3.788	3.818	3.819	3.983	4.001	3.980	3.993
4	4.006	3.876			4.468	4.167		
5	4.032	3.912			4.631	4.248		

The average physical hop lengths of the lightpaths established by each of the algorithms are shown in Table 4.6. For test cases with time correlation 0.01, the TS_{cn}/GGC algorithm established shorter lightpaths than the TS_{cg}/GGC algorithm for cases where $K = 3$, 4, and 5. The $DP\_RWA\_SLD^*$ set up shorter lightpaths than the tabu search algorithms for all cases but $K = 2$ for TS_{cg}/GGC and $K = 3$ for TS_{cn}/GGC. For test cases with time correlation 0.8, the TS_{cn}/GGC and $DP\_RWA\_SLD^*$ algorithms were better than TS_{cg}/GGC for $K = 4$ and 5, while the latter performed better for $K = 2$ and 3. The

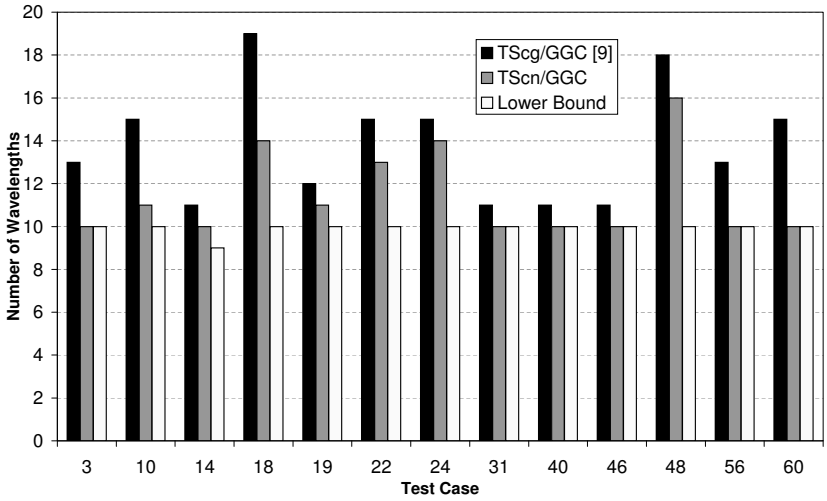


Figure 4.6: Hypothetical U.S. network [52], $\delta = 0.8$, $M = 30$: The number of wavelengths of the solutions obtained by algorithms TS_{cg}/GGC [52] and TS_{cn}/GGC, and lower bound W'_{LB} for the test cases where the number of wavelengths differ.

DP_RWA_SLD algorithm established the shortest lightpaths for both time correlations.

The algorithms were also tested on a reference European core network topology shown in Fig. 3.3 which was designed as part of the COST Action 266 project [37]. This network network consists of 14 nodes and 39 edges. 20 test cases with time correlation $\delta = 0.95$ and $M = 200$ SLDs were generated, where each SLD can request at most 10 lightpaths. The tabu search algorithms were run with $K = 5$. The average number of wavelengths and the average execution times to reach the best solution are shown in Table 4.7. For the European network, all three proposed algorithms significantly outperform the TS_{cg}/GGC algorithm with respect to both the number of wavelengths and execution time[6]. The wavelengths required for the specific test cases are shown in Fig. 4.8. The average neighborhood size for the TS_{cn}/GGC algorithm was 1.540. The average physical hop lengths of the established lightpaths were as follows: 3.503, 3.655, 2.717 and 2.730 for algorithms TS_{cg}/GGC, TS_{cn}/GGC, DP_RWA_SLD and $DP\_RWA\_SLD^*$, respectively. Here, TS_{cg}/GGC outperformed TS_{cn}/GGC, but DP_RWA_SLD and $DP\_RWA\_SLD^*$ again established shorter lightpaths than the tabu search algorithms.

[6]The run time for the GGC algorithm for the test cases generated for the European network was about 220 seconds.

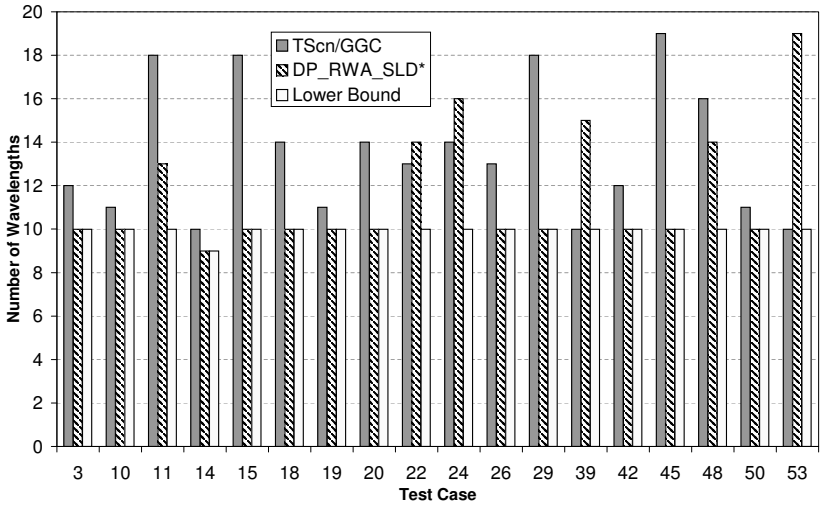


Figure 4.7: Hypothetical U.S. network [52], $\delta = 0.8$, $M = 30$: The number of wavelengths of the solutions obtained by algorithms TS_{cn}/GGC and $DP\_RWA\_SLD^*$, and lower bound W'_{LB} for the test cases where the number of wavelengths differ.

4.6.2 Discussion

All three proposed algorithms give better quality solutions in less time than the TS_{cg}/GGC [52] algorithm for the data tested. The proposed tabu search algorithm, TS_{cn}/GGC, uses less wavelengths than TS_{cg}/GGC and yet evaluates only a few neighbors in each iteration. The very efficient neighborhood reduction technique, in addition to improving the quality of the solutions, drastically reduces the execution time per iteration with respect to the previous art. The time per iteration of the TS_{cn}/GGC algorithm is not only dramatically shorter than that of the TS_{cg}/GGC algorithm, but surprisingly decreases as K increases for the cases tested. One of the reasons for this is that, for this data set, the average neighborhood size decreased as K increased (see Table 4.5). The neighborhood size depends on the topology of the conflict graph and is therefore dependent on K. Although, in general, the neighborhood size does not necessarily decrease as K increases, such was the case for the data instances evaluated in this thesis. Examining the behavior of the algorithm further, we found that when K is small, it occurs more frequently that all neighboring solutions are on the tabu list. In such cases, alternative neighboring solutions outside the reduced neighborhood set are examined until a valid neighbor is found. This slightly increases the run-time of the algorithm.

Another point worth mentioning, regarding the TS_{cn}/GGC algorithm, is that the number of iterations required to reach the best solution is higher when $K = 2$ than when $K > 2$.

Table 4.7: Hypothetical European network [37], $\delta = 0.95$, $M = 200$: Avg. no. of wavelengths, avg. iter. in which the best solution was obtained, avg. exec. time per iteration and avg. exec. time to best solution for algorithms TS_{cg}/GGC [52] and TS_{cn}/GGC; Avg. no. of wavelengths and avg. exec. time for algorithms DP_RWA_SLD and $DP\_RWA\_SLD^*$, and lower bound W'_{LB}.

	TS_{cg}/GGC [52]				TS_{cn}/GGC				*Lower bound*
K	Avg. wave-lengths	Avg. iter. found best	Avg. time/iter (s)	Avg. time to best sol. (s)	Avg. wave-lengths	Avg. iter. found best	Avg. time/iter (s)	Avg. time to best sol. (s)	Avg. W'_{LB}
5	29.00	935.40	1.403	1301.228	22.70	905.00	0.076	69.837	
	DP_RWA_SLD				$DP\_RWA\_SLD^*$				13.05
	Avg. wavelengths		Avg. exec. time (s)		Avg. wavelengths		Avg. exec. time (s)		
	21.80		0.0203		19.45		0.0227		

Since neighborhood reduction is so drastic, the search is too restrictive when K is very small. The search technique is much more effective when K is larger, which is convenient since these are the cases when the problem size is bigger and the corresponding combinatorial optimization problem is harder.

Regarding the proposed greedy algorithms, both DP_RWA_SLD and $DP\_RWA\_SLD^*$ outperform TS_{cg}/GGC in all cases with respect to the number of wavelengths and execution time. These algorithms also establish shorter lightpaths. The DP_RWA_SLD and $DP\_RWA\_SLD^*$ algorithms are easy to implement, give good quality solutions and can be applied to large networks due to their very short execution times. $DP\_RWA\_SLD^*$ is negligibly slower and establishes slightly longer lightpaths than longer lightpaths than DP_RWA_SLD, but performs significantly the number of wavelengths used.

Although the greedy algorithm $DP\_RWA\_SLD^*$ is better on average than the proposed tabu search algorithm TS_{cn}/GGC, for specific test cases this is sometimes not true (see Figure 4.7 and 4.8). An effort was made to determine a pattern in test cases in which the tabu search algorithm performed better than the greedy algorithm, and *vice versa*. However nothing conclusive was found. This is not surprising since both strategies (greedy and tabu) are heuristics and the search trajectory can be unpredictable depending on input data. If the input data in an instance is such that a greedy strategy provides an effective minimization direction, it is possible that nothing better will be obtained by the improvement mechanism of tabu search. Also, the initial solution used in tabu search can sometimes be inefficient

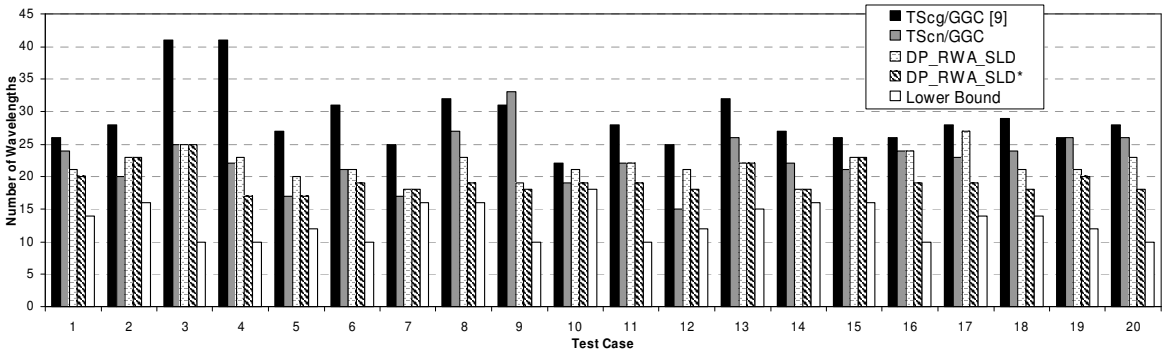


Figure 4.8: Hypothetical European network [37], $\delta = 0.95$, $M = 200$: The number of wavelengths of the solutions obtained by TS_{cg}/GGC [52], TS_{cn}/GGC, DP_RWA_SLD, and $DP\_RWA\_SLD^*$, and lower bound W'_{LB} .

(far away from the optimal solution), in which case it might be difficult to reach a very good suboptimal solution via a restricted neighborhood search. In some other instances, input data can be such that a good initial solution and effective improvements are provided with tabu search strategy, while a greedy strategy lacks flexibility in search directions and ends with an inferior solution. Due to short computational times, for smaller problems both TS_{cn}/GGC and $DP\_RWA\_SLD^*$ could be applied and the better solution selected. For larger problems it might be better to run the greedy algorithm, compare the solution with the available lower bound, and in case of a significant gap between the solution and its lower bound, the tabu search algorithm could be applied in an attempt to improve the solution.

4.7 Summary and Future Work

In order to efficiently utilize resources in wavelength-routed optical networks, it is necessary to successfully solve the problem of Routing and Wavelength Assignment. Scheduled lightpath demands, where the set-up and tear-down times of lightpaths are known *a priori*, could be considered by RWA algorithms in order to utilize the network's resources even further. In this chapter, efficient heuristic algorithms are proposed for the routing and wavelength assignment of scheduled lightpath demands in networks with no wavelength converters. Testing and comparing with an existing algorithm for the RWA SLD problem shows that these algorithms not only provide solutions of better or equal quality, but are dramatically faster. New lower bounds for the RWA SLD problem are also proposed. Further avenues of research will include developing similar algorithms for routing and wavelength assignment

in networks with full or limited wavelength conversion. Networks equipped with a limited number of transmitters and receivers at each node and/or a limited number of wavelengths on each link will also be considered. Furthermore, routing and wavelength assignment algorithms which consider physical layer QoS (Quality of Service) demands, such as target BER (Bit Error Rate) levels, could prove interesting research topics. Fault tolerant RWA and restoration schemes for scheduled lightpaths demands are important issues which could also be addressed.

Chapter 5

Multicast Routing and Wavelength Assignment

In this chapter we consider the problem of Multicast Routing and Wavelength Assignment. Since multicast routing itself is NP-hard, we first study the multicast routing problem independently in Section 5.1 and suggest an algorithm for the delay-constrained multicast routing problem. In Section 5.2, we then study the complete Multicast Routing and Wavelength Assignment problem with static light-tree demands.

5.1 Multicast Routing

Multicast is a mechanism which enables the simultaneous transmission of information to a group of destinations in a network. In other words, it is a technique that *logically* connects a subset of nodes in a network. The development of numerous real-time multimedia applications in the past several years has created an increasing need for this type of distribution of information. Many applications (e.g. video-conferencing, distance education, video-on-demand, and applications in finance) require packets of information to be sent with a certain Quality of Service (QoS). In this thesis, we will discuss one of the most important QoS demands which is the demand for a bounded end-to-end delay from the source to any destination in a multicast session. Real-time applications do not allow the end-to-end delay to exceed a certain delay bound, which represents a measure of the quality of service of that application.

In order to support these real-time applications and their respective QoS demands, networks require efficient multicast routing protocols that provide the necessary Quality of Service while minimizing the use of network resources. The routing algorithms used in these

protocols usually attempt to find a minimum cost tree that includes the source and all the destination nodes, while attempting to satisfy the delay constraint and other QoS demands. Other QoS demands could include the minimum required bandwidth, the maximum allowed packet loss ratio and the maximum delay jitter. The tree topology is most frequently used, since it enables parallel sending of packets to multiple destinations and duplicating the packets is only necessary where the tree branches.

Multicast routing is often reduced to the Minimum Steiner Tree Problem in Graphs (MStTG). Generally, for a given graph $G = (V, E)$, where V is a set of nodes and E is a set of edges, and a given subset of nodes, $D \subseteq V$, a Steiner tree is one which connects all the nodes in D using a subset of edges in E. This tree may or may not include nodes in $V \setminus D$. Nodes in $V \setminus D$ which are included in the Steiner tree are called Steiner nodes. The MStTG problem deals with searching for such a tree that is of minimal weight in a weighted graph. This basically reduces to searching for the set of Steiner nodes that gives the best solution. Since the MStTG problem has been proven to be NP-complete [27], several heuristic algorithms have been developed to solve it suboptimally. Examples of such heuristics are found in [32], [46], [102], and [107].

The MStTG problem can be augmented to include additional constraints giving rise to the constrained MStTG (CMStTG) problem. This section is concerned with delay-constrained multicast routing (the DCMR problem). This problem can be reduced to the Constrained Minimum Steiner Tree Problem in Graphs (CMStTG), where the constraint is the maximum end-to-end delay from the source to any destination. This problem refers to the search for the minimum Steiner tree that satisfies a delay constraint. We propose a heuristic algorithm for solving the CMStTG problem based on the GRASP search method. The algorithm was tested on small and medium sized problems (50 - 100 nodes) from SteinLib ([45]), and the results were compared with the results of the $TS - CST$ tabu search algorithm ([87]) and Kompella et al.'s centralized CST_C algorithm ([46]). SteinLib is a library of test data which includes optimal solutions for Steiner Tree problems and is available on the WWW. Results indicate that the proposed GRASP method is superior to both algorithms in solution quality. Further testing is required to determine more exact performance measures of this heuristic.

The rest of the section is organized as follows. In Section 5.1.1 we formally define the DCMR problem, followed by a short introduction to GRASP in Section 5.1.2. In Section 5.1.3 we describe our GRASP heuristic algorithm for the DCMR problem. We introduce the test problem set and the experimental method in Section 5.1.4. In Section 5.1.5 we summarize the obtained computational results and finish with a summary in Section 5.1.6.

5.1.1 The Delay-Constrained Multicast Routing (DCMR) Problem Model

The communication network is modelled as a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. On the graph G we define the functions $c(i, j)$ and $d(i, j)$, where $c(i, j)$ is the cost of using edge $(i, j) \in E$ and $d(i, j)$ is the delay along edge $(i, j) \in E$. Given is a source node s and a set of destination nodes S, where $\{s\} \cup S \subseteq V$. The upper delay bound on the path from s to any node in S is denoted as Δ. The delay-constrained multicast routing problem (DCMR) searches for a tree $T = (V_T, E_T)$, where $V_T \subseteq V$ and $E_T \subseteq E$, while minimizing the cost of the tree, subject to the following constraints: $\{s\} \cup S \subseteq V_T$ and $D(s, v) < \Delta$ for every $v \in S$, where $D(s, v) = \sum_{i,j} d(i, j)$ for all edges $(i, j) \in E_T$ on the path from s to v in T.

It is important to note that we assume to have centralized information about the network topology. We also assume that the delay of an edge is a constant value which represents the sum of the propagation delay along the edge and the switching delay at the previous node. The cost of an edge is not necessarily proportional to its delay. The cost of an edge can represent various values such as the actual cost or the transfer capacity of the link.

5.1.2 The GRASP Metaheuristic

GRASP (Greedy Randomized Adaptive Search Procedure) is an iterative metaheuristic used in a wide array of combinatorial optimization problems. Every GRASP iteration consists of two phases: a construction phase, followed by a local search phase. The construction phase builds a feasible solution by applying a randomized greedy algorithm. The randomized greedy algorithm builds a solution by iteratively creating a candidate list of elements that can be added to the partial solution and then randomly selecting an element from this list. Creating this candidate list, called the restricted candidate list (RCL), is done by evaluating the elements not yet included in the partial solution with a certain greedy function that depends on the specifications of the problem. Only the best elements according to this evaluation are included in the RCL. The size of this list can be limited either by the number of elements or by their quality with respect to the best candidate element. After every iteration of this greedy algorithm, the restricted candidate list is updated. The construction phase ends when all the elements needed to create a feasible solution are included. This solution is usually of good quality and offers fast local convergence as a result of the *greedy* aspect of the algorithm used. Since this greedy algorithm is *randomized*, exploration of the solution space is diversified.

The solution obtained in the construction phase is not necessarily locally optimal, so a

local search phase is applied. This phase uses a local search algorithm which iteratively replaces the current solution with a better neighboring solution until no better solution can be found. This algorithm can use different strategies for neighborhood evaluation and for moving from one feasible solution to another. It can either search for the best neighboring solution or just choose the *first improving* one.

After applying the desired number of GRASP iterations, the best solution found overall is produced as the final result. Success of a particular GRASP metaheuristic depends on a number of different factors. Some of the most important include the efficiency of the randomized greedy algorithm used, the choice of the neighborhood structure, and the neighborhood search technique. A more detailed description of the GRASP procedure is described in [77]. GRASP algorithms have been used to help solve the Minimum Steiner Tree Problem in Graphs (MStTG) in [58], [59], and [78], along with many other optimization problems. To the best of our knowledge, this method has not been applied to the Constrained MStTG problem or to multicast routing in general.

5.1.3 Description of the $GRASP - CST$ Algorithm

While solving the DCMR problem using our GRASP heuristic, the problem is first reduced to the Constrained Minimum Steiner Tree Problem in Graphs (CMStTG). In this problem, the constraint is the maximum end-to-end delay from the source node to each destination in the multicast group.

It has already been mentioned that for a given weighted graph $G = (V, E)$ and a set of nodes $D \subseteq V$, a minimum Steiner tree is such a tree which connects all the nodes in D using a subset of edges in E that give the minimum total weight. The constrained minimum Steiner tree is such a tree which is of minimum weight while satisfying the given constraint(s). In our problem, we distinguish between one source node s and a group of destination nodes S, so for us $D = \{s\} \cup S$. Nodes in $V \setminus D$, which are included in the constrained Steiner tree, are called Steiner nodes.

Graph Reductions

Before implementing the GRASP method, reducing the size of the graph in accordance with the specifics of the problem is desirable. If we decrease the number of potential Steiner nodes in the graph, the solution space becomes smaller and there are less potential solutions among which to search. We will apply a few of the standard graph reductions described in [105], with a slight modification due to the added delay constraint.

First, we prune the graph of all *non-destination* nodes (nodes in $V \setminus D$) that are of degree 1 since they will surely not be included in the solution. Secondly, we observe that the adjacent edge of every *destination* node that is of degree 1 will always be in the Steiner tree. As a result of this, we can deem the adjacent node of every such destination node as a destination node itself (if it is not already deemed as such). This reduces the size of our problem, since it reduces the number of non-destination nodes among which we have to decide which are to be included in the Steiner tree.

For further reduction, we do the following: for every non-destination node k that is of degree 2 with adjacent nodes i and j, we can replace edges (i, k) and (k, j) from E with one edge (i, j), where $c(i, j) = c(i, k) + c(k, j)$ and $d(i, j) = d(i, k) + d(k, j)$. Node k is then deleted from the graph. If there already exists an edge (i, j) in E, we compare its cost and delay parameters to those of the newly constructed edge. If one of these edges has both a lesser cost and a lesser delay, we can eliminate the other from E. Otherwise, both edges remain in E. This is because for various delay bounds the cheaper edge with the greater delay may not satisfy the delay constraint while the more expensive one might. After performing these reductions, we execute our GRASP search algorithm on the reduced graph.

The $GRASP - CST$ Algorithm

We will refer to our GRASP heuristic as the Greedy Randomized Adaptive Search Procedure - Constrained Steiner Tree ($GRASP - CST$) algorithm. As already mentioned, the GRASP method is an iterative metaheuristic algorithm where each iteration is composed of two phases: the construction phase and the local search phase. The construction phase builds a feasible solution with a randomized greedy algorithm which is further improved by executing a local search algorithm in the local search phase. After executing the desired number of iterations, the best found solution over all the iterations is kept. The efficiency and quality of various GRASP heuristic algorithms vary depending on the design of these two phases.

Potential solutions in our heuristic are potential constrained Steiner trees represented by binary sets consisting of $|V \setminus D|$ bits. Each bit corresponds to a different node in $V \setminus D$. Nodes whose corresponding bits are set to zero in a given configuration are Steiner nodes. Nodes whose corresponding bits are set to 1 are not included in the constrained Steiner tree. Each configuration corresponds to a *potential* constrained Steiner tree because there exists the possibility that for some configurations no constrained Steiner Tree can be found. Such is the case if a configuration leaves the graph unconnected because then no Steiner tree exists. Another possibility is that for a given configuration, we cannot find a Steiner tree that satisfies the given delay bound. We denote the cost of such solutions as infinite.

```
Begin GRASP
//Initialization:
Input nodes V and E from graph G
Input s := source node; S := destination nodes;
s ∪ S = D;
Input α, Δ, GraspIt, RandSeed, ItWithoutImprovement;
Reduce graph G;
//current incumbent solution
X := [x_1 ··· x_{|V\D|}], x_i ∈ [0,1], i = 1,...,|V\D|
C, D := ∞; //cost and delay of the current incumbent sol

//the first iteration of GRASP finds the pure greedy solution (α = 1)
X_pot := ConstructGreedyRandSol(1, RandSeed, Δ);
if a feasible solution exists (Δ can be met) then
   X := TS − CST(ItWithoutImprovement, X_pot, Δ);
   C := cost of DCST(X);
   D := delay of DCST(X);
end if

//the remaining iterations of GRASP use α > 1
i := 0;
while i < GraspIt − 1 do
   X_pot := ConstructGreedyRandSol(α, RandSeed, Δ);
   if a feasible solution exists (Δ can be met) then
      X_pot := TS − CST(ItWithoutImprovement, Xpot, Δ);
      C_pot := cost of DCST(X_pot);
      D_pot := delay of DCST(Xpot);
      if C_pot < C then
         X := X_pot; C := C_pot; D := D_pot;
      end if
   end if
end while
endGRASP
```

Figure 5.1: Pseudocode of the $GRASP - CST$ algorithm

The pseudocode of the $GRASP - CST$ algorithm is shown in Fig. 5.1. Details of the construction and local search phase follow.

The Construction Phase: In order to construct a good starting solution which is feasible with respect to the delay constraint, we do the following: we construct a constrained Steiner tree T which initially consists of only the source node s (i.e. $T = \{s\}$). Next, we create a candidate list by evaluating the cost of adding each destination node not yet included in the solution (nodes in $D \setminus T$) to the existing tree while making sure that the delay from the source to this candidate destination node is less than the given delay bound.

To perform this evaluation, we compute the shortest paths with respect to the cost function from each unconnected destination node to the existing tree (which in this first iteration consists only of the source node) using Dijkstra's single-source shortest paths algorithm ([20]). For each node $i \in D \setminus T$, we denote its shortest path to the existing tree with respect to cost as $ShCPath(i)$. We also compute the shortest paths from each destination node to the source node with respect to the *delay* function. This

```
Begin ConstructGreedyRandSol(α, RandSeed, Δ)
T := s;
X_greedyRand := [x_1 ··· x_|V\D|], x_i ∈ [0,1], i = 1,...,|V\D|;
if the delay of ShDPath(i) > Δ, for any i ∈ S then
   exit the GRASP − CST algorithm; // no feasible solution exists
end if
for all i ∈ S do
   Find ShCPath(i) and ShDPath(i) from s;
   if delay of ShCPath(i) < Δ then
      ConnecPath(i) := ShCPath(i);
      ConnecCost(i) := cost of ShCPath(i);
   else
      ConnecPath(i) := ShDPath(i);
      ConnecCost(i) := cost of ShDPath(i);
   end if
end for
while D ⊈ T do
   BestConnecCost := min(ConnecCost(i)), i ∈ D \ T;
   Make RCL of all i ∈ D \ T where ConnecCost(i) ≤ α · BestConnecCost;
   Node k = random(RCL, RandSeed);
   T = T ∪ ConnecPath(k);
   Update ConnecCost and ConnecPath for all D \ T
end while
for all nodes in T \ D do
   Set their corresponding bits in X_greedyRand to 0;
end for
return X_greedyRand;
endConstructGreedyRandSol
```

Figure 5.2: Pseudocode of the construction phase of $GRASP - CST$

path is denoted as $ShDPath(i)$. These paths can include any unconnected destination or non-destination node in the graph ($V \setminus T$).

Note that the shortest delay paths are computed with respect to the source node s and not the existing tree T. In other words, they are only computed in the first iteration of the construction phase when $T = \{s\}$. This is because our delay constraint is defined as the maximum end-to-end delay from the source to any destination node in the multicast session. These shortest delay paths computed in this first iteration of the construction phase serve as our 'back up' paths when our shortest cost paths violate the delay constraint. The shortest delay path found for each destination node must satisfy the given delay constraint, otherwise no feasible solution exists.

We define the value of adding each unconnected destination node i to the existing tree, as the cost of its respective shortest *cost* path ($ShCPath(i)$) if this path satisfies the delay constraint, otherwise as the cost of its respective shortest delay path ($ShDPath(i)$). We denote this value as $ConnecCost(i)$ and its respective path as $ConnecPath(i)$. We then sort these candidate nodes with respect to the value of these determined connection costs.

To create a restricted candidate list (RCL), we include only those nodes $i \in D \setminus T$ for

which $ConnecCost(i) \leq \alpha \cdot ConnecCost(j)$, where $\alpha \geq 1$ and $j \in D \setminus T$ for which $ConnecCost(j) \leq ConnecCost(k)$, for every $k \in D \setminus T$. If $\alpha = 1$, then the algorithm is pure greedy. This means that only the node(s) with the least connection cost can be in the RCL. If $\alpha > 1$, the RCL can also include other nodes whose connection costs are good, but not necessarily best.

We now choose a candidate node at random from the RCL. We add this chosen node i, $i \in D \setminus T$, along with all the other nodes found along its respective connection path $ConnecPath(i)$ to tree T. We update the connection costs and paths of the remaining unconnected destination nodes ($D \setminus T$) by computing their shortest paths to any of the newly connected nodes. If any of these computed paths improve their existing connection costs while satisfying the delay constraint, their respective connection costs and paths are updated. This procedure ends when all the destination nodes are included in the tree ($D \subseteq T$).

As already mentioned, the greedy aspect of the construction phase provides good solution quality and fast local convergence. The random aspect of the construction phase enables diversified exploration of the solution space. Diversification allows the search procedure to visit various areas of the solution space that may contain the optimal solution. Since the pure greedy algorithm gives high quality average solutions, our GRASP heuristic is designed in such a way that the first iteration of $GRASP-CST$ performs its construction phase with $\alpha = 1$ (pure greedy). The remaining $GRASP - CST$ iterations perform their construction phases with $\alpha > 1$. This is done so that we have a pretty good solution even after the first $GRASP - CST$ iteration and then search for an even better one in the remaining number of iterations, depending on how much execution time we are willing to spend. In other words, since there is a trade off between solution quality and execution time, this method ensures that if in a certain situation it is more important to produce a solution in less time, we can run $GRASP - CST$ for only a few iterations, or even one, and we will still get a reasonably good result.

The pseudocode of the construction phase of $GRASP - CST$ is shown in Fig. 5.2.

The Local Search Phase: Since the feasible solution built in the construction phase is not necessarily locally optimal, applying a local search procedure to find the local optimum is desirable. A better solution might also be close by, but not necessarily local. For this purpose we designed the search phase of $GRASP - CST$ to enable us to explore further than just the local optimum if desired. Local search algorithms usually iteratively replace the current solution with a better neighboring solution until no bet-

```
Begin TS-CST(ItWithoutImprovement, X_pot, Δ)
TabuList := {}; i := 0; iter := 0;
X_TS−CST := X_pot; //the initial solution is that found in the construction phase
C_TS−CST := cost of DCST(X_TS−CST);
D_TS−CST := cost of DCST(X_TS−CST);
X_i := X_pot;
while iter < ItWithoutImprovement do
  C_it := ∞; X_it := {};
  for n = 1, ..., |V \ D| do
    if n is not on TabuList then
      X_neighbor = Flip bit n in X_i;
      Evaluate X_neighbor; //find DCST(X_i)
      C_neighbor := cost of DCST(X_neighbor);
      if unfeasible (Δ cannot be met or graph unconnected) then
        C_neighbor := ∞;
      end if
      if C_neighbor < C_it then
        C_it := C_neighbor; X_it := X_neighbor; n_it := n;
      end if
    end if
  end for
  if C_it = ∞ (no feasible neighbor was found) then
    n_it := i modulo |V \ D| + 1;
    add n_it to TabuList;
  else
    X_i := X_it;
  end if
  if C_it < C_TS−CST then
    X_TS−CST := X_it, C_TS−CST = C_it, D_TS−CST = D_it;
  else
    iter = iter + 1; //increment iterations without improvement
  end if
  Add n_it to TabuList; i := i + 1; //total iterations performed (with or without improvement)
end while
return X_TS−CST;
endTS-CST
```

Figure 5.3: Pseudocode of the local search phase of $GRASP - CST$

ter solution can be found. $GRASP - CST$ enables us to specify the desired 'number of iterations without improvement' so that the search procedure does not necessarily terminate at the first local optimum.

We use the tabu search heuristic algorithm $TS - CST$ suggested in [87] with a modification enabling us to specify the desired number of iterations without improvement. We also modify this algorithm so its initial solution is that obtained in the construction phase of $GRASP - CST$ instead of that suggested in [87]. Here, we will briefly describe the $TS - CST$ algorithm. As already mentioned, potential solutions are represented by binary sets consisting of $|V \setminus D|$ bits. Each bit corresponds to a different node in $V \setminus D$. Nodes whose corresponding bits are set to zero in a given configuration of bits are Steiner nodes. Nodes whose corresponding bits are set to 1 are not included in the constrained Steiner tree. The neighborhood of a certain potential solution includes all those solutions whose binary sets differ from the chosen solution by exactly

one bit. In other words, neighboring solutions are all those solutions obtained by either adding or removing exactly one Steiner node.

In each iteration of the $TS - CST$ algorithm, we start with some current solution, explore all its neighboring solutions and then choose the best neighboring solution which becomes the new current solution in the next iteration. This procedure is called an *move*. $TS - CST$ is a tabu search heuristic, which means that it has a memory structure of variable size called a tabu list which prevents the algorithm from visiting previously visited solutions. Therefore, when exploring the neighborhood of the current solution, those neighboring solutions that are forbidden by the tabu list are ignored. Following every iteration or move, the tabu-list is updated circularly by adding the last performed move (or some attribute of this move) to the list and removing the oldest member. For our purposes, the size of tabu-list is set to one which is enough to prevent the algorithm from oscillating between neighboring solutions.

In order to evaluate each neighboring solution and select the best one to become the current solution in the next iteration, the following is done: First all the non-Steiner non-destination nodes (that is, those nodes whose corresponding bits are set to 1) are eliminated from graph G along with all their adjacent edges. Next, a spanning tree of the remaining graph that attempts to minimize the cost while satisfying the delay constraint is found. This is referred to as the Delay Constrained Spanning Tree (DCST). To find the DCST, a modified version of Prim's Minimum Spanning Tree algorithm ([20]) is used so as to yield a solution in which the end-to-end delay from the source to every destination node is less than the given delay bound Δ. The tree initially consists of only the source node. Then the algorithm subsequently searches for the closest node to the existing tree by examining all its adjacent edges. That edge which is cheapest but whose addition to the tree does not exceed the delay bound is chosen and added to the existing tree. The procedure is finished when all the nodes are included in the tree. The value of each neighboring solution is defined as the cost of the found Delay Constrained Spanning Tree.

In each iteration of the $TS - CST$ algorithm, the cost of the corresponding DCST of each neighbor of the current solution is found (except for those forbidden by the tabu list), and the best among them is chosen to pass into the next iteration. This solution does not necessarily have to improve the current solution. If it does not, we increment the number of iterations performed without improvement. If in some iteration i no feasible solution in the neighborhood of the current solution exists, we choose a non-feasible neighbor in a pseudo-random manner to become the new current solution

Table 5.1: Characteristics of the problem set and the solution quality obtained while simulating the MStTG problem ($\Delta = \infty$)

Probl.	$\|V\|$	$\|D\|$	$\|E\|$	C_{opt}	*GRASP-CST*		*TS-CST*		*CSTc*	
					$\delta_{GRASP\text{-}CST}$ (%)	$D_{GRASP\text{-}CST}$	$\delta_{TS\text{-}CST}$ (%)	$D_{TS\text{-}CSTC}$	δ_{CSTc} (%)	D_{CSTc}
B01	50	9	63	82	0	30	0	30	0	30
B02	50	13	63	83	0	55	0	55	8.43	55
B03	50	25	63	138	0	78	0	78	1.45	78
B04	50	9	100	59	0	58	0	58	0	58
B05	50	13	100	61	0	43	1.64	39	4.92	26
B06	50	25	100	122	0	93	0	93	4.92	65
B07	75	13	94	111	0	51	0	51	0	51
B08	75	19	94	104	0	49	0	49	0	49
B09	75	38	94	220	0	66	0	66	2.27	51
B10	75	13	150	86	0	66	0	66	13.95	78
B11	75	19	150	88	0	65	11.36	91	4.55	75
B12	75	38	150	174	0	66	0	75	0	125
B13	100	17	125	165	0	38	0	38	6.06	53
B14	100	25	125	235	0	70	1.28	80	1.28	70
B15	100	50	125	318	0	81	0	81	2.52	77
B16	100	17	200	127	0	63	7.09	95	7.87	64
B17	100	25	200	131	0	59	1.53	71	2.29	66
B18	100	50	200	218	0	113	0	113	3.67	80

in order to prevent the algorithm from getting stuck. This can be done in various ways. In our algorithm, we chose to flip the n^{th} bit of the current solution, where $n = (i)modulo(|V \setminus D| + 1)$, and this becomes the new current solution in the next iteration. For a more detailed description of the $TS - CST$ algorithm refer to [87]. After running $TS-CST$ for the desired number of iterations without improvement the algorithm ends. If we set the number of iterations without improvement to 1, we find the local optimum. If this number is greater than one, we expand the search beyond the local optimum.

The pseudocode of the local search phase of $GRASP - CST$ is shown in Fig. 5.3.

Table 5.2: Solution quality for $\Delta_1 = min(D_{GRASP-CST}, D_{TS-CST}, D_{CST_C}) + 1$

		GRASP-CST			***TS-CST***			***CST_C***		
Probl.	***Δ_1***	***$C_{GRASP-CST}$***	***$D_{GRASP-CST}$***	***$T_{GRASP-CST}$ (s)***	***C_{TS-CST}***	***D_{TS-CST}***	***T_{TS-CST} (s)***	***C_{CSTc}***	***D_{CSTc}***	***T_{CSTc} (s)***
B01*	31	**82***	30	0.170	82*	30	0.290	82*	30	0.591
B02*	56	**83***	55	0.199	83*	55	0.310	90	55	1.152
B03*	79	**138***	78	0.329	138*	78	0.410	140	78	1.692
B04*	59	**59***	58	1.041	59*	58	2.664	75	58	1.421
B05	27	**62**	26	1.292	76	26	3.143	63	26	0.561
B06	66	**126**	65	1.762	126	63	2.403	128	65	1.571
B07*	52	**111***	51	0.430	111*	51	0.890	118	51	3.153
B08*	50	**104***	49	0.480	104*	49	0.661	110	39	3.034
B09	52	231	48	0.830	231	48	0.670	225	51	3.134
B10*	67	**86***	66	2.733	86*	66	12.467	106	51	4.606
B11*	66	**88***	65	2.254	92	61	14.020	99	57	4.516
B12*	67	**174***	66	6.057	180	54	11.315	182	66	4.717
B13*	39	**165***	38	1.361	165*	38	2.913	187	38	5.197
B14*	71	**235***	70	1.252	239	64	3.634	238	70	9.874
B15	78	330	61	1.702	330	61	3.444	328	71	10.975
B16*	64	**127***	63	4.737	149	58	33.478	146	54	9.624
B17*	60	**131***	59	8.862	131*	59	33.569	165	50	9.033
B18	81	**219**	80	11.175	219	80	24.404	226	80	12.367

5.1.4 The Set of Test Problems and the Experimental Method

We implemented $GRASP - CST$, along with the $TS - CST$ algorithm ([87]) and Kompella et al.'s centralized CST_C algorithm ([46]), in C++. We tested the above mentioned algorithms on problem set B from Steinlib ([45]) using the experimental method suggested in [87] which will be briefly described. All three algorithms were executed on a PC powered by a Pentium 2 450MHz processor.

Steinlib is a publicly available library of test data for the Minimum Steiner Tree Problem in Graphs (MStTG). Since we consider the Delay-Constrained Multicast Routing (DCMR) problem which reduces to the Constrained MStTG problem, the test data as such is not sufficient. Since the edges in the test data have only a cost function assigned, their respective delay values are generated randomly. Set D is the set of nodes given in the test data that must be spanned by the Steiner tree. The first node in set D is chosen to serve as our source s. The remaining nodes in $D \setminus \{s\}$ are destination nodes S.

The algorithms were then run with a high enough value of the delay bound so as not to act as a constraint. (The delay bound Δ cannot actually be set to ∞ since the time complexity of the CST_C algorithm is $O(\Delta|V|^3))$. These obtained solutions are really the solutions to the (unconstrained) MStTG problem since the delay values of the edges play no role in constructing the Steiner tree. If the cost of the obtained solution is that supplied by the test data, we know that it is optimal. After running each algorithm, the cost of the obtained Steiner tree along with the maximum end-to-end delay from the source to any destination is calculated. The deviation (δ) of the calculated cost above the optimal cost supplied by the test data and the corresponding maximum end-to-end delay (D) are shown in Table 5.1.

The inhibiting factor in the Delay-Constrained Multicast Routing problem is, of course, the value of the delay bound. This means that the smaller the delay bound, the stronger the constraint. For this reason, the following is done: The smallest of the three corresponding maximum delay values found for each test problem while simulating the MStTG problem (Table 5.1), is chosen. This value is then incremented by 1, and set as delay bound Δ_1. The cost (C) and maximum delay values (D) that correspond to the Steiner trees obtained by testing the algorithms with delay bound Δ_1 , along with their execution times (T), are shown in Table 5.2. The algorithms are then tested for two more delay bounds: Δ_2 (Table 5.3) and Δ_3 (Table 5.4). Δ_2 is 10% greater than Δ_1, rounded up to the nearest integer, while Δ_3 is 10% less than Δ_1, rounded down to the nearest integer.

Since the $GRASP - CST$ algorithm gave the optimal solution to all 18 test problems for the MStTG problem, we know the maximum delay that corresponds to all of the optimal solutions. As a result, if the delay bound of the CMStTG problem is set to a value *greater* than the maximum delay of the optimal solution to the MStTG problem, we know that this optimal solution to the MStTG problem is also optimal for the CMStTG problem. Such problems are marked with '*****' in Tables 5.2 and 5.3 to let us know that the optimal solution for these cases is known. For problems where we do not know the optimal solution, we simply compare the performance of the three implemented algorithms. Unfortunately, to the best of our knowledge, there is no test data available for the CMStTG problem.

To determine appropriate values for the input parameters for the $GRASP - CST$ algorithms, a number of experiments were performed. The goal was to use a small number of GRASP iterations and a small number of iterations in the local search phase to reduce execution time, and yet obtain good solutions for this set of problems. Regarding the local search procedure, we first set the number of iterations without improvement to 1 to make it a strictly local neighborhood search. However, the neighborhood of the $TS - CST$ algorithm used in the local search procedure proved too restrictive. One of the reasons for this is the

Table 5.3: Solution quality for $\Delta_2 = 1.1 \cdot \Delta_1$

		GRASP-CST			*TS-CST*			CST_C		
Probl.	Δ_2	$C_{GRASP\text{-}CST}$	$D_{GRASP\text{-}CST}$	$T_{GRASP\text{-}CST}$ *(s)*	$C_{TS\text{-}CST}$	$D_{TS\text{-}CST}$	$T_{TS\text{-}CST}$ *(s)*	C_{CSTc}	D_{CSTc}	T_{CSTc} *(s)*
B01*	35	**82***	30	0.160	82*	30	0.281	82*	30	0.680
B02*	62	**83***	55	0.199	83*	55	0.310	90	55	1.302
B03*	87	**138***	78	0.331	138*	78	0.410	140	78	1.872
B04*	65	**59***	58	1.031	59*	58	2.634	64	58	1.571
B05	30	**62**	26	1.252	76	29	3.114	66	27	0.631
B06	73	**124**	72	1.961	124	72	2.463	128	65	1.752
B07*	58	**111***	51	0.450	111*	51	0.901	118	51	3.554
B08*	55	**104***	49	0.509	104*	49	0.670	110	39	3.374
B09	58	**221**	57	0.850	221	57	0.740	225	51	3.534
B10*	87	**86***	66	3.044	86*	66	12.447	99	63	6.078
B11*	73	**88***	65	2.283	92	61	14.009	93	57	5.128
B12*	74	**174***	66	5.227	174*	66	11.456	175	71	5.127
B13*	43	**165***	38	1.391	165*	38	2.923	187	38	5.768
B14*	79	**235***	70	1.351	238	74	4.004	238	70	11.085
B15*	86	**318***	81	2.674	318*	81	3.854	322	50	12.677
B16*	71	**132**	50	4.987	149	58	33.709	137	64	10.754
B17*	66	**131***	59	8.573	134	59	32.858	148	49	9.914
B18	90	**219**	80	10.867	222	84	25.045	226	80	13.839

neighborhood structure of the $TS - CST$ algorithm. Namely, since potential solutions are represented by a set of Steiner nodes, and the neighborhood is defined as all those solutions where the status of only a single node is changed, the neighborhood of a current solution often consists of all infeasible solutions (i.e. unconnected trees). Allowing more flexibility drastically improved results. To provide this flexibility, we set the number of iterations without improvement to 2 and, thus, allowed 2 nodes to be changed with regards to their status as Steiner nodes before obtaining a solution better than the current one. This little nudge beyond the first local optimum significantly improved results. Of course, raising the value of this parameter even further could potentially lead to even better solutions but testing indicated that the gain on solution quality was not significant with respect to the increase in execution time. Also, for several of the cases tested, the obtained solutions were optimal. Thus, further raising this value seemed unnecessary.

Regarding the remaining parameters, it was necessary to determine a good balance between parameter α and the number of GRASP iterations. Parameter α should be large enough

Table 5.4: Solution quality for $\Delta_3 = 0.9 \cdot \Delta_1$

		GRASP-CST			*TS-CST*			*CST$_C$*		
Probl.	Δ_3	$C_{GRASP\text{-}CST}$	$D_{GRASP\text{-}CST}$	$T_{GRASP\text{-}CST}$ *(s)*	$C_{TS\text{-}CST}$	$D_{TS\text{-}CST}$	$T_{TS\text{-}CST}$ *(s)*	C_{CSTc}	D_{CSTc}	T_{CSTc} *(s)*
B01	27	**-**	-	-	-	-	-	-	-	-
B02	50	**91**	43	0.281	91	43	0.260	91	43	1.062
B03	71	**144**	59	0.400	144	59	0.360	155	70	1.761
B04	53	**62**	35	0.860	64	42	2.554	80	48	1.232
B05	24	**66**	19	1.221	75	21	3.184	66	18	0.480
B06	59	138	58	1.302	127	53	2.604	135	52	1.091
B07	46	**-**	-	-	118	33	0.900	128	32	2.813
B08	45	**107**	36	0.429	107	34	0.720	111	34	2.703
B09	46	**-**	-	-	-	-	-	-	-	-
B10	60	**88**	51	2.464	91	57	12.798	100	51	4.106
B11	59	**89**	43	2.263	94	57	13.629	99	57	4.116
B12	60	**177**	56	6.329	190	58	12.046	200	57	4.217
B13	35	**172**	28	0.971	-	-	-	217	34	4.607
B14	63	**240**	62	1.292	246	55	3.524	243	50	8.953
B15	70	**330**	61	2.293	330	61	3.454	330	63	9.974
B16	57	**129**	51	5.877	153	55	33.919	146	54	8.502
B17	54	**136**	53	7.209	158	53	25.316	159	50	8.272
B18	72	**223**	67	11.816	227	69	25.716	228	70	10.914

to enable a diversified search and yet small enough to intensify the search around good solutions. Recall that candidates in the restricted candidate list (RCL) are chosen according to the cost of adding them to the existing tree, i.e. if the cost of adding a node is less than or equal to the cost of the best candidate multiplied by factor α, the node is included in the RCL. Since the costs on edges in the networks ranged from 1 to 10, the costs of the paths connecting various nodes to the existing tree often varied significantly. As a result, setting α to a value close to 1 proved fairly restrictive, resulting in a construction phase that ran almost like a pure greedy algorithm. This caused most of the GRASP iterations run to give the same solution. For most cases, this solution was good but it could still be improved. Tests showed that setting α to 5 provided enough flexibility in the construction phase to enable the $GRASP - CST$ algorithm to perform a diversified search. Values higher than 5 often obtained poor quality solutions in the construction phase and, thus, a large number of GRASP iterations had to be run to obtain good solutions. When parameter α was set to 5, only 5 iterations of $GRASP - CST$ were required to obtain high quality results for this problem

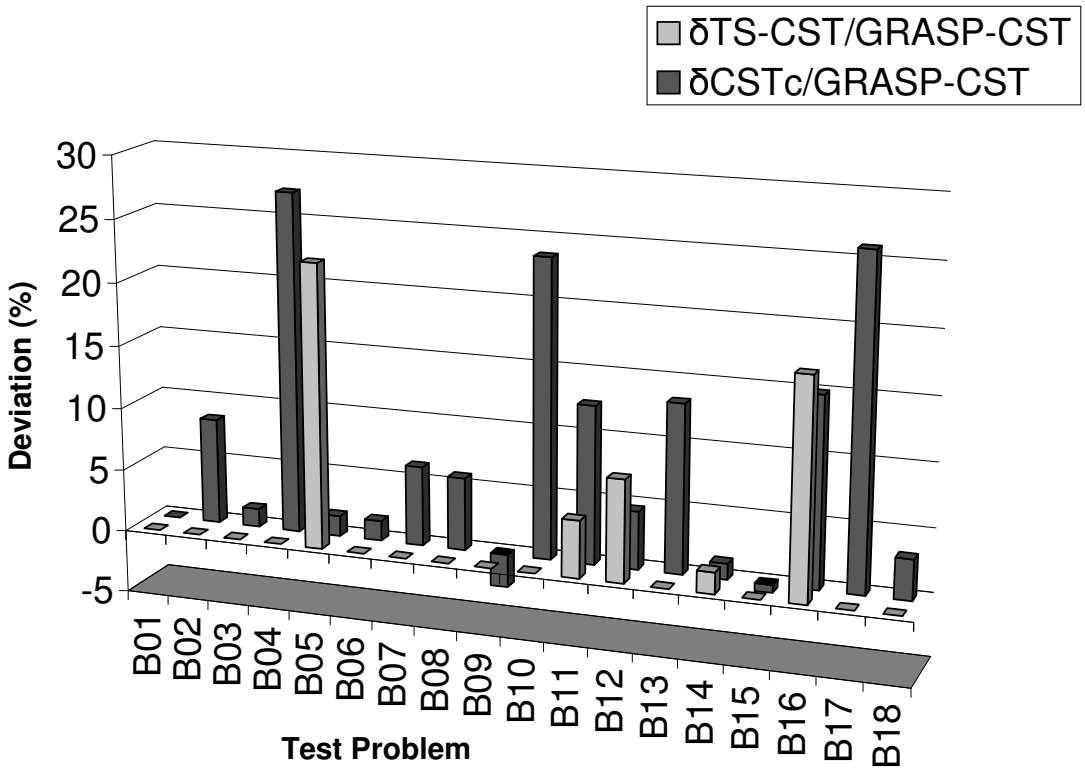


Figure 5.4: Deviation of the cost of the solutions obtained by $TS-CST$ and CST_C over $GRASP-CST$ for Δ_1

set.

It follows that the results shown in Tables 5.1-5.4 are those obtained with the following input values: the number of GRASP iterations is set to 5, α is set to 5, and the number of iterations without improvement of the local search procedure is set to 2. Parameters for testing the $TS-CST$ algorithm are those chosen in [87] where the number of iterations for problems B01-B09 is 25, while the remaining problems are run for 40 iterations.

For easier visualization of the obtained results, the deviation of the cost of the solutions found by the $TS-CST$ and CST_C algorithms above the cost of the corresponding solution obtained by the $GRASP-CST$ algorithm for the middle delay bound (Δ_1) are shown in Fig. 5.4. The average deviation of the cost of the constrained Steiner tree obtained by the $TS-CST$ algorithm over that obtained by the $GRASP-CST$ algorithm ($\delta_{TS-CST/GRASP-CST}$) is +3.01%. In the case of the CST_C algorithm ($\delta_{CST_C/GRASP-CST}$), the average deviation is +8.25%.

5.1.5 Numerical Results

In Table 5.1, we can see that for the unconstrained multicast routing problem (reduced to the MStTG problem), $GRASP - CST$ gave the optimal solution in *all* cases, while the $TS - CST$ and CST_C algorithms found the optimal solution in 13 and 5 cases, respectively. These results indicate that the suggested GRASP heuristic is efficient for the general problem of multicast routing. Regarding QoS multicasting with a bounded end-to-end delay, Tables 5.2, 5.3 and 5.4 show the results of the algorithms for the CMStTG problem with various delay bounds. $GRASP - CST$ performed better than both the $TS - CST$ and CST_C algorithms for all three delay bounds. For Δ_1, $GRASP-CST$ gave better or equal solutions (marked in bold) for 16 out of 18 problems. For Δ_2, this was the case for all 18 problems, while for Δ_3, $GRASP - CST$ performed better or equal to the $TS - CST$ and CST_C algorithms for 16 out of 18 problems.

Regarding optimality, we can see from Table 5.2 that for Δ_1, $GRASP - CST$ obtained the optimal solution (denoted as '*****') in all 13 cases where the optimal solution is known. The $TS - CST$ algorithm did so in 9 cases, while CST_C did so in only 1 case. For the problems for which the optimal solution is not known, we compare the obtained results with lower bounds. Namely, the optimal solutions for the unconstrained minimum Steiner tree problem (shown in Table 5.1) represent lower bounds on the solutions for the constrained problem. For Δ_1, the *maximum* deviation of a solution obtained by the $GRASP - CST$ algorithm over its corresponding lower bound was 5.00%. This occurred for problem B09. The largest deviations of solutions obtained by the $TS - CST$ and CST_C algorithms over the corresponding lower bounds were 24.59% (problem B05) and 27.12% (problem B04), respectively. Note that for problem B04, the optimal solution is known. Therefore, this deviation is the deviation over the optimal solution, and not just the lower bound.

We can see from Table 5.3 that for Δ_2, $GRASP - CST$ found the optimal solution in all but one of the 13 cases where the optimal solution is known. The $TS - CST$ algorithm found the optimal solution in 9 cases, while the CST_C algorithm in 1 case. For Δ_2, the solution obtained by the $GRASP - CST$ algorithm deviated most over the lower bound (in this case, the optimal solution) for problem B16. $GRASP - CST$ gave a solution more expensive by 5 units of cost (3.94%). The $TS - CST$ algorithm deviated most for problem B05, giving a solution more expensive by 15 units of cost (24.59%). The maximum deviation of the CST_C algorithm was for problem B10. The obtained solution was 13 units (15.12%) more expensive than the optimal solution.

For the smallest delay bound, Δ_3, the optimal solutions are not known for any of the cases so we compare with lower bounds from Table 5.1. We can see from the results in

Table 5.4, that for Δ_3 the $GRASP - CST$ algorithm deviated over the lower bound (for cases when a feasible solution was found[1]) most by 13.11% for problem B06. For cases where a feasible solution was found, the $TS - CST$ algorithm deviated most over the lower bound for problem B05, i.e. by 22.95%. The CST_C algorithm did so for problem B04 where it obtained a solution 35.59% more expensive than the lower bound. All these results indicate that the $GRASP - CST$ algorithm is more robust than $TS - CST$ and CST_C, and *consistently* gives high quality solutions.

Comparing the execution times of the algorithms is difficult since both $GRASP - CST$ and $TS - CST$ can be terminated at any time depending on the desired number of iterations. CST_C on the other hand ends deterministically. Even so, for the chosen number of iterations, $GRASP - CST$ performed better than, or equal, to $TS - CST$ for all but one case with respect to solution quality *and* all but two cases where both algorithms found a feasible solution with respect to execution time. Recall that the local search phase of $GRASP-CST$ uses the $TS - CST$ algorithm. Comparison of the execution times of the algorithms tested evidently shows that fewer iterations of $TS - CST$ are run in the local search phase of $GRASP - CST$ than in the $TS - CST$ algorithm itself as run in [87], and yet $GRASP - CST$ obtains better solutions. This shows that the construction phase of $GRASP - CST$ often gives good solutions and that the local search phase converges quickly. This is one of the main advantages of the GRASP metaheuristic.

The execution time of $GRASP - CST$ did not exceed 12 seconds for even the largest problems, while the execution time of $TS - CST$ ran up to 33.919 seconds and yet produced a solution of inferior quality. For the chosen number of iterations, $GRASP - CST$ also performed better than the CST_C algorithm in solution quality as well as in execution time. For each delay bound, $GRASP - CST$ was faster for all but two problems where both algorithms found a feasible solution. The average execution time of the $GRASP - CST$ algorithm run for the above specified number of iterations over all the tested problems for all three delay bounds was 2.722 seconds. The average execution time for the $TS - CST$ algorithm was 8.696 seconds, while the CST_C algorithm on average ran for 5.012 seconds. We can see that the $GRASP - CST$ algorithm gives superior solutions in less time than both $TS - CST$ and CST_C for this set of problems.

[1]Note that there are cases for all three algorithms where no feasible solution was found. Thus, the deviation over the lower bound for these cases is infinite.

5.1.6 Summary and Future Work

In this section, we proposed a GRASP heuristic algorithm for solving the Delay-Constrained Multicast Routing problem. In the past couple of years there has been an increased development of numerous multimedia network applications, many of which transfer information in real-time interactive environments to a group of users. Many of these applications can tolerate only a bounded end-to-end delay and therefore require delay-constrained multicast routing algorithms.

In the proposed algorithm, the Delay-Constrained Multicast Routing problem is first reduced to the Constrained Minimum Steiner Tree problem and then the GRASP method is applied. Testing on small and medium sized problems available in SteinLib has shown that the proposed algorithm gives near-optimal solutions in moderate time for this set of problems. The results were also compared to those obtained by a tabu-search algorithm ([87]) and Kompella et al.'s centralized algorithm ([46]) for the same problem. The proposed GRASP heuristic algorithm outperforms both of the above mentioned algorithms for this problem set.

GRASP (Greedy Randomized Adaptive Search Procedure) is a metaheuristic proven to be efficient for a wide array of optimization problems. This search procedure seems little used in research dealing with QoS-driven multicast routing. The encouraging results obtained in this thesis indicate that further research in this field could be useful. Introducing multiple QoS demands to the multicast routing problem such as the minimum bandwidth or the maximum delay jitter could be interesting for further avenues of research. The adaptation of GRASP strategies to the problem of dynamic multicast routing, or rather *re-routing* when multicast members join or leave the group during the lifetime of the connection, could also prove interesting.

5.2 Static Multicast Routing and Wavelength Assignment

To establish a virtual topology composed of a set of *light-trees* (as opposed to lightpaths) in wavelength routed networks, we must solve the *Multicast Routing and Wavelength Assignment* (MC_RWA) problem. In this section, we consider *static* multicast requests, i.e. all the the requests are known *a priori* and the virtual topology is established 'semi-permanently'. Given is a network and a set of multicast requests. For each multicast request, it is necessary to find a multicast tree, i.e. a light-tree, which connects the source node to all the destination nodes.

In this section develop heuristic algorithms for the routing and wavelength assignment of multicast requests by efficiently applying bin packing based algorithms. These heuristics are

motivated by the concepts used by RWA algorithms for unicast (lightpath) demands ([86]) described in Chapter 3. They also apply the GRASP heuristic for the delay constrained multicast routing problem ([89]) described in Section 5.1. The objective of the proposed heuristic algorithms for the MC_RWA problem is to minimize the number of wavelengths used. We also consider a second objective, which is to minimize the costs of the established light-trees. The cost of a light-tree can represent various values such as the actual cost, the total number of hops, the total length or the maximum transmission delay in the tree. Delay constrained multicasting, where each multicast request has an end-to-end delay bound associated with it, is also considered.

The algorithms were tested on random networks and on a benchmark problem set for the Steiner tree problem from [45]. Comparison with lower bounds indicates that the proposed algorithms obtain solutions of good quality both with respect to the number of wavelengths used and average light-tree cost, particularly for denser networks. These algorithms are highly flexible and can consider unicast, multicast and broadcast requests with or without QoS constraints.

The rest of the section is organized as follows. In Section 5.2.1 we informally define the MC_RWA problem and discuss related work in Section 5.2.2. In Section 5.2.3 we suggest heuristic algorithms for the MC_RWA problem based on bin packing algorithms. Lower bounds are briefly discussed in Section 5.2.4. Numerical results and a summary are given in Sections 5.2.5 and 5.2.6, respectively.

5.2.1 Problem Definition

The physical optical network is modelled as a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. Edges are assumed to be bidirectional (each representing a pair of optical fibers, i.e. one fiber per direction). On graph G we define the functions $c(i, j)$ and $d(i, j)$, where $c(i, j)$ can represent the cost of using edge $(i, j) \in E$ and $d(i, j)$ can represent the length or propagation delay along edge $(i, j) \in E$. The cost of an edge is not necessarily proportional to its delay. Given is a set of multicast requests $\tau = \{(s_1, S_1, \Delta_1), \ldots, (s_n, S_n, \Delta_n)\}$, where $\{s_i \cup S_i\} \subseteq V, i = 1, \ldots, n$. Each multicast request is defined by a source node $s_i \in V$, a group of destination nodes $S_i \subseteq V$, and an upper bound on the delay from s_i to any node in S_i denoted as Δ_i. If we are considering multicasting with no QoS demands, Δ_i is set to ∞. If we are considering multicasting with a bounded end-to-end delay, Δ_i is set to the desired bound.

The Multicast Routing and Wavelength Assignment problem consists of finding a set of trees $T = \{T_1, \ldots, T_n\}$ in G, each corresponding to one multicast request, and assigning

wavelengths to them. We assume that the trees are bidirectional, i.e. that data is transmitted between the source and destination nodes in both directions. Each tree $T_i = (V_{T_i}, E_{T_i})$, where $V_{T_i} \subseteq V$ and $E_{T_i} \subseteq E$, is subject to the following constraints. $s_i \cup S_i \subseteq V_{T_i}$ and $D(s_i, v) \leq \Delta_i$ for every $v \in S_i$ where $D(s_i, v) = \sum_{(j,k)} d(j,k)$ for all edges $(j,k) \in E_{T_i}$ on the path from s_i to v in T_i. The cost of tree T_i is $c(T_i) = \sum_{(j,k)\in E_{T_i}} c(j,k)$. Trees T_i and T_j where $i \neq j$, $i, j = 1, \ldots, n$, cannot be assigned the same wavelength if they share a common edge. We assume no bound on the degree of a multicast tree, i.e. the optical signal can be split into an arbitrary number of signals. The objective is to minimize the number of wavelengths required to successfully route and assign wavelengths to all the multicast requests in τ. We also consider a second objective which is to minimize the average cost of the established trees, i.e. $min \frac{\sum_{i=1}^{n} c(T_i)}{|\tau|}$.

5.2.2 Related Work

Previous works regarding the MC_RWA problem consider various problem models and solution approaches. In [8], the authors decompose the MC_RWA problem into two subproblems, routing and wavelength assignment, solved subsequently. For multicast routing, a heuristic is suggested which minimizes the cost of the multicast trees. The authors consider cost to include not only the bandwidth cost, but the cost of wavelength conversion and light splitting as well. Furthermore, the authors prove that wavelength assignment for a given routing scheme is not NP-hard and propose a polynomial optimal wavelength assignment algorithm. In [33], wavelength assignment for dynamic multicasting, i.e. where multicast sessions are dynamically set up and released over time, was also shown to be solvable in linear time if the number of wavelengths per link, transmitters and receivers per node, and switch degree are constants.

In [31] and [34], the authors explore multicast routing under the *multi-tree* model. In such a model, one multicast request is realized with a collection of light-trees where each light-tree can have at most a specified number of destinations. A 4-approximation routing algorithm is proposed which minimizes the cost of the established trees. A wavelength assignment algorithm is also suggested.

QoS multicast routing and wavelength assignment is studied in [40]. The QoS demand considered is a bounded end-to-end delay from the source node to any destination node in a multicast session. Heuristic algorithms with the objective to minimize the number of wavelengths using two different approaches are proposed. The first approach reduces the maximal link load in the system, while the second tries to free the least used wavelength.

Multicasting in all-optical networks where each node can receive only one signal at a

time, referred to as the *single reception constraint*, is studied in [72]. Using some properties of expander graphs, the authors obtain an upper bound on the number of wavelengths required to support such multicasting. Protective MC_RWA, where back-up trees are reserved to protect multicast sessions, is studied in [84]. The authors give a mathematical formulation for this problem, along with an expanded formulation for protective MC_RWA in sparse splitting networks.

Since multicasting in WDM networks requires multicast-capable switches, their cost and design have been widely studied [64] [47]. Multicasting in optical networks where some switches in the network are incapable of splitting light due to evolutional and/or economic reasons is studied in [103]. The authors propose heuristic algorithms for multicast routing in such networks by constructing a so-called 'light-forest' consisting of a collection of multicast trees for each multicast session. A low cost architecture, referred to as *Tap-and-Continue*, for multicasting in WDM networks along with a 4-approximation algorithm for multicast routing was proposed in [1].

5.2.3 Heuristic Algorithms for the MC_RWA Problem

In order to solve the MC_RWA problem we propose fast and simple heuristic algorithms developed by applying concepts used for bin packing. Recall that the bin packing problem is a classical combinatorial NP-complete optimization problem already discussed in Section 3.3.1. Here, we briefly summarize the concepts relevant to understanding the algorithms proposed in this section. Given is a list of n items of various sizes and identical bins of limited capacity. To solve the bin packing problem, it is necessary to pack these items into the minimum number of bins, without violating the capacity constraints, so that all items are packed. Since this problem is NP-hard [27], a vast array of approximation algorithms have been proposed and studied. Four well-known classical bin packing algorithms are the First Fit (FF), Best Fit (BF), First Fit Decreasing (FFD) and Best Fit Decreasing (BFD) algorithms. The FF algorithm packs each item, in the order in which they are given, into the first bin into which it fits. The BF algorithm packs each item into the bin which leaves the least room left over after packing the item. The FFD and BFD algorithms sort the given items in non-increasing order of their corresponding sizes, and then perform packing in the same manner as the FF and BF algorithms, respectively. These algorithms perform significantly better than FF and BF. Surveys of bin packing algorithms can be found in [19] and [18].

We apply these classical bin packing methods to help solve the Multicast Routing and Wavelength Assignment problem. Since each link in the physical network G can support multiple wavelengths, we can consider G to be a multilayered graph where each layer rep-

resents one wavelength. Our main objective is to 'pack' a set of multicast requests into this graph using the least number of layers, i.e. using the minimum number of wavelengths. As a result, we consider multicast requests to represent the 'items' in bin packing, while copies of graph G (individual layers) represent 'bins'. Each copy of G, referred to as bin $G_i, i = 1, 2, 3, \ldots$, corresponds to one wavelength. The capacity of each bin is limited by the edges in G since light-trees routed on the same layer cannot traverse any of the same edges due to the *wavelength clash constraint*.

Since the FFD and BFD bin packing algorithms sort 'items' in decreasing order of their corresponding sizes, we must define the size of a multicast request. Herein, we suggest two evaluation functions.

- $|S_i|$: The first evaluation function considers the size of a multicast request to be the number of destination nodes, i.e. the cardinality of set S_i. This size is easy to calculate but may not be a good representative of the actual size of the multicast tree. Namely, if all the destination nodes are set close to each other in the network, the multicast tree may be much smaller than a tree whose destination nodes are spread out over the graph even though they are fewer in number. Also, this measure may not be relevant if all the multicast sessions have a similar number of destination nodes.

- MCT_j: The second evaluation function considered for the size of a multicast request is an approximation of the corresponding minimum cost multicast tree. As already mentioned, finding a multicast tree, i.e. multicast routing, reduces to the minimum Steiner tree problem in graphs. Since this problem itself is NP-hard [27], there is no polynomial time algorithm known which can guarantee the optimal minimum cost Steiner tree. Therefore, we consider the size of each multicast request $(s_j, S_j, \Delta_j) \in \tau$ to be the length of the suboptimal minimum cost tree, MCT_j, in graph G found using the multicast routing heuristic algorithm, called GRASP-CST ([89]) described in Section 5.1.3. However, it is important to note that multicast requests will not necessarily be routed on these found suboptimal trees. This measure is used only by the algorithms in order to sort the 'items' or multicast requests in non-increasing order of their corresponding sizes.

A description of the proposed heuristics for the MC_RWA problem follows. Their corresponding pseusocodes are shown in Fig. 5.5. Algorithms referred to as FF_MC_RWA and BF_MC_RWA are based on the classical bin packing algorithms FF and BF, respectively. Two algorithms, FFD_MC_RWA and FFTD_MC_RWA, are suggested which correspond to the bin packing FFD algorithm. The two differ with respect to the evaluation of the size of

FF_DCMC_RWA (FFD_DCMC_RWA; FFCD_DCMC_RWA)
Input:
$G = (V, E)$;*//physical network*
$\tau = \{(s_1, S_1, \Delta_1), \ldots, (s_n, S_n, \Delta_n)\}$; *//multicast requests*
Begin:

ONLY FOR **FFD_DCMC_RWA**:
Sort and renumerate demands τ in nonincreasing order of the number of destination nodes in each request, $|S_i|, i = 1, \ldots, n$

ONLY FOR **FFCD_DCMC_RWA**:
for $i = 1$ to n **do**
Run $GRASP\text{-}CST(s_i, S_i, \Delta_i; G)$ to obtain Steiner tree MCT_i;
end for
Sort and renumerate demands τ in nonincreasing order of the cost of the obtained Steiner trees $c(MCT_i), i = 1, \ldots, n$.

$T = \{\}$; *//The final trees*
Create 1 copy (bin) of $G : G_1$;
$BINS := \{G_1\}$;
while τ is not empty **do**
for $j = 1$ to $|\tau|$ **do**
$T_j = \emptyset$;
for $i = 1$ to $|BINS|$ **do**
Find Steiner tree T_j^i by running
$GRASP\text{-}CST(s_j, S_j, \Delta_j; G_i)$;
if feasible **then**
$T_j = T_j^i$;
Assign wavelength i to tree T_j;
Delete edges in T_j^i from G_i;
$i = |BINS|$;
end if;
end for;
if $T_j = \emptyset$ **then**
$New := |BINS| + 1$;
Create copy of G: G_{New};
$BINS := BINS \cup \{G_{New}\}$;
Find Steiner tree, T_j^{New}, by running
$GRASP\text{-}CST(s_j, S_j, \Delta_j; G_{New})$;
$T_j = T_j^{New}$;
Assign wavelength New to path T_j;
Delete edges in T_j^{New} from G_{New}
end if;
$T = T \cup T_j$;
$\tau = \tau \setminus (s_j, S_j, \Delta_j)$;
end for;
end while;
return T;
End

BF_DCMC_RWA (BFD_DCMC_RWA; BFCD_DCMC_RWA)
Input:
$G = (V, E)$;*//physical network*
$\tau = \{(s_1, S_1, \Delta_1), \ldots, (s_n, S_n, \Delta_n)\}$; *//multicast requests*
Begin:

ONLY FOR **BFD_DCMC_RWA**:
Sort and renumerate demands τ in nonincreasing order of the number of destination nodes in each request, $|S_i|, i = 1, \ldots, n)$

ONLY FOR **BFCD_DCMC_RWA**:
for $i = 1$ to n **do**
Run $GRASP\text{-}CST(s_i, S_i, \Delta_i; G)$ to obtain Steiner tree MCT_i;
end for
Sort and renumerate demands τ in nonincreasing order of the cost of the obtained Steiner trees $c(MCT_i), i = 1, \ldots, n$.

$T = \{\}$; *//The final trees*
Create 1 copy (bin) of $G : G_1$;
$BINS := \{G_1\}$;
while τ is not empty **do**
for $j = 1$ to $|\tau|$ **do**
$T_j = \emptyset$, $c(T_j) = \infty$;
$BestBin := 0$;
for $i = 1$ to $|BINS|$ **do**
Find Steiner tree T_j^i by running
$GRASP\text{-}CST(s_j, S_j, \Delta_j; G_i)$;
if feasible and $c(T_j^i) < c(T_j)$ **then**
$BestBin = i$;
$T_j = T_j^i$;
Assign wavelength i to tree T_j;
end if;
end for;
if $T_j \neq \emptyset$ **then**
Delete edges in $T_j^{BestBin}$ from $G_{BestBin}$;
else
$New := |BINS| + 1$;
Create copy of G: G_{New};
$BINS := BINS \cup \{G_{New}\}$;
Find Steiner tree, T_j^{New}, by running
$GRASP\text{-}CST(s_j, S_j, \Delta_j; G_{New})$;
$T_j = T_j^{New}$;
Assign wavelength New to tree T_j;
Delete edges in T_j^{New} from G_{New}
end if;
$T = T \cup T_j$;
$\tau = \tau \setminus (s_j, S_j, \Delta_j)$;
end for;
end while;
return T;
End

Figure 5.5: Pseudocodes of the FF_MC_RWA, BF_MC_RWA, FFD_MC_RWA, BFD_MC_RWA, FFTD_MC_RWA, and BFTD_MC_RWA algorithms.

a multicast request. Analogously, algorithms BFD_MC_RWA and BFTD_MC_RWA correspond to bin packing algorithm BFD.

FF_MC_RWA**:** The First Fit Multicast Routing and Wavelength Assignment algorithm, referred to as FF_MC_RWA, runs as follows. Layers of G, or bins, are created as needed and sequentially indexed. The algorithm begins by creating one layer of G, called G_1. Multicast requests (s_j, S_j, Δ_j) are selected at random and routed on the lowest indexed layer of G in which there is room. Bin G_i is considered to have room for multicast request (s_j, S_j, Δ_j) if we can find a multicast tree, using the GRASP-CST algorithm, connecting s_j to all the nodes in S_j in G_i. This tree is denoted as T_j^i. If we are considering delay constrained multicasting, this tree must satisfy the

delay constraint. If a multicast request is routed in bin G_i, the request is assigned wavelength i and the edges along tree T_j^i are deleted from G_i. If all the edges from bin G_i are deleted, the bin no longer needs to be considered. If no existing bin can accommodate multicast request (s_j, S_j, Δ_j), a new bin is created.

BF_MC_RWA**:** The Best Fit Multicast Routing and Wavelength Assignment algorithm, BF_MC_RWA, runs as follows. Multicast requests are routed on the layer of G in which they fit 'best'. We consider the best fit to be the layer on which we can find the least cost feasible multicast tree. In other words, if at some point in running the algorithm, there are B bins created, bin G_i, $1 \leq i \leq B$, is considered to be the best bin for multicast request (s_j, S_j, Δ_j) if $c(T_j^i) \leq c(T_j^k)$, for all $k = 1, \ldots, B$. This is not necessarily the suboptimal minimum cost tree, MCT_j, found on the original graph G, since it is possible that none of the existing bins have this tree available. If there is no feasible tree available in any of the B bins, a new bin is created.

The benefit of such a 'best fit' approach is that it attempts to minimize the cost of the established multicast trees which is the second objective we consider for the MC_RWA problem. Of course, we could route each multicast request (s_j, S_j, Δ_j) strictly on its suboptimal minimum cost tree, MCT_j, but this would in most cases lead to using a larger number of layers, which in turn means using a larger number of wavelengths.

FFD_MC_RWA**:** The First Fit Decreasing Multicast Routing and Wavelength Assignment algorithm sorts the multicast requests in non-increasing order of their corresponding number of destination nodes, i.e. $|S_j|$. Requests with an equal number of destination nodes are placed in random order. The rest of the algorithm runs as FF_MC_RWA. Sorting the requests in this order may establish multicast trees using less wavelengths. The reasoning behind this is that routing requests with a large number of destination nodes is in most cases more demanding than routing multicast requests with fewer destination nodes. Therefore, if we route the requests which are more demanding first, i.e. when there are more edges available on low indexed layers of G, routing on these layers will most likely be successful. We then may be able to fill up the remaining space on these already used layers with less demanding requests and thus eliminate the need for creating higher indexed layers. This may lead to a routing and wavelength assignment using less wavelengths.

BFD_MC_RWA**:** The Best Fit Decreasing Multicast Routing and Wavelength Assignment algorithm sorts the multicast requests in non-increasing order of their corresponding number of destination nodes, i.e $|S_j|$, and then runs as BF_MC_RWA.

$FFTD\_MC\_RWA$**:** The First Fit Tree Decreasing Multicast Routing and Wavelength Assignment algorithm sorts the multicast requests in non-increasing order of the cost of the suboptimal multicast trees in G, MCT_j, found for each request using the GRASP-CST algorithm from [89]. The algorithm then proceeds as FF_MC_RWA. This method of sorting requests is more complex than that in FFD_MC_RWA but seems a better indicator of multicast request size and thus may help obtain better solutions for some instances.

$BFTD\_MC\_RWA$ The Best Fit Tree Decreasing Multicast Routing and Wavelength Assignment algorithm sorts the multicast requests in non-increasing order of the cost of the suboptimal multicast trees in G, MCT_j, and then runs as BF_MC_RWA.

5.2.4 Lower Bounds

To assess the quality of the solutions obtained by the proposed algorithms, we suggest lower bounds for the number of wavelengths used and the average cost of the established light-trees. A lower bound on the number of wavelengths needed to establish a given set τ of multicast requests in network $G = (V, E)$ is

$$LB_W = \max_{i \in V} \lceil \frac{\Delta_l(i)}{\Delta_p(i)} \rceil. \tag{5.1}$$

This is similar to the lower bound on the number of wavelengths for the routing and wavelength assignment problem for unicast demands used in [86]. $\Delta_l(i)$ represents the logical degree of node i, while $\Delta_p(i)$ represents the node's physical degree. The logical degree of a node is the number of multicast requests for which the node is the source or destination node. Recall that the established multicast trees are bidirectional so trees which terminate and originate from the same node cannot be assigned the same wavelength if they traverse the same edge adjacent to the node. This is due to the *wavelength clash constraint*. Since node i has $\Delta_p(i)$ adjacent physical links and is the source or destination node for $\Delta_l(i)$ multicast trees, at least one physical link will have $\lceil \frac{\Delta_l(i)}{\Delta_p(i)} \rceil$ multicast trees routed over it. Since trees routed on the same physical links cannot be assigned the same wavelength, at least $\lceil \frac{\Delta_l(i)}{\Delta_p(i)} \rceil$ wavelengths are needed to route the corresponding multicast requests. The highest such ratio among all the nodes in the network is a lower bound on the number of wavelengths needed to solve the MC_RWA problem.

A lower bound on the average cost of the established multicast trees can be found in the following manner. Since finding the minimum cost multicast tree is itself NP-hard, we need to find a lower bound for the minimum cost multicast tree for each request. A simple lower bound for a multicast request with one source node and $|S_i|$ destination nodes is the sum of the $|S_i|$ cheapest edges in G. It follows that a lower bound on the average cost of the multicast trees corresponding to requests in $\tau = \{(s_1, S_1, \Delta_1), \ldots, (s_n, S_n, \Delta_n)\}$, is the average of the lower bounds corresponding to each of the n requests. We refer to this lower bound as LB_C. If we sort edges (i, j) in $|E|$ in increasing order of their costs, $c(i, j)$, and rename them as $\{e_1, \ldots, e_{|E|}\}$, the bound is as follows.

$$LB_C = \frac{\sum_{i=1}^{n}(\sum_{j=1}^{|S_i|} c(e_j))}{n}. \tag{5.2}$$

5.2.5 Numerical Results

The FF_MC_RWA, BF_MC_RWA, FFD_MC_RWA, BFD_MC_RWA, FFTD_MC_RWA, and BFTD_MC_RWA algorithms were implemented in C++ and run on a PC powered by a P4 2.8GHz processor. We generated a series of random 50-node networks with average degrees of 3, 4, 5, and 6 (5 networks per average degree). Next, we generated random sets of multicast requests, consisting of 50, 100, 150, 200 and 250 requests, for each test network. Each request was generated with a random number of destination nodes ranging from 1 to 49. This way, unicast and broadcast traffic was also included since they are special cases of multicast traffic. Functions $c(i, j)$ and $d(i, j)$ were both set to 1 if there was an edge between nodes i and j, and 0 otherwise. In other words, instead of the actual cost, the algorithms try to minimize the number of hops in a multicast tree. The upper bound on the end-to-end delay from the source node to any destination in a multicast session was set to $\max(diam(G), \sqrt{|E|})$ as used for unicast RWA algorithms in [57] and [86]. The input parameters chosen for the delay constrained multicast routing algorithm GRASP-CST are those used in Section 5.1.4.

The average number of wavelengths needed to successfully perform Multicast Routing and Wavelength Assignment by each of the algorithms for the 50-node test networks are shown in Table 5.5. The lower bound, LB_W, is also shown. The best obtained solution for each test case is marked in bold. The FFD_MC_RWA algorithm performed best in 7 cases, the BFD_MC_RWA algorithm in 2 cases, the FFTD_MC_RWA algorithm in 13 cases and the BFTD_MC_RWA algorithm in 6 cases. The FF_MC_RWA and BF_MC_RWA algorithms did not obtain the best solution in any of the cases. For easier visualization of the obtained

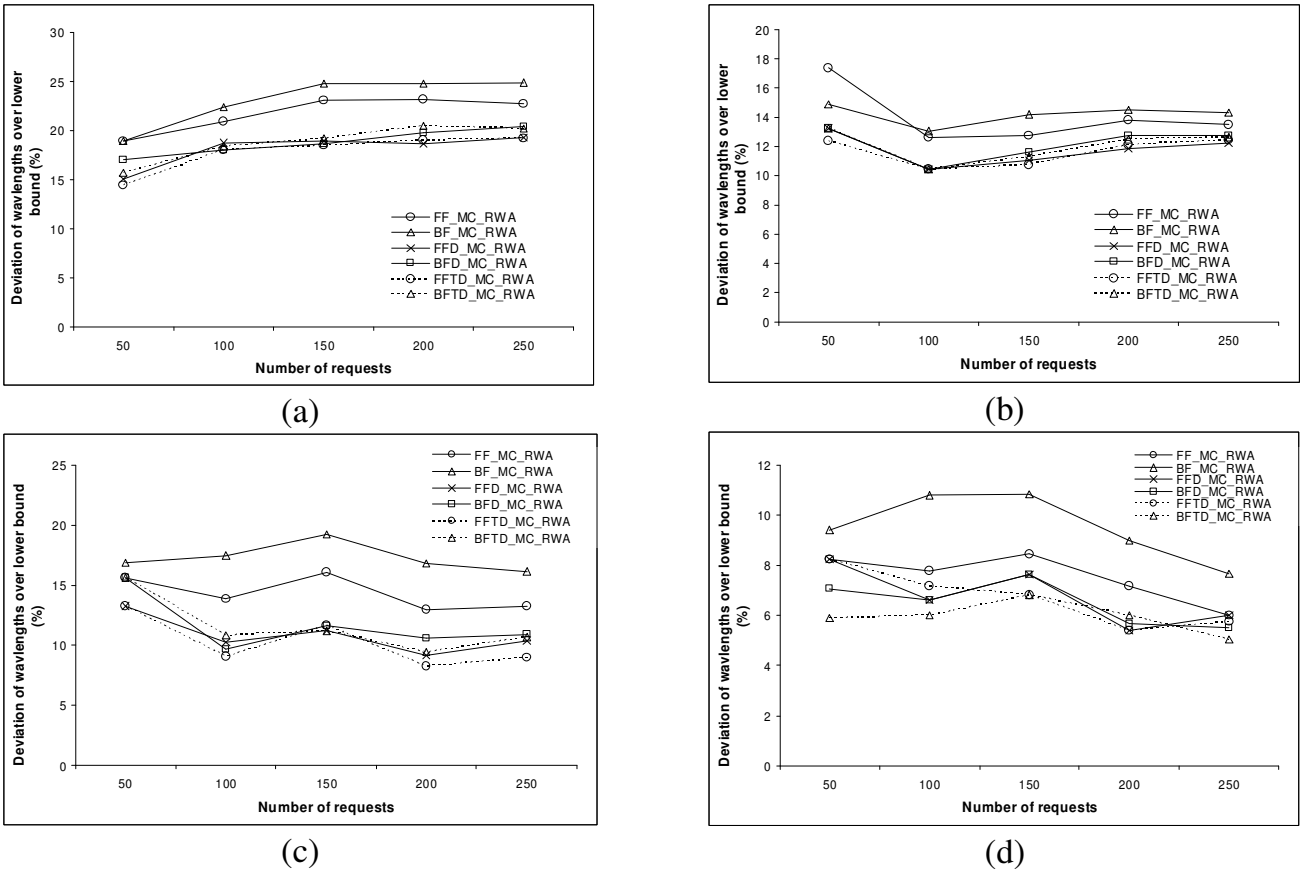


Figure 5.6: The deviation of the number of wavelengths required by the FF_MC_RWA, BF_MC_RWA, FFD_MC_RWA, BFD_MC_RWA, FFTD_MC_RWA, and BFTD_MC_RWA algorithms over the lower bound, LB_W, for random networks with 50 nodes with average degrees (a) 3, (b) 4, (c) 5, and (d) 6.

results, the deviation of the number of wavelengths required by the solutions obtained by each of the algorithms over the lower bound are shown in Fig. 5.6 for networks with an average degree of (a) 3, (b) 4, (c) 5, and (d) 6.

We can see from the results that the gap between the obtained solutions and the lower bound decreases as the density of the network increases. This may be due to the fact that RWA can be solved using less wavelengths in denser networks since more links are available (i.e. the wavelength clash and continuity constraints are easier to meet). As a result, the optimal solution is closer to the ratio of the logical to physical degree in the network and, thus, the lower bound as it is defined in Section 5.2.4 may be closer to the optimal solution. Furthermore, we can see that sorting multicast requests in decreasing order of either the number of destination nodes or the suboptimal multicast trees leads to better solutions, particularly

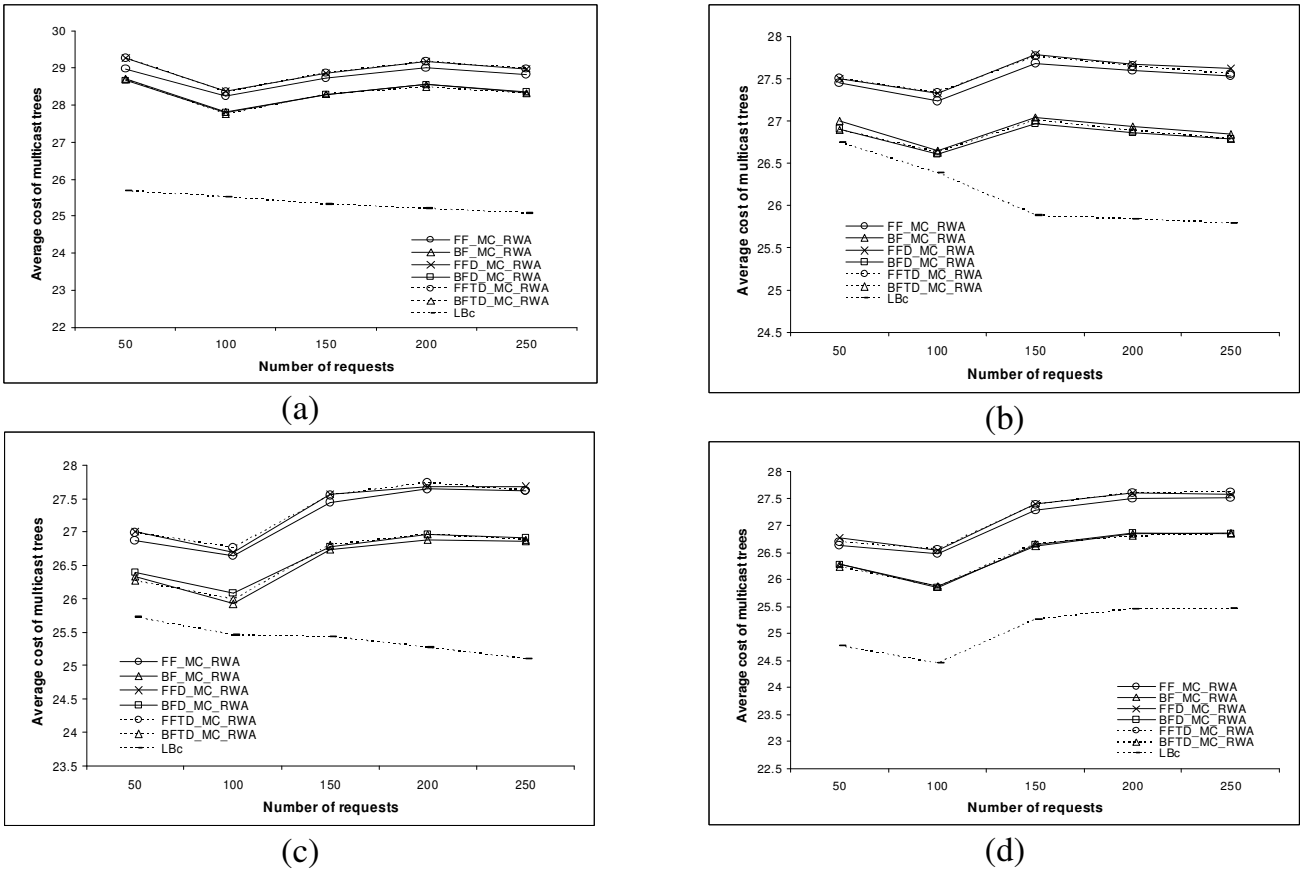


Figure 5.7: The average cost of the multicast trees established by the FF_MC_RWA, BF_MC_RWA, FFD_MC_RWA, BFD_MC_RWA, FFTD_MC_RWA, and BFTD_MC_RWA algorithms and the lower bound, LB_C, for random networks with 50 nodes with average degrees (a) 3, (b) 4, (c) 5, and (d) 6.

in dense networks. Since more resources are available in denser networks, more requests can be packed into a single 'bin' (i.e. copy of graph G) and thus the advantage of sorting the requests becomes more evident.

Routing demands according to the 'best fit' strategy leads to solutions inferior to those obtained using the 'first fit' strategy with respect to the number of wavelengths for the cases tested. However, these algorithms obtain solutions which consistently establish lower cost multicast trees. The average cost of the multicast trees established by the FF_MC_RWA, BF_MC_RWA, FFD_MC_RWA, BFD_MC_RWA, FFTD_MC_RWA, and BFTD_MC_RWA algorithms and the lower bound, LB_C, are shown in Fig. 5.7 for the 50-node test networks with an average degree of (a) 3, (b) 4, (c) 5, and (d) 6. Here we can see the gain of using the 'best fit' strategy.

Furthermore, we tested the algorithms on the set of 18 benchmark network topologies from problem set B from SteinLib [45]. Recall that Steinlib is a publicly available library of test data for Steiner tree problems. The characteristics of the networks are shown in Table 5.7. The costs, $c(i,j)$, of the edges in each network topology are those given in [45]. The delay of an edge, $d(i,j)$, was set to the same value as the cost. To limit the delay from the source to each destination node we set the delay bound to a value proportional to the maximum delay of the shortest delay path in G between the source and any destination node included in the multicast request. In other words, for multicast request (s_i, S_i, Δ_i), $\Delta_i \equiv \beta \cdot max\{SD(s_i,v)|v \in S_i\}$ where $SD(s_i,v) = \sum_{(j,k)} d(j,k)$ for all edges (j,k) on the shortest delay path between nodes s_i and v in G. Such a delay bound was suggested in [40]. β was set here to 2. For each network, we generated 5 random sets of 30 multicast requests with the number of destination nodes ranging from 1 to 29.

The average number of wavelengths and the average cost of the multicast trees obtained by each of the algorithms and the lower bounds are shown in Table 5.6. Here, FFD_MC_RWA seemed to perform best. The gain of sorting multicast requests is not as prominent since these networks are fairly sparse. Still we can see that at least one of the 'decreasing' algorithms obtained the best solution in all of the cases tested. The 'best fit' strategy here again consistently obtained lower cost multicast trees.

We can see from the obtained results that sorting multicast requests in non-increasing order, with respect to either of the evaluation functions presented in this thesis often helps to obtain solutions using fewer wavelengths. Sorting with respect to the number of destination nodes is simpler and yet seems a good measure of size for the cases tested. This seems logical since the number of nodes in the multicast sessions varied significantly. If multicast groups are primarily composed of a similar number of destination nodes, and more so if they are spread out across the network, sorting according to the cost of their corresponding suboptimal trees may perform better. With respect to the method of routing the requests, the algorithms that route light-trees using the 'best fit' strategy, as we define it in this paper, helps to consistently reduce the cost of the multicast trees. The 'first fit' strategy, however, in more cases obtains solutions using fewer wavelengths. As a result, if wavelength are very scarce, it is probably better to use FFD_MC_RWA or FFTD_MC_RWA. If the cost metric is critical, BFD_MC_RWA or BFTD_MC_RWA should be used. Since all these algorithms are one-pass greedy algorithms, running both methods of sorting and choosing the better solution seems reasonable.

Another point which should be mentioned is that in addition to efficiently solving the *static* MC_RWA problem, the FF_MC_RWA and BF_MC_RWA algorithms can be used for

dynamic MC_RWA. Namely, for the *dynamic* MC_RWA problem, multicast requests arrive dynamically and must therefore be established in a specific order. To solve the dynamic MC_RWA problem using the FF_MC_RWA and BF_MC_RWA algorithms, multicast requests in τ are simply established in the specific order in which they arrive according to the corresponding 'first fit' or 'best fit' strategies.

5.2.6 Summary and Future Work

In this section, heuristic algorithms are proposed for the Multicast Routing and Wavelength Assignment problem in wavelength-routed optical networks. These algorithms are extended to solve delay-constrained multicasting as well. All the suggested heuristics are greedy algorithms based on classical bin packing algorithms. Proposed are methods for sorting multicast requests according to two different evaluation functions in order to minimize the number of wavelengths used. A method of routing multicast trees on specific wavelengths, referred to as the 'best fit' strategy, is suggested to minimize the cost of the established trees. The algorithms were tested on random networks and a set of test networks from [45] and the results were compared with analytical lower bounds. Both methods of sorting multicast requests proved efficient with respect to the number of wavelengths used, while using the 'best fit' strategy consistently lowered the cost of the multicast trees. The encouraging results indicate that further research in this field is worthwhile. Further work will include developing heuristics for Multicast Routing and Wavelength Assignment with multiple QoS demands. Networks with limited splitting capabilities will also be studied.

Table 5.5: The average number of wavelengths required by the FF_MC_RWA, BF_MC_RWA, FFD_MC_RWA, BFD_MC_RWA, FFTD_MC_RWA, and BFTD_MC_RWA algorithms and the lower bound, LB_W, for random networks with 50 nodes.

Avg. Degree	*No. Of Requests*	LB_W	*FF_ MC_ RWA*	*BF_ MC_ RWA*	*FFD_ MC_ RWA*	*BFD_ MC_ RWA*	*FFTD_ MC_ RWA*	*BFTD_ MC_ RWA*
3	50	30.6	36.4	36.4	35.2	35.8	**35.0**	35.4
	100	55.6	67.2	68.0	66.0	**65.6**	**65.6**	65.8
	150	82.4	101.4	102.8	98.0	97.8	**97.6**	98.2
	200	109.6	135.0	136.8	**130.0**	131.2	130.4	132.0
	250	135.8	166.6	169.6	162	163.4	**161.8**	163.2
4	50	24.2	28.4	28.7	27.4	27.4	**27.2**	27.4
	100	46.0	51.8	52.0	**50.8**	**50.8**	**50.8**	**50.8**
	150	70.6	79.6	80.6	78.4	78.8	**78.2**	78.6
	200	92.6	105.4	106.0	**103.6**	104.4	103.8	104.2
	250	115.8	131.4	132.4	**130.0**	130.6	130.2	130.4
5	50	16.6	19.2	19.4	**18.8**	19.2	**18.8**	19.2
	100	33.2	37.8	39.0	36.6	36.4	**36.2**	36.8
	150	49.8	57.8	59.4	**55.4**	55.6	55.6	**55.4**
	200	67.8	76.6	79.2	74.0	75.0	**73.4**	74.2
	250	84.4	95.6	98	93.2	93.6	**92.0**	93.4
6	50	17	18.4	18.6	18.4	18.2	18.4	**18.0**
	100	33.4	36.0	37.0	35.6	35.6	35.8	**35.4**
	150	49.8	54.0	55.2	53.6	53.6	**53.2**	**53.2**
	200	66.8	71.6	72.8	**70.4**	70.6	**70.4**	70.8
	250	83.4	88.4	89.8	88.4	88.0	88.2	**87.6**

Table 5.6: The average number of wavelengths and cost of the multicast trees established by the proposed algorithms and the lower bounds for the B network data set from [45] for cases with 30 multicast requests and $\beta = 2$.

Networks	*Lower bounds*	*FF_ MC_ RWA*	*BF_ MC_ RWA*	*FFD_ MC_ RWA*	*BFD_ MC_ RWA*	*FFTD_ MC_ RWA*	*BFTD_ MC_ RWA*
Wavelengths Used							
B1,B2,B3	18.93	27.13	27.13	**27.07**	27.20	27.20	27.13
B4,B5,B6	17.6	21.47	21.40	**20.93**	21.00	21.00	**20.93**
B7,B8,B9	19.6	29.93	**28.00**	**28.00**	28.33	28.33	28.47
B10,B11,B12	17.27	21.3	21.73	20.87	**20.73**	21.00	21.00
B13,B14,B15	18.87	27.47	27.53	**27.27**	27.87	27.33	27.80
B16,B17,B18	18.47	23.40	23.47	23.47	**23.33**	23.67	**23.33**
Average Cost of Established Light-trees							
B1,B2,B3	84.49	145.43	145.28	145.39	**145.05**	145.32	145.06
B4,B5,B6	52.47	115.74	114.81	117.52	113.21	117.48	**112.82**
B7,B8,B9	111.07	212.57	212.40	212.50	211.84	212.17	**211.65**
B10,B11,B12	101.12	169.20	166.50	172.10	**165.30**	171.57	165.48
B13,B14,B15	157.24	285.46	285.28	286.08	**284.66**	286.09	284.76
B16,B17,B18	93.23	204.43	203.08	206.08	**199.65**	206.11	200.57

Table 5.7: The B network data set from [45]

Networks	*Nodes*	*Edges*	*Avg. Degree*
B1, B2, B3	50	63	1.26
B4, B5, B6	50	100	2
B7, B8, B9	75	94	1.2533
B10, B11, B12	75	150	2
B13, B14, B15	100	125	1.25
B16, B17, B18	100	200	2

Chapter 6

Virtual Topology Design

In this chapter we consider the design of virtual topologies in wavelength routed WDM optical networks. Recall that this includes determining a set of potential lightpaths and then solving the RWA problem for this set. We refer to a combination of these subproblems as the *V*irtual topology and *R*outing and *W*avelength *A*ssignment problem ($VRWA$). Finally, packet switched traffic must be routed over the established virtual topology. This will be referred to as *T*raffic *R*outing (TR). Determining a good virtual topology with respect to various optimization criteria is a complex problem. Most algorithms suggested for virtual topology design are evaluated by considering a single optimization criterion to be the measure of quality of their obtained solutions. In this chapter, we discuss various objectives for virtual topology design and introduce an objective criterion which we call *virtual hop distance* which is independent of the traffic matrix. We discuss its importance and derive an effective lower bound.

The majority of approaches used to solve the virtual topology design problem decompose it into sub-problems and use highly intractable MILP formulations. These problems are most often solved using LP-relaxations and various rounding techniques. Here, we suggest alternative rounding schemes used to determine virtual topologies from LP-relaxations. These methods have shown to be effective not only for congestion, but with respect to packet and virtual hop distances as well. Routing and wavelength assignment is not considered. To determine virtual topologies *and* perform routing and wavelength assignment (i.e. the $VRWA$ problem), we propose very effective yet simple and fast greedy algorithms motivated by $HLDA$ [76]. The variations between the algorithms are each meant to better satisfy different optimization criteria. Traffic routing over the virtual topology is done using the LP formulation suggested in [48].

We analyze the performance of the proposed algorithms with respect to several aspects

of the obtained solutions. To asses their quality, we compare some of our results to that of existing algorithms for virtual topology design and with their respective analytical lower bounds. Furthermore, we analyze the benefits and drawbacks of establishing multiple lightpaths between pairs of nodes. We discuss the trade-offs associated with each algorithm and the network scenarios in which it may perform best.

The rest of this chapter is organized as follows. In Section 6.1, we informally define the $VRWA$ problem and discuss related work in Section 6.2. Various objective criteria and lower bounds are discussed in Sections 6.3 and 6.4, respectively. Alternative rounding schemes used to determine virtual topologies from solutions obtained by LP-relaxations are presented in Section 6.5. In Section 6.6, we suggest greedy heuristic algorithms for the $VRWA$ problem. Numerical results and a detailed analysis of the obtained results are given in Sections 6.7 and 6.8. We finish with some suggestions for further research and a chapter summary in Section 6.9.

6.1 Problem Definition

The physical optical network is modelled as a graph $G_p = (V, E_p)$, where V is the set of nodes ($|V| = N$) and E_p is the set of physical edges. Edges are assumed to be bidirectional (each representing a pair of optical fibers - one fiber per direction) and have assigned weights representing their length or cost. Given is a long term traffic matrix $\Lambda = (\lambda^{sd}), s, d \in V$, where each element represents the average traffic flow from a source node s to a destination node d. The number of available wavelengths W on each link, and the number of available transmitters Tr and receivers Re at each node, are given. We include an additional parameter (value h) which represents an upper bound on the physical length[1] of a lightpath.

The $VRWA$ problem searches for a set of lightpaths which creates a virtual topology on top of the physical topology. The virtual topology can be modelled as a directed graph $G_v = (V, E_v)$. Each directed edge in E_v represents one lightpath $(i, j), i, j \in V$, defined by the source node i and destination node j of the lightpath. No more than Tr lightpaths can share the same source node, and no more than Re lightpaths can share the same destination node. In other words, Tr and Re are the maximum out-degree and in-degree, respectively, of any node in G_v. Furthermore the $VRWA$ problem searches for a set of physical paths $P = \{P_1(i_1, j_1), \ldots, P_{|E_v|}(i_{|E_v|}, j_{|E_v|})\}$ in G_p, each corresponding to one lightpath or virtual link from G_v, and assigns wavelengths to these paths. Paths $P_k(i_k, j_k)$ and $P_l(i_l, j_l)$ where $k \neq l$, $k, l = 1, \ldots, |E_v|$, cannot be assigned the same wavelength if they share a common

[1]Length can be considered in terms of hops or actual distance.

edge in G_p. At most W distinct wavelengths can be assigned to the paths in P. The length of any path $P_k(i_k, j_k)$, $k = 1, \ldots, |E_v|$, is upper bounded by value h.

There are several objectives to consider when solving the $VRWA$ problem. The most common is to design such a virtual topology G_v and the corresponding routing and wavelength assignment which enables traffic Λ to be routed over G_v with the minimal congestion. It is also desirable that the virtual topology have small packet and virtual hop distances and yet consist of a small number of lightpaths to reduce total transceiver cost. With respect to routing and wavelength assignment, the number of distinct wavelengths used and the lengths of physical routes of individual lightpaths should both be minimized.

6.2 Related Work

In networks equipped with wavelength converters, the virtual topology design problem is less complex since the wavelength continuity constraint does not apply. An exact mixed integer linear formulation (MILP) for complete virtual topology design in WDM networks with full wavelength conversion is given in [5]. The objective is to minimize the average packet hop distance. Heuristic algorithms for the same problem are suggested in [66].

In [76], the authors formulate a MILP for virtual topology design with the objective to minimize congestion. There is no constraint on the number of wavelengths used. The authors suggest various heuristic algorithms, the best of which are the LP Logical Design Algorithm ($LPLDA$) and the Heuristic Topology Design Algorithm ($HLDA$). $LPLDA$ relaxes the integer constraints in the MILP formulation and rounds the variables representing the virtual topology. Routing and wavelength assignment is not considered.

$HLDA$ has become a well-known heuristic algorithm for the *VRWA* problem which considers a limited number of wavelengths in networks with no wavelength conversion. Traffic Routing (TR) is solved subsequently using an LP formulation which minimizes congestion. $HLDA$ attempts to establish lightpaths between pairs of nodes in decreasing order of their corresponding traffic. These lightpaths are routed on the shortest available path and assigned the lowest available wavelength found on that path. After establishing a lightpath between a pair of nodes, the value of their corresponding traffic is decreased by the value of the next highest traffic demand and all the demands are again sorted in decreasing order. This allows multiple lightpaths to be established between pairs of nodes with high traffic. After the procedure terminates, transceivers may be left over at some nodes in the network. If such is the case, $HLDA$ establishes lightpaths at random between these nodes until all the available resources are exhausted. This algorithm, although simple, performs very well with respect

to congestion.

In [100] the authors propose a heuristic for the $VRWA$ problem which does the following. Lightpath routing is predetermined such that each potential lightpath is routed on its shortest physical hop path. Wavelengths are subsequently assigned to as many lightpaths as possible (without violating the wavelength clash and continuity constraints) in descending order of traffic. Degree constraints (transceivers) are not considered. This approach performs well with respect to resource utilization although it tends to lead to unconnected virtual topologies when resources are scarce. A drawback is that lightpaths are routed on predetermined paths which significantly limits the possibilities.

In [73], the authors propose a method to reduce the complexity of the MILP formulations for lightpath selection and routing. An approach to solving these two subproblems in a combined manner is suggested. In [50], the authors propose a tabu search meta-heuristic algorithm for defining the set of lightpaths to be established and routing packet switched traffic over them. The trade-offs associated with establishing more expensive virtual topologies with smaller congestion and *vice versa* are studied.

A MILP which minimizes congestion in networks with a limited number of wavelengths and no wavelength converters is given in [48]. This formulation is not computationally tractable, hence a heuristic approach is suggested. The MILP is relaxed and iteratively run 25 times using a cutting plane. The variables representing the virtual topology and physical paths are rounded while a wavelength assignment heuristic is applied to assign wavelengths to individual lightpaths. Traffic is routed over the virtual topology using a linear programming formulation (LP) consisting of only the traffic constraints of the relaxed MILP. We will refer to this heuristic as $MILP+WA$. One of the drawbacks of the $MILP+WA$ heuristic is the following. Supposing there are W available wavelengths on each fiber, the relaxed MILP obtains a solution which satisfies this constraint. However, since the wavelength assignment algorithm which is subsequently applied gives suboptimal solutions, it does not guarantee a successful wavelength assignment with at most W wavelengths. As a result, the $MILP + WA$ algorithm does not necessarily give feasible solutions for all cases.

6.3 Optimization Criteria in Virtual Topology Design

A brief description of various optimization criteria in virtual topology design follows.

6.3.1 Congestion

The most common optimization criterion in virtual topology design is the minimization of congestion. Congestion is defined as the maximum traffic load on any virtual link.

6.3.2 Packet Hop Distance

If delay is an important issue, it is desirable to minimize the average number of lightpaths traversed by a unit of traffic (packet) on its path from source to destination in the virtual network. This is called the average packet hop distance and is a function of the virtual topology and the long term traffic matrix.

6.3.3 Wavelengths Used

In order to leave more room for future expansion of the virtual topology, minimizing the total number of distinct wavelengths used is desirable. In [56], the authors consider the maximum number of lightpaths routed on any physical link to be a measure of the expandability of the virtual topology. This is equal to the maximum number of wavelengths used on any link, and is essentially the lower bound on the total number of distinct wavelengths used. This measure of network expandability is only sufficient if the network is equipped with wavelength converters at each node. If the network lacks wavelength converters, a request to add a new lightpath may be rejected even though there exists a path on which all links have available wavelengths due to the unavailability of the *same* wavelength on the entire path. If such is the case, reconfiguration or wavelength rerouting[2] must be performed otherwise the request is blocked. As a result, it seems that minimizing the total number of distinct wavelengths used, instead of minimizing the maximum number of lightpaths on a physical link, is a more appropriate objective criterion. Using less distinct wavelengths, i.e. leaving more entirely free wavelengths, decreases the chances that a new request will be blocked due to the wavelength continuity constraint.

6.3.4 Transceivers Used

Transmitters and receivers, commonly referred to as transceivers, are fairly expensive. As a result, it is desirable to set up a virtual topology with fewer transceivers (i.e. fewer lightpaths) as long as the congestion and average packet hop distance are acceptable.

[2]Wavelength rerouting is a mechanism that switches a certain number of existing lightpaths to a different wavelength in order to create a wavelength continuous path for a new request.

6.3.5 Physical Hop Length

In opaque networks in which electronic regeneration is performed at each node, minimizing the *physical* hop length of individual lightpaths is important. Such networks require a transmitter and receiver at the head and tail nodes, respectively, of each physical link included in the lightpath. As a result, longer physical paths dramatically increase the cost of the network. In addition, due to signal degradation, the minimization of the physical length of a lightpath, not only in terms of hops but also in terms of actual distance, is desirable in all WDM networks.

6.3.6 Virtual Hop Distance

An optimization criterion that has not been considered in research dealing with virtual topology design is a measure which we refer to as the average *virtual* hop distance. The average virtual hop distance is the average hop distance in the virtual topology between all source - destination pairs. This is a function of the virtual topology alone and is entirely independent of the traffic matrix. We feel that this criterion, in combination with the average *packet* hop distance, is relevant due to the following. If the average *packet* hop distance is low but the average *virtual* hop distance is high, this means that most of the lightpaths are concentrated around a small number of nodes with high traffic. Since traffic can be prone to change, and reconfiguration of the virtual topology can be costly due to service disruption, it seems that such a virtual topology could perform poorly in the long run as traffic changes. On the other hand, if the virtual topology has not only a low average *packet* hop distance but a low *virtual* hop distance as well, we know that *all* the source - destination pairs are fairly well connected. Therefore, in addition to performing well for current traffic trends, the virtual topology would perform well for changing traffic and thus postpone reconfiguration for a longer period of time.

Furthermore, ensuring a finite average virtual hop distance would eliminate unconnected virtual topologies. Suppose that there is zero traffic between a pair of nodes in the current traffic matrix. If such is the case, the hop distance between these nodes would not enter into the calculation of the average *packet* hop distance since there are no packets delivered between these two nodes. Therefore, without considering the average virtual hop distance, the distance between these nodes could be arbitrarily long or the two could even be unconnected. In the case of the latter, not even a single packet could be sent between these nodes without reconfiguration of the virtual topology.

6.3.7 Execution Time

The execution time of virtual topology design algorithms, although not a direct optimization criterion for the virtual topology problem, is certainly an important aspect of the algorithms to consider. Most algorithms proposed for virtual topology design have large execution times and thus become intractable for larger networks.

6.4 Lower Bounds

Since the algorithms considered in this chapter are heuristics which obtain upper bounds on the minimal objective function values, it is useful to have good lower bounds in order to assess the quality of the sub-optimal solutions. In this chapter we develop a lower bound for the average virtual hop distance, and our computational results demonstrate its efficiency. Similar lower bounds were previously developed for the average packet hop distance, and for congestion in [76]. For completeness, we briefly present these lower bounds as well. These lower bounds are functions of the maximum logical degree in the network which the authors consider to be a function of the node capabilities in the network. We suggest considering link capabilities along with node capabilities to obtain lower values for the maximum logical degree for some cases. This may, in turn, improve the lower bounds on congestion and average packet hop distance for these cases. Lower bounds for the number of wavelengths, transceivers and physical hop lengths are not relevant for our particular problem definition, as will be discussed in section 6.4.4.

6.4.1 Lower Bound on the Average Packet Hop Distance

A lower bound on the average packet hop distance for the virtual topology design problem with no constraint on the number of wavelengths is given in [76] and is as follows. Assuming $P = (p_{sd})$ is the average traffic distribution matrix, i.e. p_{sd} is the probability that a packet is from s to d, π_i for $1 \leq i \leq N$ is a permutation of $(1, 2, \ldots, N)$ such that $p_{i\pi_i(j)} \geq p_{i\pi_i(j\prime)}$ if $j \leq j\prime$. If Δ_l is the maximum degree of the virtual topology, the lower bound on the average packet hop distance, which we will refer to as $\overline{H_p^{LB}}$, was shown to be

$$\overline{H_p^{LB}} = \sum_{i=1}^{N} \sum_{k=1}^{m} \sum_{j=n_{k-1}+1}^{N-1} p_{i\pi_i(j)} \tag{6.1}$$

where m is the largest integer such that

$$N > 1 + \Delta_l + \ldots + \Delta_l^{m-1} = \frac{\Delta_l^m - 1}{\Delta_l - 1} \tag{6.2}$$

and $n_k = \sum_{i=1}^{k} \Delta_l^i$ for $1 \leq k \leq m-1$, $n_m = N - 1$ and $n_0 = 0$.

In [76], the maximum degree of the virtual topology Δ_l is considered to be bounded by the node capabilities in the network, i.e. the maximum number of transmitters Tr available at any node, or the maximum number of ports the electronic switch at a node can handle. Since we consider a limited number of wavelengths W on each link, the maximum degree of the virtual topology is not only bounded by node capabilities[3], but by link capabilities as well. Since the virtual degree cannot exceed $W\Delta_p$, where Δ_p is the maximum degree of the *physical* topology, we define the maximum degree of the virtual topology Δ_l to be

$$\Delta_l = min(Tr, W\Delta_p). \tag{6.3}$$

If the virtual topology is required to be a regular topology, the virtual degree is bounded by $W\delta_p$, where δ_p is the *minimum* degree of the physical topology. In that case the virtual degree could be at most

$$\Delta_l = min(Tr, W\delta_p). \tag{6.4}$$

6.4.2 Lower Bound on Congestion

Using the lower bound for the average packet hop distance described above, a lower bound on congestion

$$\lambda_{max}^{LB} = \frac{r \cdot \overline{H_p^{LB}}}{E_l}, \tag{6.5}$$

was derived in [76], where r is the total arrival rate of packets to the network and E_l is the number of directed links in the virtual topology. Better lower bounds for congestion were obtained in [76] and [48] by iteratively solving the LP-relaxations of their respective MILP formulations for the virtual topology problem using a cutting plane.

6.4.3 Lower Bound on the Average Virtual Hop Distance

We now derive a lower bound for the average virtual hop distance which we will refer to as $\overline{H_v^{LB}}$. Since the average virtual hop distance is independent of the traffic matrix, the lower

[3]Here, we will consider the node capabilities to be bounded only by the number of transmitters Tr at each node.

bound on the average virtual hop distance from any node $s \in V$ to all the other nodes in the network is the same for each node s. Therefore, the lower bound on the overall average virtual hop distance in the network is the same as the lower bound for any one node.

As noted in [76], if a network has a maximum logical degree of Δ_l, for some node $s \in V$ there can be at most Δ_l nodes one hop away from s, at most Δ_l^2 nodes two hops away, at most Δ_l^3 nodes three hops away, etc. An ideal virtual topology with respect to virtual hop distance from some node s to the remaining nodes in the network would be such a topology in which node s had Δ_l neighbors, each of which had Δ_l neighbors of their own without creating a cycle, and so on, until all the nodes were connected. This would create a tree structure of degree Δ_l, where only the last non leaf node could have a degree less than Δ_l, depending on the total number of nodes in the network.

Let m be the largest integer such that $N \geq 1 + \Delta_l + \ldots + \Delta_l^{m-1} = \frac{\Delta_l^m - 1}{\Delta_l - 1}$ holds. In the ideal virtual topology with respect to virtual hop distance from node s, Δ_l nodes would be one hop away from s, Δ_l^2 nodes would be two hops away, etc., up until Δ_l^{m-1} nodes that would be $(m-1)$ hops away. The remaining $(N-1) - (\Delta_l + \ldots + \Delta_l^{m-1})$ nodes would be m hops away. It follows that the lower bound on the average virtual hop distance $\overline{H_v^{LB}}$ would be that shown in (6.6).

$$\begin{aligned} \overline{H_v^{LB}} &= \frac{[\Delta_l + 2\Delta_l^2 + \ldots + (m-1)\Delta_l^{m-1}] + m[(N-1) - (\Delta_l + \ldots + \Delta_l^{m-1})]}{N-1} \\ &= \frac{\sum_{k=1}^{m-1} k\Delta_l^k + m[(N-1) - \sum_{k=1}^{m-1} \Delta_l^k]}{N-1} \\ &= \frac{\Delta_l[\frac{(m-1)\Delta_l^m - m\Delta_l^{m-1} + 1}{(1-\Delta_l)^2}] + m(N - \frac{\Delta_l^m - 1}{\Delta_l - 1})}{N-1} \end{aligned} \tag{6.6}$$

6.4.4 Lower Bounds for Wavelengths, Transceivers and Physical Hop Lengths

Lower bounds on the number of wavelengths, transceivers and average physical hop lengths of the lightpaths are not relevant for our particular problem. The reason for this is that according to our problem definition, these lower bounds have a value of 0. In other words, we could have zero lightpaths giving us values of zero for the wavelengths and transceivers used and physical hop lengths of lightpaths. This would, of course, give us infinite values for congestion and average packet and virtual hop distances.

Lower bounds on the number of wavelengths and the physical lengths of the lightpaths make sense when establishing fixed virtual topologies (i.e. for the RWA problem) or establishing virtual topologies with a required regular logical degree. The number of transceivers

in these cases is constant. Lower bounds on the number of wavelengths needed for the RWA problem and for the problem of designing virtual topologies with a fixed logical degree are given in [83] and [76], respectively. A lower bound on the average physical hop length of lightpaths for the RWA problem is simply the average of the shortest physical paths of all the requested lightpaths. A lower bound on the average physical hop length of lightpaths for virtual topologies with a required regular logical degree of Δ_l is the average of all the lengths of the shortest paths from each node $i \in V$ to its Δ_l closest neighbors.

Lower bounds for wavelengths, transceivers, and physical hop lengths also make sense for problems which require congestion, or average packet, or virtual, hop distances to be under certain threshold values. Developing lower bounds for these problems is out of the scope of this thesis and remains an area for further research.

6.5 Alternative Rounding Algorithms to Determine Virtual Topologies From LP-Relaxations

MILP formulations for virtual topology design are most often solved using LP-relaxations and various rounding techniques. Two such approaches, $LPLDA$ [76] and what we refer to as $MILP + WA$ [48], do as follows. The binary variables representing the virtual topology, as well as those describing the routing and wavelength assignment, are relaxed. After solving the LP-relaxation, the virtual topology variables, b_{ij}, where i and j represent the source and destination nodes of a lightpath respectively, are rounded in the following manner. The fractional values for b_{ij} are sorted in decreasing order and sequentially rounded to 1 if the degree constraints are not violated (i.e. there are available transceivers). If a variable b_{ij} is set to 1, that means that a lightpath will be established between nodes i and j.

Such a rounding scheme may not be very effective if there is a limited number of transceivers available in the network. The reason for this is that, in such cases, variables representing lightpaths between nodes with high traffic may not get a chance to be considered. For example, suppose 20 lightpath variables all have a value of 0.9. It is possible that some of these potential lightpaths never get a chance to be considered if the preceding lightpaths use up the available transceivers and are thus rejected. If these 20 lightpaths were simply considered in random order, it is possible that the traffic between the source-destination nodes of the rejected potential lightpaths is significantly higher than that between the established lightpaths. In such cases, this high traffic would have to be routed over multiple lightpaths, instead of being directly connected by a lightpath. Since establishing lightpaths between

nodes with high traffic (maximizing single hop traffic) has been shown to significantly lower congestion, as well as the average packet hop distance, we think that taking traffic trends into consideration when rounding variables obtained by solving LP-relaxations may yield better results. Giving 'high traffic' lightpath variables the advantage could prove worthwhile not only if variables have the *same* fractional values, but even in cases where the 'high traffic' variables have *lower* fractional values than the 'low traffic' variables. In fact, rounding some 'high traffic' variables to 1 even if their relaxed values are below 0.5 proved to be effective.

We propose the following rounding algorithms.

6.5.1 The TW_LPLDA Algorithm

The fractional virtual topology variables $b_{i,j}$ obtained by solving the LP-relaxation of the MILP in [76] are multiplied by the values of their respective traffic $\lambda_{i,j}$. The variables are then sorted in decreasing order and sequentially rounded to 1 if the degree constraints are not violated (i.e. there are available transceivers). We will refer to this algorithm as TW_LPLDA, for *T*raffic *W*eighed $LPLDA$. Alternatively, the variables could be multiplied by some factor representing the relative values of traffic normalized to a lower value, and in this manner vary the influence of traffic on the rounding procedure.

6.5.2 The $FRHT$ Algorithm

The *F*lexible *R*ounding of *H*igh *T*raffic algorithm is as follows. The virtual topology variables b_{ij} obtained by solving the LP-relaxation of the MILP in [76] are sorted in decreasing order of their corresponding single hop traffic, i.e. $\lambda_{i,j}$. Each variable is then rounded to 1 in sequential order if its value is greater than some parameter a, where $0 \leq a \leq 1$, and if the degree constraints are not violated. The algorithm could be run multiple times (depending on the size of the network and the acceptable execution time) with various values for a, and the best virtual topology could be selected.

6.6 Heuristic algorithms for the $VRWA$ Problem: TSO_SP, TSO_FS, $TSBS\_SP$, $TSBS\_FS$

We propose four fast greedy algorithms for the $VRWA$ problem in networks with no wavelength converters. Traffic Routing over the virtual topology is solved subsequently using the same LP which minimizes congestion used by the $MILP + WA$ heuristic in [48].

6.6.1 The TSO_SP Algorithm

The TSO_SP algorithm is a simple virtual topology design algorithm where *T*raffic is *S*orted *O*verall and routed on the *S*hortest *P*ath available. Algorithm TSO_SP is similar to the $HLDA$ algorithm suggested in [76] except that it does not establish multiple lightpaths between nodes, and does not subsequently assign lightpaths at random until all the transceivers or wavelengths are exhausted. The reason for this is that our objectives include minimizing the number of transceivers and wavelengths used.

The algorithm is as follows. Since we have W available wavelengths, suppose W copies of graph G_p referred to as $G_p^1, \ldots, G_p^W$ each representing one wavelength. This 'layered graph' approach was first introduced in [9] for the Routing and Wavelength Assignment problem. For each traffic demand in Λ in decreasing order of the traffic amount, the shortest path available in any graph $G_p^1, \ldots, G_p^W$ is found[4]. Suppose this path is found on graph G_p^w, where $w \in \{1, \ldots, W\}$. If the length of this path is less than h, and there is an available transmitter and receiver at the source and destination nodes respectively, the lightpath is established and assigned wavelength w. The edges found along the path are deleted from graph G_p^w and the number of available transceivers is updated. The procedure terminates when all the transceivers or wavelengths are exhausted, or until we have tried to establish a lightpath between every source - destination pair in the network.

The pseudocode of TSO_SP is shown in Fig. 6.1:

6.6.2 The TSO_FS Algorithm

The TSO_FS algorithm is a virtual topology design algorithm where *T*raffic is *S*orted *O*verall and routed on the *F*irst *S*atisfactory route available. The traffic demands are sorted in decreasing order as in TSO_SP but only *one* copy or layer of graph G_p referred to as G_p^1 is created. After routing the highest traffic demand on its shortest path in G_p^1, to route the second highest traffic demand we again try and route it in G_p^1. If there is a *satisfactory* path in G_p^1 (i.e. if its length is less than h), the lightpath is routed in G_p^1 even though there may be a shorter path in the original graph G which we could have used if we routed the lightpath on a higher layer as in TSO_SP. If there is no satisfactory path in G_p^1, a second copy of G_p, called G_p^2, is created on which we route the lightpath and assign to it wavelength 2. For each subsequent traffic demand, we search for the shortest path in each existing graph in sequential order until the first satisfactory route is found. If no satisfactory route is available on any graph and there are less than W graphs, a new one is created. If there already exist W

[4] If the shortest path exists in more than one graph, the graph representing the lowest wavelength is chosen.

TSO_SP
Input:
$G_p = (V, E_p)$;//*physical network*
Λ;//$N * N$ *long term traffic matrix*
W; //*available wavelengths*
Tr, Re;//*available transmitters and receivers respectively at each node*
h; //*max physical length of lightpath*
Initialization:
$Transmitters := \{t_1, \ldots, t_N\}$, $t_i = Tr$, $i = 1, \ldots, N$; //*available transmitters per node*
$Receivers := \{r_1, \ldots, r_N\}$, $r_i = Re$, $i = 1, \ldots, N$; //*available receivers per node*
$k = 1$; //*index of the potential lightpath under consideration*
Begin:
Sort traffic demands between source - destination pairs in Λ in decreasing order creating a list $\tau = \{(s_1, d_1), \ldots, (s_{|\tau|}, d_{|\tau|})\}$ of potential lightpaths, where $|\tau| = N * (N - 1)$;
Create W copies (layers) of $G_p : G_p^1, \ldots, G_p^W$;
while $k \leq N * (N - 1)$ and there are available tranceivers **do**
  For demand (s_k, d_k) from τ, find its shortest path P_k available in $G_p^1, \ldots, G_p^W$. (If more than one shortest path exists, route on lowest wavelength layer);
  if the length of $P_k < h$ **then**
    Establish lightpath (i, j), where $i = s_k$, $j = d_k$;
    If routed on graph G_p^w, delete from G_p^w all edges in P_k and assign wavelength w to lightpath (i, j);
    $t_i = t_i - 1$; $r_j = r_j - 1$;
  end if
  Remove (s_k, d_k) from τ;
  $k = k + 1$;
end while
End

Figure 6.1: Pseudocode of the TSO_SP algorithm.

graphs and the traffic demand cannot be routed on any graph G_p^1,...,G_p^W, the corresponding lightpath is not established.

The motivation for sequentially 'filling up' wavelengths as described above is to minimize the total number of distinct wavelengths used. This leaves more room for future expansion of the virtual topology. Routing lightpaths in such a manner may result in longer physical paths as a trade off to using less wavelengths. This problem is solved by bounding the physical length of the lightpaths (value h) with an acceptable value. Separate hop bounds for each source - destination could also be specified. Furthermore, TSO_FS is faster than TSO_SP since it routes lightpaths on the first found satisfactory route, instead of searching for the overall shortest path. This difference in execution time may be significant for larger networks, particularly when a large number of wavelengths and transceivers are available.

The pseudocode of TSO_FS is hoen in Fig. 6.2:

6.6.3 The $TSBS\_SP$ Algorithm

The $TSBS\_SP$ algorithm is a virtual topology design algorithm where *T*raffic is *S*orted *B*y *S*ource and routed on the *S*hortest *P*ath available. When resources are scarce it seems the above mentioned algorithms could obtain unconnected or poorly connected virtual topologies where lightpaths are concentrated around a small number of nodes with high traffic. Such solutions may be infeasible. Intuitively it seems that a virtual topology more evenly

TSO_FS
Input:
$G_p = (V, E_p)$;//*physical network*
Λ;//$N * N$ *long term traffic matrix*
W; //*available wavelengths*
Tr, Re;//*available transmitters and receivers respectfully at each node*
h; //*max physical length of lightpath*
Initialization:
$Transmitters := \{t_1, \ldots, t_N\}$, $t_i = Tr$, $i = 1, \ldots, N$; //*available transmitters per node*
$Receivers := \{r_1, \ldots, r_N\}$, $r_i = Re$, $i = 1, \ldots, N$; //*available receivers per node*
$k = 1$; //*index of the potential lightpath under consideration*
Begin:
Sort traffic demands between source - destination pairs in Λ in decreasing order creating a list $\tau = \{(s_1, d_1), \ldots, (s_{|\tau|}, d_{|\tau|})\}$ of potential lightpaths, where $|\tau| = N * (N - 1)$;
Create 1 copy (layer) of $G_p : G_p^1$;
$GRAPHS := \{G_p^1\}$;
while $k \leq N * (N - 1)$ and there are available tranceivers **do**
  For first demand (s_k, d_k) in τ, find the shortest path in each graph in $GRAPHS$ in sequential order until the first satisfactory path P_k is found
  if the length of $P_k < h$ **then**
    Establish lightpath (i, j), where $i = s_k$ and $j = d_k$;
    If routed on graph G_p^w, delete from G_p^w all edges in P_k and assign wavelength w to lightpath (i, j);
    $t_i = t_i - 1$; $r_j = r_j - 1$;
  else if the length of $P_k \geq h$ and $|GRAPHS| < W$ **then**
    Create $G_p^{|GRAPHS|+1}$ and route lightpath (i, j), where $i = s_k$ and $j = d_k$, on shortest path P_k in $G_p^{|GRAPHS|+1}$;
    Delete edges in P_k from $G_p^{|GRAPHS|+1}$
    Assign wavelength $(|GRAPHS| + 1)$ to lightpath (i, j);
    Add $G_p^{|GRAPHS|+1}$ to $GRAPHS$;
    $t_i = t_i - 1$; $r_j = r_j - 1$;
  end if
  Remove (s_k, d_k) from τ;
  $k = k + 1$;
end while
End

Figure 6.2: Pseudocode of the TSO_FS algorithm.

spread out among the nodes may perform better with respect to the average virtual hop distance, particularly when resources are very scarce. This line of thought is the basis for the $TSBS\_SP$ algorithm.

The $TSBS\_SP$ algorithm essentially works the same way as the TSO_SP algorithm, but sorts the traffic demands (potential lightpaths) differently. Here the traffic originating from each source is sorted separately. In other words, we have N separate lists, one for every node, each containing $N-1$ traffic demands to all of the remaining nodes in decreasing order. A new list is created by taking the highest traffic demand from each node, starting with the highest one overall, and continuing in decreasing order. Then the second highest traffic demand from each node is selected and so on. This procedure is repeated until all the traffic demands are in the list. The remaining steps of the algorithm are identical to those of the TSO_SP algorithm described in Section 6.6.1. The pseudocodes of the algorithms are the same except for the method of sorting the traffic demands.

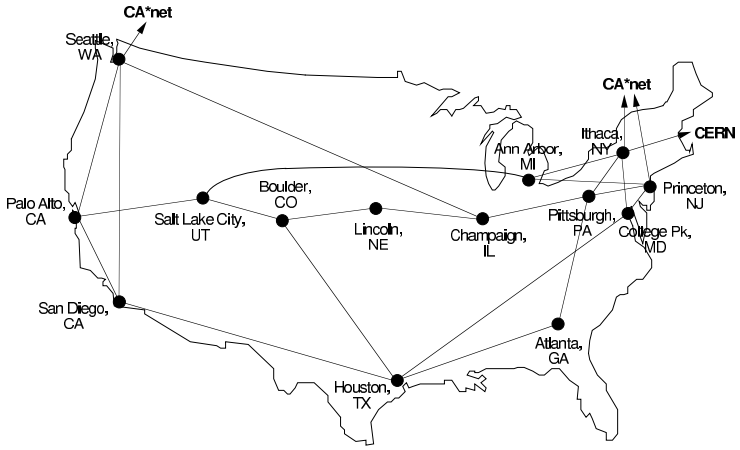


Figure 6.3: The 14-node NSF network

Table 6.1: Comparison of the congestion obtained using various rounding techniques in the NSF network for traffic matrix p1.

T	LB	LPLDA [13]	[2]	MILP+WA [6]	FRHT				TW_LPLDA
					a=0.2	a=0.3	a=0.4	a=0.5	
2	126.18	243.43	147.68	**145.74**	181.98	218.29	244.00	231.93	191.91
3	84.53	102.82	88.65	**84.58**	91.76	93.93	93.93	102.89	91.61
4	63.43	82.03	65.91	70.03	67.38	**65.85**	72.52	66.12	65.86
5	50.75	53.49	51.85	50.94	51.56	51.59	**50.75***	**50.75***	**50.75***
6	42.29	44.45	42.66	44.39	42.78	**42.29***	43.11	43.01	45.34
7	36.25	36.55	36.45	36.43	37.68	**36.25***	**36.25***	**36.25***	**36.25***
8	31.72	32.27	31.75	31.77	33.28	31.92	31.98	**31.72***	32.39

6.6.4 The $TSBS\_FS$ Algorithm

The $TSBS\_FS$ algorithm is a virtual topology design algorithm where *T*raffic is *S*orted *B*y *S*ource and routed on the *F*irst *S*atisfactory path available. The $TSBS\_FS$ algorithm sorts the traffic demands as done by the $TSBS\_SP$ algorithm but routes lightpaths on the first satisfactory route as done by the TSO_FS algorithm. The pseudocode of the $TSBS\_FS$ algorithm is identical to that of TSO_FS except for the method of sorting the traffic demands.

Table 6.2: Comparison of the congestion obtained using various rounding techniques in the NSF network for traffic matrix p2.

T	LB	LPLDA [13]	[2]	MILP+WA [6]	FRHT				TW_LPLDA
					a=0.2	a=0.3	a=0.4	a=0.5	
2	282.51	345.42	345.42	389.93	467.24	496.31	**329.67**	360.16	530.11
3	189.62	195.71	195.71	217.8	**189.78**	**189.78**	**189.78**	**189.78**	**189.78**
4	142.32	**142.33**	**142.33**	152.99	**142.33**	**142.33**	**142.33**	**142.33**	**142.33**
5	113.87	**113.87***	**113.87***	**113.87***	**113.87***	**113.87***	**113.87***	**113.87***	**113.87***
6	94.89	**94.89***	**94.89***	**94.89***	**94.89***	**94.89***	**94.89***	**94.89***	**94.89***
7	81.33	**81.33***	**81.33***	**81.33***	**81.33***	**81.33***	**81.33***	**81.33***	**81.33***
8	71.17	**71.17***	**71.17***	**71.17***	**71.17***	**71.17***	**71.17***	**71.17***	**71.17***

6.7 Numerical Results

6.7.1 Results for Alternative Rounding Algorithms

We tested the TW_LPLDA and $FRHT$ algorithms for the 14 node NSF network shown in Fig. 6.3. Two traffic matrixes, p1 and and p2, which correspond to Tables III and IV in [76], were considered. These traffic matrixes were used to test the $LPLDA$ and $MILP + WA$ heuristics in [76] and [48], respectively. In traffic matrix p1, most of the traffic is concentrated around 42 pairs of nodes, while traffic in p2 is more evenly distributed. The number of transceivers T ranged from 2 to 8. T represents the number of transmitters and the number of receivers at each node, i.e. we assume that $Tr = Re = T$. T is therefore the maximum in-degree as well as the maximum out-degree of each node in the virtual topology. The LP-relaxations of the MILP formulation given in [76] were solved using the CLPEXv6 solver. After determining the virtual topology using the proposed rounding techniques, traffic is routed over the virtual topology using an LP to minimize congestion with only the traffic constraints in [48]. Routing and wavelength assignment is not considered.

$FRHT$ was run with different values for a ranging from 0 to 1 in 0.05 increments. The best results were obtained when a ranged from 0.25-0.45. We show results with a=0.2, 0.3, 0.4 and 0.5. The values for congestion obtained for traffic matrixes p1 and p2 by the TW_LPLDA and $FRHT$ rounding techniques are shown in Tables 6.1 and 6.2, respectively. The lower bounds and results obtained by $LPLDA$ and $MILP + WA$ from [76] and [48], respectively, are also shown. In [48], the authors compare their results to those in [38], so these results are included in the tables. Since the $MILP + WA$ algorithm has a limited number of wavelengths, the best obtained solutions from Tables V and VI in [48] are shown. It is important to note that the $MILP + WA$ algorithm does not necessarily give

Table 6.3: Comparison of the average packet hop distances obtained using various rounding techniques in the NSF network for traffic matrix p1.

T	LB	LPLDA [13]	FRHT				TW_LPLDA
			a=0.2	a=0.3	a=0.4	a=0.5	
2	1.22	2.52	**1.86**	2.38	2.58	2.52	2.30
3	1.10	1.88	1.67	1.69	**1.65**	1.88	1.67
4	1.06	1.67	**1.56**	1.60	1.79	1.67	**1.56**
5	1.04	1.92	1.52	**1.50**	1.53	1.54	**1.50**
6	1.02	1.63	**1.57**	**1.57**	1.60	1.58	1.59
7	1.01	1.57	**1.56**	**1.56**	1.57	1.60	1.57
8	1.01	1.82	1.65	**1.59**	1.57	**1.59**	1.61

better results as the number of wavelengths increases. In fact, for most cases when we tested the algorithm with W higher than that shown in [48], the algorithm gave the same or even poorer solutions. The $MILP + WA$ heuristic seems to perform best when W is between $T - 1$ and $T + 1$.

The best obtained solution for each case is marked in bold. If the obtained solution is equal to the lower bound, i.e. the obtained solution is optimal, it is marked as '*'. We can see that for traffic matrix p1 (Table 6.1), the best solution for cases with the number of transceivers ranging from 4-8 was obtained by at least one of the proposed algorithms. For the cases with 2 and 3 transceivers, $MILP + WA$ performed best. For the number of transceivers ranging from 5-8, at least one of the solutions obtained by the proposed heuristics was optimal. For traffic matrix p2 (Table 6.2), all runs of the heuristic algorithms for the number of transceivers ranging from 3-8 performed better than previously suggested algorithms, while the $FRHT$ algorithm with $a = 0.4$ performed best in all cases.

Since the average packet and virtual hop distances are functions of the obtained virtual topologies, we show these results as well. The average packet hop distances for traffic matrixes p1 and p2 for the proposed heuristics are shown in Tables 6.3 and 6.4, respectively. Since we ran our rounding techniques on the LP-relaxation used by $LPLDA$, we found the average packet and virtual hop distances for the the virtual topologies obtained by $LPLDA$ as well. In all cases, the best solution was found by either TW_LPLDA or $FRHT$. The average virtual hop distances for traffic matrixes p1 and p2 are shown in Tables 6.5 and 6.6, respectively. For all cases, at least one of the proposed approaches obtained a solution better than or equal to that obtained by $LPLDA$.

Table 6.4: Comparison of the average packet hop distances obtained using various rounding techniques in the NSF network for traffic matrix p2.

T	LB	LPLDA [13]	FRHT				TW_LPLDA
			a=0.2	a=0.3	a=0.4	a=0.5	
2	1.22	1.66	2.97	3.36	3.54	**2.72**	3.01
3	1.10	1.37	2.45	2.50	2.45	2.54	**2.37**
4	1.06	1.28	2.58	2.57	2.42	2.57	**2.37**
5	1.04	1.21	2.36	2.37	**2.35**	2.51	2.38
6	1.02	1.15	2.51	2.54	2.46	2.45	**2.44**
7	1.01	1.10	**2.47**	2.50	2.50	2.48	2.51
8	1.01	1.06	2.60	2.56	2.62	2.66	**2.42**

Table 6.5: Comparison of the average virtual hop distances obtained using various rounding techniques in the NSF network for traffic matrix p1.

T	LB	LPLDA [13]	FRHT				TW_LPLDA
			a=0.2	a=0.3	a=0.4	a=0.5	
2	2.38	2.88	3.05	2.92	**2.85**	2.88	2.90
3	1.85	2.10	**2.08**	2.09	2.10	2.10	2.13
4	1.69	**1.78**	1.81	1.82	1.83	**1.78**	1.85
5	1.62	1.65	1.66	1.65	**1.63**	1.64	1.65
6	1.54	**1.54***	**1.54***	**1.54***	**1.54***	**1.54***	1.55
7	1.46	1.47	1.48	1.47	1.47	**1.46***	**1.46***
8	1.38	1.39	**1.38***	1.39	1.40	1.39	1.39

6.7.2 Results for Heuristic Algorithms for the $VRWA$ Problem

The TSO_SP, TSO_FS, $TSBS\_SP$, and $TSBS\_FS$ algorithms for the $VRWA$ problem were implemented in C++ and run on a PC powered by a P4 2.8GHz processor. To solve the LP for traffic routing, the CPLEXv6 solver was used. The algorithms were tested on data from the NSF backbone and the European optical core networks shown in Figures 6.3 and 3.3 respectively. Both of these networks consist of 14 nodes. The algorithms were tested for the two already mentioned traffic matrixes: p1 and p2.

Furthermore, the algorithms were tested on 5 randomly generated 30 node networks for which the probability P_e of there being an edge between node was set to 0.2. Two types of traffic matrixes were generated for each test case. The first was a uniform traffic matrix

Table 6.6: Comparison of the average virtual hop distances obtained using various rounding techniques in the NSF network for traffic matrix p2.

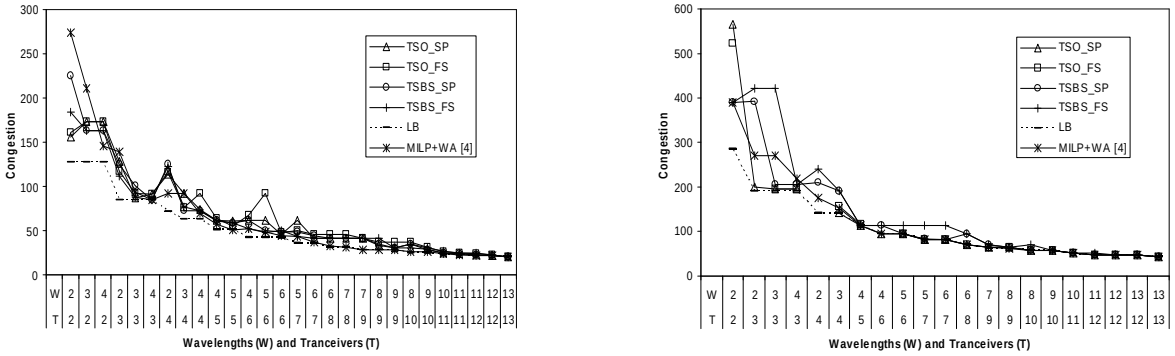
T	LB	LPLDA [13]	FRHT				TW_LPLDA
			a=0.2	a=0.3	a=0.4	a=0.5	
2	2.38	2.98	2.96	2.92	**2.80**	2.98	3.05
3	1.85	2.05	**2.02**	2.03	2.07	2.05	2.14
4	1.69	1.87	**1.82**	1.85	1.85	1.86	1.86
5	1.62	1.67	1.68	1.69	1.67	**1.66**	1.68
6	1.54	**1.54***	1.55	1.55	1.55	**1.54***	**1.54***
7	1.46	1.50	**1.48**	1.49	1.50	1.49	1.49
8	1.38	1.41	**1.40**	**1.40**	1.41	1.41	1.41

with traffic uniformly distributed over $[0, 100]$ for each source-destination pair. The second type of traffic matrix, which we will refer to as nonuniform, was generated by the method used in [5] where a fraction F of the traffic is uniformly distributed over $[0, C/a]$ while the remaining traffic is uniformly distributed over $[0, C*\Upsilon/a]$. The values were set to $C = 1250$, $a = 20$, $\Upsilon = 10$ and $F = 0.7$ as in [5].

We ran the TSO_SP, TSO_FS, $TSBS\_SP$, and $TSBS\_FS$ algorithms with the number of transceivers T ranging from 2 to 13. The number of wavelengths W ranged from $W = T - 1$ to $W = T + 1$ for each value of T. Value h which restricts the physical length of a lightpath was set to $\max(diam(G_p), \sqrt{|E_p|})$ as suggested in [57] for a routing and wavelength assignment algorithm. Length was considered in terms of hops.

In addition, to compare the congestion of the solutions obtained by the proposed algorithms with those obtained by the $MILP + WA$ algorithm (since it considers a limited number of wavelengths), we ran the proposed algorithms with the values T and W corresponding to those in [48]. In Fig. 6.4, we plot the corresponding values for congestion for traffic matrixes (a) p1 and (b) p2. The lower bound on congestion, denoted as LB, and the congestion obtained by the $MILP + WA$ heuristic are taken from Tables V and VI in [48].

The TSO algorithms performed better than the $TSBS$ algorithms for traffic matrix p2 (Fig. 6.4.b), but had a few peaks when run for traffic matrix p1 (Fig. 6.4.a). All the greedy algorithms give similar results as the number of transceivers and wavelengths increases and for many cases are close to or equal to those obtained by the $MILP + WA$ heuristic. For a larger number of wavelengths, they often give the optimal solution, particularly for traffic matrix p2. For some cases when resources are very scarce, the greedy algorithms perform better than $MILP + WA$.



(a) (b)

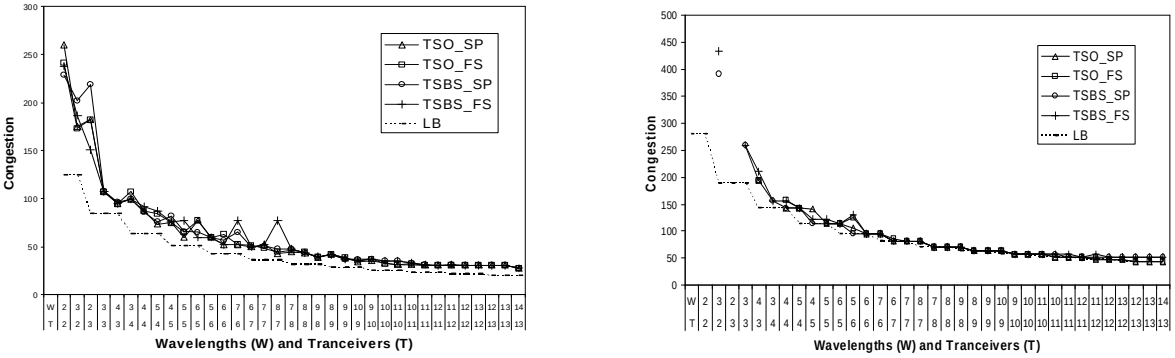
Figure 6.4: Comparison of congestion of the solutions obtained by the TSO_SP, TSO_FS, $TSBS\_SP$, $TSBS\_FS$, and $MILP + WA$ [48] heuristics and the lower bound (LB) for traffic matrix (a) p1 and (b) p2 in the NSF network.

It is important to note that for many cases the $MILP+WA$ algorithm uses more than the available number of wavelengths since wavelength assignment is performed subsequently. This occurs for the following test cases. For p1: (T=3, W=2), (T=4, W=2), (T=6, W=4), (T=6, W=5), (T=7, W=5), (T=9, W=7), (T=9, W=8), (T=10, W=8), and (T=12, W=11), while for p2: (T=2, W=1), (T=4, W=2), (T=6, W=4), (T=7, W=5), (T=9, W=7), (T=10, W=8), (T=10, W=9), (T=12, W=11), and (T=12, W=12). These solutions are thus infeasible.

The execution times between the $MILP + WA$ algorithm and the proposed greedy algorithms differ substantially. As mentioned in [48], the average execution time to solve the relaxed MILP in the $MILP + WA$ heuristic took about 5 minutes on an IBM 43P/RS6000. This is done iteratively 25 times and then a rounding heuristic is used which runs about 1 minute. The execution time of the wavelength assignment heuristic is not mentioned. This means that the average execution time of $MILP + WA$ was at least 5*25+1 = 126 minutes = 2 hours and 6 minutes. The average execution times of the TSO_SP, TSO_FS, $TSBS\_SP$, and $TSBS\_FS$ algorithms for the same test cases when run on a PC powered by a P4 2.8GHz processor were all under half a second[5].

In Fig. 6.5 we plot the congestion of the solutions obtained for the European core network for traffic matrixes (a) p1 and (b) p2 by each of the proposed algorithms and the lower bound obtained by iteratively solving LP relaxations. All 4 algorithms behave similarly, although the SP algorithms perform slightly better than the FS algorithms which tend to have peaks

[5]Note that the same LP was used for Traffic Routing in the $MILP+WA$ heuristic and in combination with the TSO_SP, TSO_FS, $TSBS\_SP$, and $TSBS\_FS$ algorithms, so these execution times were omitted. They ranged from 1 to 20 seconds depending on the values of T and W.

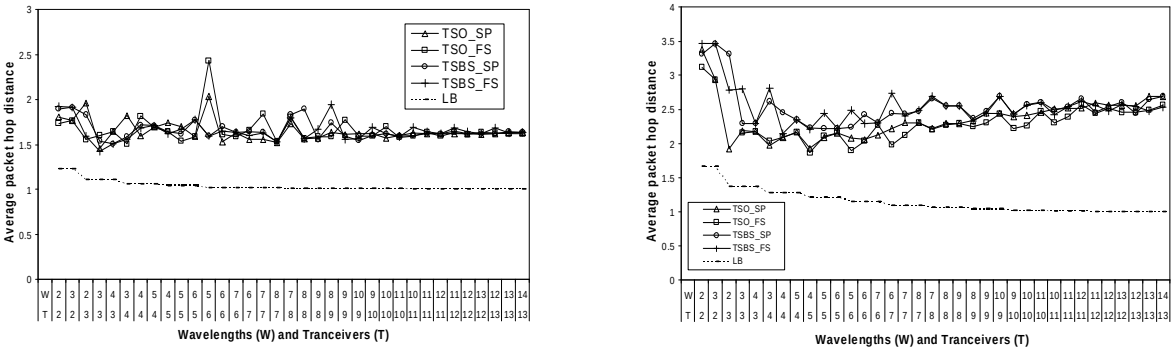


(a) (b)

Figure 6.5: Comparison of congestion of the solutions obtained by the TSO_SP, TSO_FS, $TSBS\_SP$, and $TSBS\_FS$ heuristics for traffic matrix (a) p1 and (b) p2 in the European core network.

for some test cases. All four algorithms performed almost the same with respect to congestion when run for the 30 node networks with nonuniform and uniform traffic. Namely, the 30 node networks, which have an average degree of 6, are denser and better connected than the NSF and European core networks. In such well connected networks there exist several edge disjoint paths and therefore it is possible to set up several lightpaths even when wavelengths are scarce. In other words, for most cases the algorithms terminated when all the available transceivers were exhausted and not as a result of the lack of wavelengths. This means that the virtual topologies obtained by each of the algorithms consist of the same number of lightpaths. Since several alternative paths are available and it is thus unlikely that a lightpath be rejected due to unavailability of a physical route, the method of routing (i.e. SP and FS) does not make much of a difference with respect to the obtained virtual topology. Furthermore, as the number of transceivers at each node increases, the method of sorting lightpaths does not make a significant difference either with respect to the lightpaths established. It follows that the congestion, average packet hop distance, virtual hop distance, and the number of transceivers used, do not differ significantly since these measures are functions of the obtained virtual topology. The differences in the behavior of the proposed algorithms for the 30 node networks are evident with respect to other optimization criteria. These include the number of wavelength used, physical hop length of the lightpaths, and execution time. Here, the SP and FS aspects of the algorithms play a significant role.

When the algorithms were run for the NSF and European core networks, they usually terminated when all the available wavelengths were exhausted. As a result, the established virtual topologies differed somewhat leading to differences in congestion, average packet



(a) (b)

Figure 6.6: Comparison of average packet hop distance of the solutions obtained by the TSO_SP, TSO_FS, $TSBS\_SP$, and $TSBS\_FS$ algorithms for traffic matrix (a) p1 and (b) p2 in the NSF network.

hop distance, average virtual hop distance and the number of transceivers used. The average packet hop distances of the solutions obtained by each of the proposed algorithms in the NSF network are shown in Figure 6.6. The lower bound for the average packet hop distance is also plotted, although it is not very effective. Even though the lower bound decreases as available resources increase, the average packet hop distance of the obtained solutions increases. The reason for this is that after solving the *VRWA* problem using the greedy heuristics, *TR* is solved with an LP which minimizes *congestion*. In solutions to the *VRWA* problem which establish more lightpaths due to more transceivers and wavelengths available, it is possible to route packet switched traffic over longer paths (since more paths are available) to better minimize congestion. This, however, is a trade-off to increasing the average packet hop distance. As a result, the average packet hop distance increases even though the lower bound decreases. The results for the European core network are analogous. We can see from the graphs that the TSO algorithms perform better than the $TSBS$ algorithms for most cases, particularly for traffic matrix p2. This makes sense since the main objective of the TSO algorithms is to establish lightpaths between nodes with the overall highest traffic without considering overall connectivity. As a result, they maximize single hop traffic.

Sorting the traffic overall may not be desirable if traffic is prone to change since the obtained virtual topology may be very poorly connected or even unconnected. This is particularly true for cases when the current traffic matrix has zero or very little traffic flowing between some pairs of nodes, as is the case in traffic matrix p2. In Fig 6.7, the average virtual hop distances of the solutions obtained by the TSO_SP, TSO_FS, $TSBS\_SP$, $TSBS\_FS$ algorithms, and the lower bound for the NSF and European core networks for

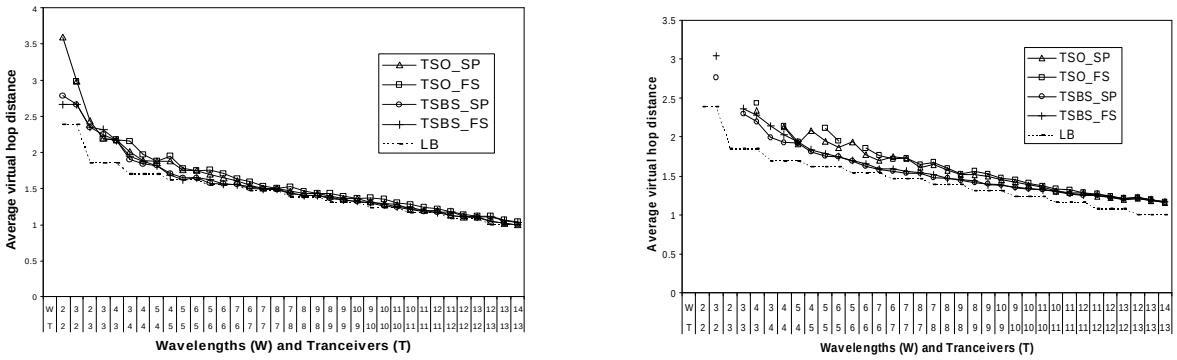


Figure 6.7: Comparison of the average virtual hop distance obtained by the TSO_SP, TSO_FS, $TSBS\_SP$, and $TSBS\_FS$ algorithms in the (a)NSF and (b) European core network for traffic matrix p2.

traffic matrix p2, are shown. Several of the obtained values are close to the lower bound, particularly for the NSF network. For test cases where there is no point plotted, the corresponding algorithm did not obtain a feasible solution (i.e. a connected virtual topology), and thus the average virtual hop distance is infinite. We can see that the $TSBS$ algorithms yield virtual topologies that are better connected overall and as a result may perform better as the traffic matrix changes. The situation is analogous, but the algorithms differ less for traffic matrix p1.

It is logical that the virtual topologies that are better connected overall establish more lightpaths and therefore use more transceivers. The number of transceivers used to create virtual topologies in the NSF and European core networks for traffic matrix p2 are shown in Fig. 6.8. We can see that the TSO algorithms use fewer transceivers than the $TSBS$ algorithms, but as a trade-off to virtual hop distance.

To determine the behavior of the algorithms with respect to virtual connectivity for cases when resources are very scarce, we ran each algorithm for the number of transceivers ranging from 2-5 and the number of wavelengths ranging from 2-4. The cases where the algorithms failed to find feasible solutions for the NSF and European core networks for traffic matrix p2 are shown in Table 6.7. (The column entitled $HLDA^*$ will be explained later on.) We can see that the TSO_SP and TSO_FS algorithms yielded infeasible solutions for the NSF network in 4 and 2 cases respectively, while the $TSBS\_SP$ and $TSBS\_FS$ algorithms obtained feasible solutions in all cases. For the European core network, both TSO algorithms yielded unconnected virtual topologies in 9 cases, while the $TSBS$ algorithms did so in only 3 cases. All algorithms obtained feasible solutions for all cases for traffic matrix p1.

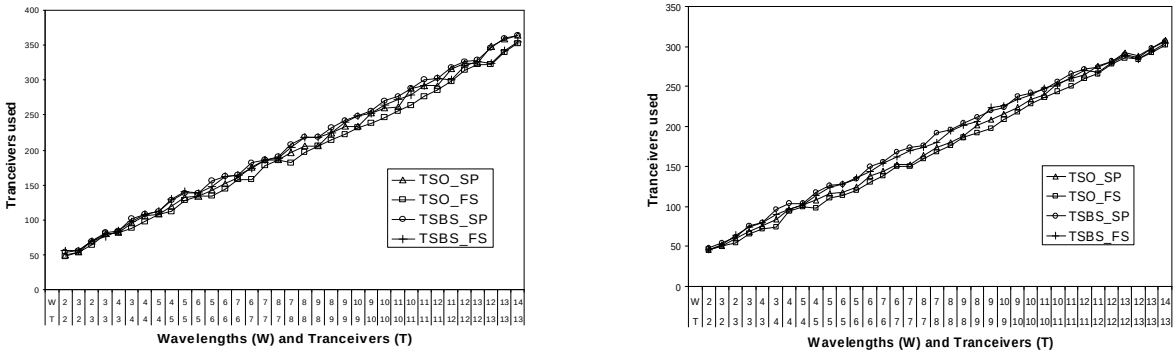


Figure 6.8: Comparison of the number of transceivers used by the TSO_SP, TSO_FS, $TSBS\_SP$, and $TSBS\_FS$ algorithms in the (a) NSF and (b) European core network for traffic matrix p2.

Recall that all four algorithms usually terminated due to lack of available wavelengths when run for the NSF and European core networks. As a result, they use the same number of distinct wavelengths. On the other hand, when run for the 30 node networks the algorithms terminated when all the transceivers were used up. The average number of wavelengths used in the 30 node networks for nonuniform and uniform traffic are shown in Fig. 6.9. We can clearly see the FS algorithms use fewer wavelengths than the SP algorithms. Recall that congestion, average packet and virtual hop distance, and the number of transceivers used, were almost the same for the 30 node networks. Therefore, to establish virtual topologies which perform equally well, the FS algorithms use significantly fewer wavelengths than the SP algorithms. This leaves more room for future expansion of the virtual topology.

Since the FS algorithms route lightpaths on satisfactory, but not necessarily shortest paths, it is to be expected that their corresponding physical hop lengths will be longer. The average lengths of the established lightpaths in the 30 node networks for nonuniform traffic are shown in Fig. 6.10. Results for uniform traffic are analogous. Since we limit the hop distance to an acceptable value using parameter h, this is not necessarily a problem.

To determine the benefits of establishing multiple lightpaths between source-destination pairs with respect to the various objective criteria, we implemented the well known $HLDA$ algorithm [76] and compared it to TSO_SP. $HLDA$ sorts traffic overall, routes it on the shortest path available but also allows multiple lightpaths to be established between nodes with heavy traffic. We eliminated step 4 of the $HLDA$ algorithm which randomly establishes lightpaths until all the transceivers or wavelengths are exhausted. This step is eliminated since two of our objectives for the $VRWA$ problem are to minimize the number of

Table 6.7: Cases where the algorithms failed to find a feasible solution in the (a) NSF and (b) European core networks for traffic matrix p2. (Cases marked 'x' are those where the obtained solutions were infeasible.)

T	W	TSO_SP	TSO_FS	TSBS_SP	TSBS_FS	HLDA*
2	2	**X**				**X**
2	3					
2	4					
3	2	**X**				**X**
3	3					**X**
3	4					
4	2	**X**	**X**			**X**
4	3					
4	4					
5	2	**X**	**X**			**X**
5	3					
5	4					

(a)

T	W	TSO_SP	TSO_FS	TSBS_SP	TSBS_FS	HLDA*
2	2	**X**	**X**			**X**
2	3	**X**	**X**			**X**
2	4	**X**	**X**			**X**
3	2	**X**	**X**	**X**	**X**	**X**
3	3					
3	4	**X**	**X**			**X**
4	2	**X**	**X**	**X**	**X**	**X**
4	3	**X**	**X**			**X**
4	4					
5	2	**X**	**X**	**X**	**X**	**X**
5	3	**X**	**X**			**X**
5	4					**X**

(b)

transceivers and wavelengths used. We refer to this algorithm as $HLDA^*$.

In Fig. 6.11 we plot the (a) congestion, (b) average packet hop distance, (c) number of transceivers used and the (d) average virtual hop distance of the solutions obtained by $HLDA^*$ and TSO_SP for the European core network for traffic matrix p1. The results obtained for traffic matrix p2 as well as the results obtained for the NSF network for both traffic matrixes are fairly similar. With respect to congestion, $HLDA^*$ performed slightly better. This seems logical since $HLDA^*$ can establish multiple lightpaths where traffic is high. Neither algorithm performed consistently better with respect to average packet hop distance. The average packet hop distance obtained by the TSO_SP algorithm seems less dependent on the available number of wavelengths than that obtained by $HLDA^*$ which tends to vary more. In the 30 node networks, $HLDA^*$ and TSO_SP yielded virtual topologies with the same values for congestion and average packet hop distance.

The number of wavelengths used by both algorithms was the same and is therefore not plotted. The TSO_SP algorithm used less transceivers as can be seen in Fig. 6.11.(d) which makes sense since it cannot set up multiple lightpaths between nodes. The average virtual hop distances of the obtained are shown in Fig. 6.11.(c). Even though the though the TSP_SP algorithm established fewer lightpaths (i.e. used fewer transceivers), it obtained virtual topologies with lower average virtual hop distances than $HLDA^*$. As a result, such virtual topologies are cheaper and yet better connected overall. For cases when wavelengths are scarce, $HLDA^*$ performed worst with respect to feasibility when compared to the proposed algorithms (see Table 6.7).

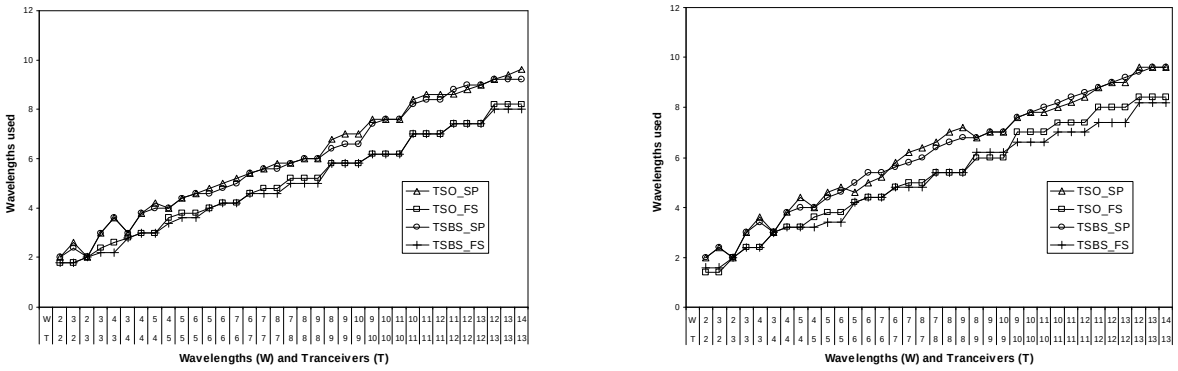


Figure 6.9: Comparison of the average number of distinct wavelengths used in the 30-node networks for (a) nonuniform and (b) uniform traffic.

With respect to execution time, all the proposed algorithms are very fast. For the 30 node networks, all the algorithms ran under half a second. However, the execution times of the SP algorithms grow faster with the number of available resources than those of the FS algorithms. This makes sense since the SP algorithms search for the shortest path available, while the FS algorithms establish a lightpath as soon as a feasible path is found. Since today there can be over 100 wavelengths available, this growth in execution time could be significant, particularly for larger networks. This could especially be important if lightpath requests arrive dynamically over time, requiring new virtual topologies to be created in real time. The differences in execution times of the proposed algorithms with respect to $HLDA^*$ become substantial as the network grows. We ran the $HLDA^*$, TSO_SP and TSO_FS algorithms for a randomly generated network with 250 nodes and the probability of there being an edges between nodes 0.02. The algorithms were run for cases with up to 8 transceivers. The TSO_FS algorithm ran between 5-35 seconds, the TSO_SP ran between 25-135 seconds, while $HLDA^*$ ran approximately an hour.

6.8 Discussion

Several conclusions can be drawn from the obtained results. If congestion is critical and the network is fairly small, relaxing a MILP formulation for virtual topology design and applying the TW_LPLDA or $FRHT$ rounding techniques to obtain good virtual topologies seems the best choice. The $FRHT$ algorithm could be run multiple times with various values for a, and the best virtual topology could be selected. Routing and wavelength assignment would

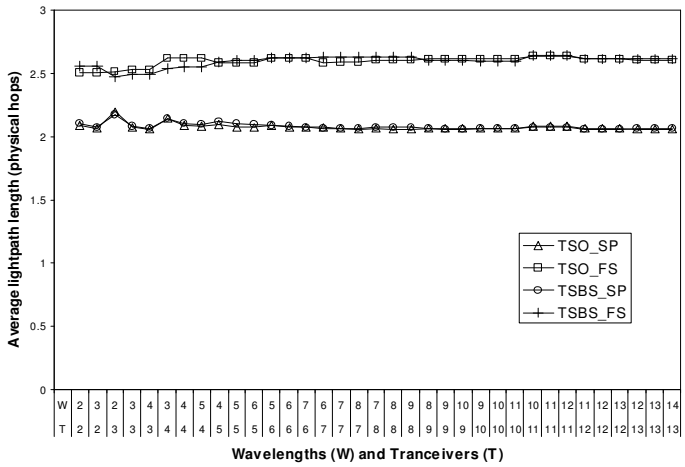


Figure 6.10: Comparison of the average physical length of the established lightpaths in the 30-node networks for (a) nonuniform and (b) uniform traffic.

be performed subsequently using alternative heuristics for the *RWA* problem. This method does not necessarily guarantee a limited number of wavelengths.

For larger networks, such approaches may be intractable. The proposed greedy heuristics are fast and yet perform well. When comparing the effects of sorting the traffic demands differently, we can see that with respect to congestion and average packet hop distance, the TSO algorithms in most cases perform better than, or as well as, the $TSBS$ algorithms. Even so, if traffic is prone to change, $TSBS$ may be the wiser choice since these algorithms yield better connected virtual topologies (i.e. have lower average virtual hop distances). Lower virtual hop distances may postpone the need for reconfiguration for a longer period of time, but as a trade-off to using more transceivers and thus increasing network cost. This is particularly true in sparse networks. If wavelengths are very scarce, $TSBS$ may also help prevent from establishing unconnected virtual topologies. Since all the algorithms are fast, for moderate size networks it may be best to run a TSO algorithm, and if no feasible solution is found, run a $TSBS$ algorithm. If congestion is critical and traffic is predicted to remain constant for a long period of time, establishing multiple lightpaths as done by $HLDA^*$ may be desirable. Otherwise, the gain of using multiple lightpaths does not seem significant, and yet increases the network cost by establishing more lightpaths.

The method of routing and assigning wavelengths does not significantly affect the objective criteria which are functions of the virtual topology. These include congestion and average packet and virtual hop distance. Still, the SP algorithms perform slightly better than the FS algorithms for these criteria. The main advantage of the FS algorithms over the

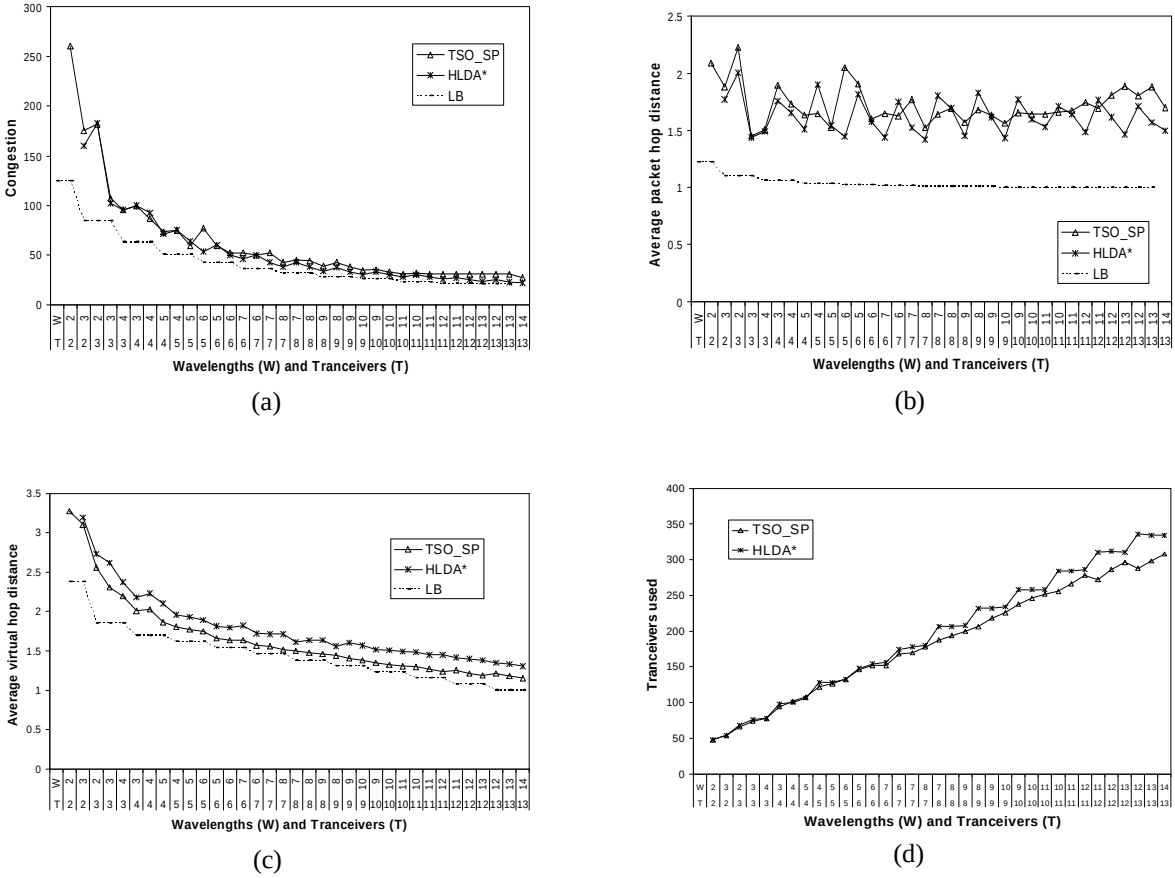


Figure 6.11: Comparison of the (a) congestion and (b) average packet hop distance, (c) transceivers used and (d) average virtual hop distance of the solutions obtained by the TSO_SP and $HLDA^*$ algorithms for traffic matrix p1 in the European core network.

SP algorithms is that they perform routing and wavelength assignment using less distinct wavelengths, particularly in dense networks. If virtual topology expansion is anticipated, FS may be the better choice for routing. This is certainly not the case in opaque networks since the resulting physical lengths of the established lightpaths are longer. For very large networks with many available wavelengths, using an FS algorithm may be desirable due to shorter execution times.

6.9 Summary and Future Work

In order to efficiently utilize resources in wavelength routed optical networks, it is necessary to successfully solve the virtual topology design problem. This problem is very complex and several aspects of the obtained solutions should be considered. In this chapter, efficient

rounding techniques and greedy heuristic algorithms are proposed for the Virtual topology and Routing and Wavelength Assignment problem in networks with no wavelength conversion. The algorithms differ with respect to the order in which the lightpaths are established, and the method of routing and assigning wavelengths. These variations are intended to improve the performance of the algorithms with respect to various objective criteria such as congestion, average physical and packet hop distances, and the number of transceivers and distinct wavelengths used. Furthermore, proposed is a new criterion referred to as the average virtual hop distance, aimed at increasing the connectivity of the virtual topology. A detailed analysis and testing on real and randomly generated networks with uniform and nonuniform traffic indicate the advantages and disadvantages of the suggested variations. Further avenues of research will include developing similar algorithms for virtual topology design in networks with full or limited wavelength conversion. Designing virtual topologies which support multicast traffic by establishing light trees will also be considered.

Conclusion

In this thesis we investigated optimization problems arising in the design of virtual topologies in wavelength routed WDM networks. Specifically, we studied the Routing and Wavelength Assignment (RWA) problem considering static and scheduled lightpath demands, as well as static light-tree demands. Virtual Topology Design which includes determining a virtual topology, RWA and routing packet switched traffic over the virtual topology, was also investigated. Successfully solving these problems is critical to efficiently utilizing resources in optical networks. This is of great importance since the tremendous growth of data traffic incurs an ever-increasing need for high-speed transport networks. Until optical burst and packet switching technology matures, circuit switched (wavelength routed) optical networks are the best candidate to satisfy these high bandwidth requirements.

Following a brief introduction to optical transmission and enabling technologies, as well as a glance at the optical networking evolution, we discussed problems and issues arising in wavelength routed networks. In Chapter 3 we tackled the Routing and Wavelength Assignment problem of static lightpath demands. Highly efficient algorithms developed by applying the classical bin packing problem were presented. The methods used to perform RWA consistently minimize the number of wavelengths used and methods were suggested to minimize the physical lengths of the lightpaths established. Lower bounds were developed to help evaluate the efficiency of the proposed algorithms. Testing indicated that the algorithms give optimal or near optimal solutions in many cases and significantly outperform an established existing algorithm for the same problem.

In Chapter 4 we investigated RWA for the case of scheduled lightpath demands. Two approaches were proposed. The first solves the problem by separately solving the routing and wavelength assignment subproblems. We developed a tabu search algorithm for the routing subproblem with a novel evaluation function and neighborhood reduction technique. Wavelength assignment was solved using an existing graph coloring algorithm. The second approach used to solve the scheduled RWA problem presented in this thesis is based on highly efficient greedy algorithms. Comparison with an existing algorithm for the same

problem shows the superiority of the suggested approaches. Furthermore, we developed a new lower bound on the number of wavelength required to perform scheduled RWA.

Multicast Routing and Wavelength Assignment was studied in Chapter 5. We first investigated the problem of multicast routing which can be reduced to the classical optimization problem of finding the minimum Steiner tree in a graph. This problem can be augmented to include additional constraints which represent QoS (*Quality of Service*) demands included in multicast requests. We developed a GRASP meta-heuristic algorithm to solve the multicast routing problem with the added constraint of a bounded end-to-end delay from the source node to all the destination nodes in a multicast session. This algorithm was tested on a benchmark set of problems and shown to outperform existing algorithms and provide near-optimal solutions. The rest of the chapter deals with simultaneous multicast routing and wavelength assignment. We developed greedy algorithms for multicast RWA based on bin packing which use the suggested GRASP multicast routing algorithm in intermediate steps. We developed lower bounds which indicate the efficiency of the proposed algorithms.

In the last chapter, the design of virtual topologies was studied. We investigated various objective criteria for this problem and proposed an additional criterion aimed at improving the connectivity of the virtual topology. It is also aimed at postponing the need for reconfiguration in order to minimize the cost associated with connection disruption. We developed a highly efficient lower bound for this criterion, and discussed lower bounds for other objective criteria. We proposed an approach to virtual topology design which uses novel rounding techniques upon solving the LP-relaxation of the problem. Furthermore, we developed highly efficient greedy algorithms whose mutual variations are aimed at optimizing various objective criteria. We conducted a detailed analysis of the advantages and drawbacks associated with the proposed approaches.

Future avenues of research include developing similar methods for solving problems in wavelength routed networks with sparse or full wavelength conversion. Optimal placement of wavelength converters in networks with sparse wavelength conversion is also an important problem to consider. In addition to wavelength converter placement, amplifier placement is critical in the cost effective design of long-haul optical transport networks. Considering transmission impairments, such as target BER (*Bit Error Rate*) levels, upon solving the RWA problem could help improve the performance of such networks. Another possible future venue is concerned with dynamic routing and wavelength assignment and the control and management issues associated with this problem to further utilize network resources.

Bibliography

[1] M. Ali and J. S. Deogun, "Cost effective implementation of multicasting in wavelength-routed networks," *Journal of Lightwave Technology*, vol. 18, no. 12, pp. 1628–1638, Dec. 2000.

[2] M. Ali and J. S. Deogun, "Power-efficient design of multicast wavelength routed networks," *IEEE J. Select. Areas Commun.*, vol. 18, pp. 1852–1862, Oct. 2000.

[3] A. C. F. Alvim, F. Glover, C. C. Ribeiro, and D. J. Aloise, "A hybrid improvement heuristic for the one-dimensional bin packing problem," *Journal of Heuristics*, vol. 10, pp. 205–229, 2004.

[4] D. Banerjee and B. Mukherjee, "A practical approach for routing and wavelength assignment in large wavelength-routed optical newtorks," *IEEE J. Select. Areas Commun.*, vol. 14, pp. 903–908, June 1996.

[5] D. Banerjee and B. Mukherjee, "Wavelength-routed optical networks: Linear formulation, resource budgeting tradeoffs, and a reconfiguration study," *IEEE/ACM Trans. Networking*, vol. 8, no. 5, pp. 598–607, 2000.

[6] D. Banerjee and B. Mukherjee, "Wavelength-routed optical networks: Linear formulation, resource budgeting tradeoffs, and a reconfiguration study," *IEEE/ACM Trans. Networking*, vol. 8, no. 5, pp. 598–607, Oct. 2000.

[7] R. L. Brooks, "On coloring the nodes of a network," in *Proc. Cambridge Phil. Soc.*, vol. 37, 1941, pp. 194–197.

[8] B. Chen and J. Wang, "Efficient routing and wavelength assignment for multicast in WDM networks," *IEEE J. Select. Areas Commun.*, vol. 20, no. 1, pp. 97–109, Jan. 2002.

[9] C. Chen and S. Banerjee, "A new model for optimal routing and wavelength assignment in wavelength division multiplexed optical networks," in *Proc. IEEE INFOCOM '96*, 1996, pp. 164–171.

[10] X. Cheng and D.-Z. Du, Eds., *Steiner Trees in Industry*. Boston: Kluwer Academic Publishers, 2001.

[11] I. Chlamtac, A. Ganz, and G. Karmi, "Lightpath communications: An approach to high-bandwidth optical WANs," *IEEE Trans. Commun.*, vol. 40, pp. 1171–1182, 1992.

[12] J. S. Choi, N. Golmie, F. Lapeyere, F. Mouveaux, and D. Su, "A functional classification of routing and wavelength assignment schemes in DWDM networks: Static case," in *Proc. VII Int. Conf. on Optical Communication and Networks*, Nagoya, Japan, Jan. 2000.

[13] X. Chu and B. Li, "Dynamic routing and wavelength assignment in the presence of wavelength conversion for all-optical networks," *IEEE/ACM Trans. Networking*, vol. 13, no. 3, pp. 704–715, June 2005.

[14] X. Chu, B. Li, and I. Chlamtac, "Wavelength converter placement under different RWA algorithms in wavelength-routed all-optical networks," *IEEE Trans. Commun.*, vol. 51, no. 4, pp. 607–617, Apr. 2003.

[15] X. Chu, B. Li, K. Sohraby, and Z. Zhang, "Routing and wavelength assignment issues in the presence of wavelength conversion for all-optical networks," in *Proc. of GLOBECOM 2002 - IEEE Global Telecommunications Conference*, vol. 21, Nov. 2002, pp. 2794–2798.

[16] X. Chu, B. Li, and Z. Zhang, "A dynamic RWA algorithm in a wavelength-routed all-optical network with wavelength converters," in *Proc. of IEEE INFOCOM 2003 - The Conference on Computer Communications*, vol. 22, no. 1, Mar. 2003, pp. 1795–1804.

[17] L. Cline, C. Maciocco, and M. Mishra, "Enabling architectures for next generation optical networks," in *Emerging Optical Network Technologies: Architecture, Protocols and Performance*, K. M. S. ans S. Subramaniam, Ed. New York, NY: Springer, 2005, ch. 1, pp. 3–22.

[18] E. G. Coffman, J. Csirik, and G. Woeginger, "Bin packing theory," in *Handbook of Applied Optimization*, P. Pardalos and M. G. C. Resende, Eds. New York: Oxford University Press, 2002.

[19] E. G. Coffman, M. R. Garey, and D. S. Johnson, "Bin packing approximation algorithms: A survey," in *Approximation Algorithms for NP-Hard Problems*, D. Hochbaum, Ed. Boston, MA: PWS Publishing Co., 1996.

[20] T. H. Cormen, C. E. Leiserson, and R. L. Revest, *Introduction to algorithms*. Cambridge: MIT Press, 1997.

[21] T. Deng, S. Subramaniam, and J. Xu, "Crosstalk-aware wavelength assignment in dynamic wavelength-routed optical networks," in *Proceedings of the First International Conference on Broadband Networks (BROADNETS'04)*, 2004.

[22] A. Ding and G.-S. Poo, "A survey of optical multicast over WDM networks," *Computer Communications*, vol. 26, no. 2, pp. 193–200, Feb. 2003.

[23] S. S. Dixit, Ed., *IP over WDM:Building the Next Generation Optical Internet*. Hoboken, NJ: John Wiley & Sons, Inc., 2003.

[24] R. Dutta and G. N. Rouskas, "A survey of virtual topolgy design algorithms for wavelength routed optical netorks," *Optical Netw. Mag.*, vol. 1, no. 1, pp. 73–89, Jan. 2000.

[25] D. Eppstein, "Finding the k shortest paths," *SIAM J. Computing*, vol. 28, no. 2, pp. 652–673, 1998.

[26] R. W. Floyd, "Algorithm 97: Shortest paths," *Comm. Of the AMC*, vol. 5, no. 6, p. 345, 1962.

[27] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: Freeman, 1979.

[28] A. Gençata and B. Mukherjee, "Virtual-topology adaptation for WDM mesh networks under dynamic traffic," *IEEE/ACM Trans. Networking*, vol. 11, no. 2, pp. 236–247, Apr. 2003.

[29] F. Glover and M. Laguna, *Tabu Search*. Boston, MA: Kluwer Academic Publishers, 1997.

[30] D. Griffith, "The GMPLS control plane architecture for optical networks," in *Emerging Optical Network Technologies: Architecture, Protocols and Performance*, K. M. S. ans S. Subramaniam, Ed. New York, NY: Springer, 2005, ch. 9, pp. 193–218.

[31] J. Gu, X.-D. Hu, X. Jia, and M.-H. Zhang, "Routing algorithm for multicast under multi-tree model in optical networks," *Theoretical Computer Science*, vol. 314, pp. 293–301, 2004.

[32] B. K. Haberman and G. Rouskas, "Cost, delay, and delay variation concious multicast routing," North Carolina State University," Technical Report TR/97/03, 1997.

[33] R. L. Hadas and R. Melhem, "Multicast routing and wavelength assignment in multihop optical networks," *IEEE/ACM Trans. Networking*, vol. 10, no. 5, pp. 621–629, Oct. 2002.

[34] X.-D. Hu, T.-P. Shaui, X. Jia, and M.-H. Zhang, "Multicast routing and wvalength assignment in WDM networks with limited drop-offs," in *Proc. IEEE INFOCOM*, Hong Kong, Mar. 2004.

[35] E. Hyytia and J. Virtamo, "Wavelength assignment and routing in WDM networks," Nordic Telegraffic Seminar 14, pp. 31–40, 1998.

[36] Common control and measurement plane. IETF Working Group. [Online]. Available: http://www.ietf.org/html.charters/ccamp-charter.html

[37] R. Inkret, A. Kuchar, and B. Mikac, *Advanced Infrastructure for Photonic Networks: Extended Final Report of COST Action 266*. Zagreb: Faculty of Electrical Engineering and Computing, University of Zagreb, 2003, pp. 19–21.

[38] M. Jain, "Topology design for wavelength routed optical networks," Master's thesis, Indian Institute of Science, Bangalore, Jan. 1996.

[39] X. Jia, X.-D. Hu, and D.-Z. Du, *Multiwavelength Optical Networks*. Norwell, MA: Kluwer Academic Publishers, 2002.

[40] X.-H. Jia, D.-Z. Du, X.-D. Hu, M.-K. Lee, and J. Gu, "Optimization of wavelength assignment for QoS multicast in WDM networks," *IEEE Trans. Commun.*, vol. 49, no. 2, pp. 341–350, Feb. 2001.

[41] D. S. Johnson and M. A. Trick, Eds., *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, ser. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, 1993, vol. 26.

[42] S. Kirckpatrick and E. Stoll, "A very fast shift-register sequence random number generator," *J. Computational Physics*, vol. 40, pp. 517–526, 1981.

[43] D. Kirovski and M. Potkonjak, "Efficient coloring of a large spectrum of graphs," in *Proc. 35th Conf. Design Automation*, San Francisco, CA, June 1998, pp. 427–432.

[44] J. M. Kleinberg, "Approximation algorithms for disjoint paths problems," Ph.D. dissertation, MIT, Cambridge, May 1996.

[45] T. Koch, A. Martin, and S. Voß. (2001) Steinlib: An updated library on steiner tree problems in graphs. [Online]. Available: http://elib.zib.de/steinlib

[46] V. P. Kompella, J. C. Pasquale, and G. C. Plyzos, "Multicast routing problems," *IEEE/ACM Trans. Networking*, vol. 1, pp. 286–292, 1993.

[47] M. Kovačević and A. S. Acampora, "Electronic wavelength translation in optical networks," *Journal of Lightwave Technology*, vol. 14, pp. 1161–1169, June 1996.

[48] R. M. Krishnaswamy and K. N. Sivarajan, "Design of logical topologies: a linear formulation for wavelength-routed optical networks with no wavelength changers," *IEEE/ACM Trans. Networking*, vol. 9, no. 2, pp. 186–198, Apr. 2001.

[49] J. Kuri, "Optimization problems in WDM optical transport networks with scheduled lightpath demands," Ph.D. dissertation, Ecole Nationale Supérieure des Télécommunications, France, Sept. 2003.

[50] J. Kuri, N. Puech, and M. Gagnaire, "A tabu search algorithm to solve a logical topology design problem in WDM networks considering implementations costs," in *Proc. of SPIE Asian Pacific Optical Conference*, Shangai, China, Oct. 2002.

[51] J. Kuri, N. Puech, and M. Gagnaire, "Diverse routing of scheduled lightpath demands in an optical transport network," in *Proc. Design of Reliable Communication Networks (DCRN2003)*, Oct. 2003, pp. 69–76.

[52] J. Kuri, N. Puech, M. Gagnaire, E. Dotaro, and R. Douville, "Routing and wavelength assignment of scheduled lightpath demands," *IEEE J. Select. Areas Commun.*, vol. 21, pp. 1231–1240, Oct. 2003.

[53] K. Lee, K. C. Kang, T. Lee, and S. Park, "An optimization approach to routing and wavelength assignment in WDM all-optical mesh networks without wavlength conversion," *ETRI Journal*, vol. 24, no. 2, pp. 131–141, 2002.

[54] G. Li and R. Simha, "The partition coloring problem and its application to wavelength routing and assignment," in *Proc. of Optical Networks Workshop*, Richardson, Texas, 2000.

[55] E. Mannie, "Generalized multi-protocol label switching (GMPLS) architecture," RFC 3945, Oct. 2004.

[56] P. Manohar, D. Manjunath, and R. K. Shevgaonkar, "Effect of objective function on performance in wavelength routed optical networks," in *Proc of Eighth National Communications Conference (NCC-2002)*, Mumbai, Jan. 2002.

[57] P. Manohar, D. Manjunath, and R. K. Shevgaonkar, "Routing and wavelength assignment in optical networks from edge disjoint paths algorithms," *IEEE Commun. Lett.*, vol. 6, pp. 211–213, May 2002.

[58] S. L. Martins, P. M. Pardalos, M. G. C. Resende, and C. C. Ribeiro, "Greedy randomized adaptive search procedures for the Steiner problem in graphs," in *Randomization Methods in Algorithmic Design*, ser. Vol. 43 ofDIMACS Series on Discrete Mathematics and Theoretical Computer Science, P. M. Pardalos, S. Rajasejaran, and J. Rolim, Eds. American Mathematical Society, 1999.

[59] S. L. Martins, M. G. C. Resende, C. C. Ribeiro, and P. M. Pardalos, "A parallel GRASP for the steiner tree problem in graphs using a hybrid local search strategy," *Journal of Global Optimization*, vol. 17, pp. 267–283, 2000.

[60] E. Modiano and A. Narula-Tam, "Survivable lightpath routing: A new approach to the design of WDM-based networks," *IEEE J. Select. Areas Commun.*, vol. 20, no. 4, pp. 800–809, May 2002.

[61] H. T. Mouftah and P.-H. Ho, *Optical Networks: Architecture and Survivability*. Norwell, MA: Kluwer Academic Publishers, 2003.

[62] B. Mukherjee, "WDM-based local lightwave networks - Part I: Single-hop systems," *IEEE Network Mag.*, vol. 6, pp. 12–27, May 1992.

[63] B. Mukherjee, "WDM-based local lightwave networks - Part II: Multihop systems," *IEEE Network Mag.*, vol. 6, pp. 20–32, July 1992.

[64] B. Mukherjee, *Optical Communication Networks*. New York: McGraw-Hill, 1997.

[65] B. Mukherjee, “WDM optical communication networks: Progress and challenges,” *IEEE J. Select. Areas Commun.*, vol. 18, no. 10, pp. 1810–1824, Oct. 2000.

[66] B. Mukherjee, D. Banerjee, S. Ramamurthy, and A. Mukherjee, “Some principles for designing a wide-area WDM optical network,” *IEEE/ACM Trans. Networking*, vol. 4, no. 5, pp. 684–696, Oct. 1996.

[67] C. S. R. Murthy and M. Gurusamy, *WDM Optical Networks: Concepts, Design, and Algorithms*. New Jersey: Prentice Hall, 2002.

[68] J. Nittayawan and S. Runggeratigul, “Optimum regular logical topology for wavelength routed WDM networks,” *IEICE Trans. Commun.*, vol. E87-B, no. 12, Dec. 2004.

[69] T. F. Noronha and C. C. Ribeiro, “Routing and wavelength assignment by partition coloring,” *European Journal of Operational Research*, 2004, (Accepted for publication).

[70] C. A. S. Oliveira, P. M. Pardalos, and M. G. C. Resende, “Optimization problems in multicast tree construction,” in *Handbook of Optimization in Telecommunications*. Dordrecht: Kluwer, 2005, pp. 701–733.

[71] A. Ozdaglar and D. Bertsekas, “Routing and wavelength assignment in optical networks,” *IEEE/ACM Trans. Networking*, vol. 11, pp. 259–272, Apr. 2003.

[72] R. K. Pankaj, “Wavelength requirements for multicasting in all-optical networks,” *IEEE/ACM Trans. Networking*, vol. 7, no. 3, pp. 414–424, June 1999.

[73] N. Puech, J. Kuri, and M. Gagnaire, “Topological design and lightpath routing in WDM mesh networks: A combined approach,” *Photonic Network Communications*, vol. 4, no. 3/4, pp. 443–456, July 2002.

[74] B. Ramamurthy and B. Mukherjee, “Wavelength conversion in WDM networking,” *IEEE J. Select. Areas Commun.*, vol. 16, no. 7, pp. 1061–1073, Sept. 1998.

[75] B. Ramamurthy and A. Ramakrishnan, “Virtual topology reconfiguration of wavelength-routed optical WDM networks,” in *Proc. of GLOBECOM 2000 - IEEE Global Telecommunications Conference*, vol. 1, Nov. 2000, pp. 1269–1275.

[76] R. Ramaswami and K. N. Sivarajan, "Design of logical topologies for wavelength-routed optical networks," *IEEE J. Select. Areas Commun.*, vol. 14, no. 5, pp. 840–851, June 1996.

[77] M. G. C. Resende and C. C. Ribeiro, "Greedy randomized search procedures," in *Handbook of Metaheuristics*, F. Glover and G. Kochenberger, Eds. Kluwer Academic Publishers, 2003.

[78] C. C. Ribeiro, E. Uchoa, and R. F. Werneck, "A hybrid GRASP with perturbations for the Steiner problem in graphs," *INFORMS Journal on Computing*, vol. 14, pp. 228–246, 2002.

[79] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," RFC 3031, Jan. 2001.

[80] G. N. Rouskas and L. Xu, "Optical packet switching," in *Emerging Optical Network Technologies: Architecture, Protocols and Performance*, K. M. S. ans S. Subramaniam, Ed. New York, NY: Springer, 2005, ch. 5, pp. 111–127.

[81] L. H. Sahasrabuddhe and B. Mukherjee, "Light-trees: Optical multicasting for improved performance in wavelength-routed networks," *IEEE Commun. Mag.*, vol. 37, pp. 67–73, Feb. 1999.

[82] C. V. Saradhi, L. K. Wei, and M. Guruswami, "Provisioning fault-tolerant scheduled lightpath demands in WDM mesh networks," in *Proc. Broadband Networks*, Oct. 2004, pp. 150–159.

[83] A. R. Sharafat, "The most congested cutset: Deriving a tight lower bound for the chromatic number in the RWA problem," *IEEE Communication Letters*, vol. 8, no. 7, pp. 473 – 475, 2004.

[84] N. K. Singhal and B. Mukherjee, "Protecting multicast sessions in WDM optical mesh networks," *Journal of Lightwave Technology*, vol. 21, no. 4, pp. 884–892, Apr. 2003.

[85] N. Skorin-Kapov, "Heuristic algorithms for the routing and wavelength assignment of scheduled lightpath demands in optical networks," *IEEE J. Select. Areas Commun.*, (Accepted for publication).

[86] N. Skorin-Kapov, "Routing and wavelength assignment in optical networks using bin packing based algorithms," *Europ. J. Op. Res.*, (Accepted for publication).

[87] N. Skorin-Kapov and M. Kos, "The application of Steiner trees to delay-constrained multicast routing: A tabu search approach," in *Proc. Of Contel2003 - Conference on Telecommunications*, Zagreb, Croatia, 2003.

[88] N. Skorin-Kapov and M. Kos, "Heuristic algorithms considering various objectives for virtual topology design in WDM optical networks," in *In Proc. of the 2005 International Conference on Telecommunication Systems: Modelling and Analysis ICTSMA2005*, Dallas, Texas, USA, Nov. 2005, pp. 195–209.

[89] N. Skorin-Kapov and M. Kos, "A GRASP heuristic for the delay-constrained multicast routing problem," *Telecommunication Systems*, (Accepted for publication).

[90] L. Stacho, "New upper bounds for the chromatic number of a graph," *J. Graph Theory*, vol. 36, no. 2, pp. 117–120, Feb. 2001.

[91] T. E. Stern and K. Bala, *Multiwavelength Optical Networks: A Layered Approach*. Upper Saddle River, NJ: Prentice Hall, Inc., 2000.

[92] S. Subramaniam, M. Azizoglu, and A. K. Somani, "On optimal converter placement in wavelength-routed networks," *IEEE/ACM Trans. Networking*, vol. 7, no. 5, pp. 754–766, Oct. 1999.

[93] A. Tervonen, "Optical enabling technologies for WDM systems," in *IP over WDM:Building the Next Generation Optical Internet*, S. S. Dixit, Ed. Hoboken, NJ: John Wiley & Sons, Inc., 2003, ch. 3, pp. 59–100.

[94] C. Xin, S. S. Dixit, and C. Qiao, "IP-over-WDM control and signaling," in *IP over WDM:Building the Next Generation Optical Internet*, S. S. Dixit, Ed. Hoboken, NJ: John Wiley & Sons, Inc., 2003, ch. 14, pp. 421–437.

[95] C. Xin, Y. Ye, S. S. Dixit, and C. Qiao, "An emulation-based comparative study of centralized and distributed control schemes for optical networks," in *Proc. of SPIE IPCom 2001*, vol. 4527, Denver, Aug. 2001, pp. 65–72.

[96] S. Yao, B. Mukherjee, and S. S. Dixit, "Advances toward optical subwavelength switching," in *IP over WDM:Building the Next Generation Optical Internet*, S. S. Dixit, Ed. Hoboken, NJ: John Wiley & Sons, Inc., 2003, ch. 6, pp. 157–180.

[97] M. Yoo and Y. Kim, "Internetworking optical internet and optical burst switching," in *IP over WDM:Building the Next Generation Optical Internet*, S. S. Dixit, Ed. Hoboken, NJ: John Wiley & Sons, Inc., 2003, ch. 13, pp. 397–420.

[98] J. Y. Youe and S.-W. Seo, "An algorithm for virtual topology design in WDM optical networks under physical constraints," in *Proc. of ICC 1999 - IEEE International Conference on Communications*, vol. 1, 1999, pp. 1719–1723.

[99] H. Zang, *WDM Mesh Networks: Management and Survivability*. Norwell, MA: Kluwer Academic Publishers, 2003.

[100] Z. Zang and A. S. Acampora, "A heuristic wavelength assignment algorithm for multihop WDM networks with wavelength routing and wavelength re-use," *IEEE/ACM Trans. Networking*, vol. 3, no. 3, pp. 281–288, June 1995.

[101] J. Zhang and B. Mukherjee, "A review of fault management in WDM mesh networks: Basic concepts and research challenges," *IEEE Network*, vol. 18, no. 2, pp. 41–48, Mar. 2004.

[102] Q. Zhang and Y. W. Leung, "An orthogonal genetic algorithm for multimedia multicast routing," *IEEE Trans. on Evolutionary Computation*, vol. 3, no. 1, pp. 53–61, 1999.

[103] X. Zhang, J. Y. Wei, and C. Qiao, "Constrained multicast routing in WDM networks with sparse light splitting," *Journal of Lightwave Technology*, vol. 18, no. 12, pp. 1917–1927, Dec. 2000.

[104] L. Zhong and B. Ramamurthy, "Optimization of amplifier placements in switch-based optical networks," in *Proc. of ICC 2001 - IEEE International Conference on Communications*, vol. 1, June 2001, pp. 224–228.

[105] X. Zhou, C. Chen, and G. Zhu, "A genetic algorithm for multicasting routing problem," in *Proceedings of International Conference on Communication Technologies (ICCT2000)*, Beijing, 2000.

[106] K. Zhu and B. Mukherjee, "Traffic grooming in a WDM mesh network," *IEEE J. Select. Areas Commun.*, vol. 20, no. 1, pp. 122–133, Jan. 2002.

[107] Q. Zhu, M. Parsa, and J. J. Garcia-Luna-Aceves, "A source based algorithm for delay-constrained minimum-cost multicasting," in *Proceedings of IEEE INFOCOM*, Boston, MA, 1995.

Summary

This thesis investigates the problem of designing virtual topologies in wavelength routed WDM (Wavelength Division Multiplex) optical networks. In such networks, a virtual topology is created over the physical optical network by establishing all-optical connections, called lightpaths, between pairs of nodes. Transport via a lightpath is entirely in the optical domain. A virtual topology can also be composed of a set of light-trees which optically connect a subset of nodes in the network. In order to establish a virtual topology, it is necessary to determine a set of lightpaths/light-trees, find for them corresponding paths in the physical topology and assign wavelengths to them. Finally, packet-switched traffic is routed over the virtual topology. The thesis focuses on the problem of routing and assigning wavelengths to lightpaths and light-trees, and the virtual topology design problem in WDM networks. These problems are NP-complete so heuristic algorithms are needed to help solve them.

Proposed are efficient heuristic algorithms for the Routing and Wavelength Assignment (RWA) of static and scheduled lightpath demands. Furthermore, developed is a heuristic for multicast routing and algorithms for static multicast RWA. Virtual Topology Design is investigated considering various objective criteria, and an additional objective criterion is proposed. Efficient heuristic algorithms are developed to help solve this problem. To assess the quality of the solutions obtained by the proposed algorithms, new analytical lower bounds for the corresponding problems are developed.

Keywords

WDM, wavelength routed optical networks, routing and wavelength assignment, virtual topology design, heuristic algorithms, lightpaths/light-trees, bin packing, optical multicasting, tabu search, GRASP

Curriculum Vitae

Nina Skorin-Kapov was born in Zagreb, Croatia on January 14th, 1981. She grew up in Vancouver, Canada and Long Island, NY, USA. In 1996, she moved back to Croatia, finished high school and enrolled in the undergraduate program at the University of Zagreb, Faculty of Electrical Engineering and Computing in the area of Telecommunications and Informatics. In 2003, she completed a nine semester program with an emphasis on scientific research and a diploma thesis entitled "The Application of Steiner Trees to Delay Constrained Multicast Routing". In 2003, she enrolled in the graduate program at the University of Zagreb, Faculty of Electrical Engineering and Computing, Croatia, in the field of Telecommunications. She has been working as a research assistant at the same university since 2003.

Nina Skorin-Kapov is the author of 4 refereed journal articles and 3 articles in refereed proceedings. She investigates heuristic algorithms for optimization problems in telecommunications. Her specific field of research deals with optimization in wide-area WDM optical networks, network topology design, and network routing algorithms. She is a member of the IEEE Communications Society and The Croatian Society of Operations Research.