Department of Electrical and Computer Engineering

Heuristic Multicast Routing and Protection Algorithms for Optical WDM Networks with Arbitrary Mesh Topologies

Costas K. Constantinou

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

at the University of Cyprus

December, 2012

APPROVAL PAGE

Costas K. Constantinou

Heuristic Multicast Routing and Protection Algorithms for Optical WDM Networks with Arbitrary Mesh Topologies

The present Doctorate Dissertation was submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in the Department of Electrical and Computer Engineering, and was approved on December 19, 2012 by the members of the Examination Committee.

Committee Chair _____
Dr. Christos Panayiotou

Research Supervisor _____
Dr. Georgios Ellinas

Committee Member _____
Dr. Theocharis Theocharides

Committee Member _____
Dr. Vasos Vassiliou

Committee Member _____
Dr. Ioannis Sahalos

Abstract

Over the last few decades, the size and complexity of telecommunications networks have steadily increased and this trend will continue for the foreseeable future. Fiber-optic communication networks that provide a huge available amount of capacity and low bit-error rates are currently widely used as the telecommunication medium of choice that is able to supply high-speed and reliable communications. Optical networks, where the provisioning and fault recovery functionalities are dealt with at the physical layer, have been at the forefront of research for several years, especially for unicast applications, and these networks are currently being realized today mostly in the backbone arena.

However, new traffic requirements are currently emerging for these new types of network architectures, including new high-bandwidth applications (such as video-on-demand, teleconferencing, distance-learning, remote medical diagnostic applications, etc.) that require point-and-click provisioning of multicast sessions in the physical domain.

Furthermore, due to the very large amount of information carried on the optical fiber links, the fact that for multicast applications a single link failure may potentially disrupt traffic to a large number of destinations, and the availability service requirements offered by the service providers to the customers (and quantified via service level agreements (SLAs)), fast recovery from failures is also an essential part of network operation in backbone mesh optical networks.

This thesis addresses precisely the problem of multicast routing and fault recovery in wavelength-division multiplexed (WDM) optical networks with arbitrary mesh topologies. The main contribution of the thesis is in the design of novel multicast routing and protection heuristic algorithms for undirected- and mixed-graph networks. An undirected graph is defined as one having only bidirectional connections between its nodes, and a mixed graph as one having both bidirectional and

unidirectional connections. The modeling of a network as a mixed graph is closer to practical networks, since even if the network is designed to be undirected, the already established requests hold resources of it, resulting in a mixed-graph network for the subsequent requests. For both the undirected and mixed-graph networks, new multicast routing and protection techniques are developed, and it is shown through examples and simulations on randomly created networks that they demonstrate enhanced network performance compared to the currently most widely used relevant techniques in mesh optical networks.

Various network architectures were investigated in this work including networks with full- and sparse-splitting capabilities, and networks with drop-and-continue and drop-or-continue nodes. For these types of networks, heuristic algorithms were developed that give efficient solutions to the problems of splitter allocation in the network and to the problem of multicast routing.

Furthermore, a load balancing technique is presented, that takes into account the already established multicast requests and the congestion they cause to each link, in order to route the upcoming requests more efficiently.

While all the aforementioned heuristics were developed in order to deal precisely with the problem of multicast routing and protection in mesh optical networks, it must be noted that most of them are general and they can be used in other types of networks as well, with minor modifications.

Περίληψη

Κατά τα τελευταία χρόνια, το μέγεθος και η πολυπλοκότητα των τηλεπικοινωνιακών δικτύων αυξάνονται σταθερά και η τάση αυτή αναμένεται να συνεχιστεί και στα επόμενα χρόνια. Τα τηλεπικοινωνιακά συστήματα οπτικών ινών τα οποία παρέχουν τεράστια χωρητικότητα και χαμηλά ποσοστά σφάλματος, χρησιμοποιούνται ευρέως στα σημερινά τηλεπικοινωνιακά δίκτυα και παρέχουν αξιόπιστες επικοινωνίες υψηλής ταχύτητας. Τα αμιγώς οπτικά δίκτυα, όπου το σήμα παραμένει συνεχώς στο οπτικό επίπεδο, και οι λειτουργίες δρομολόγησης και αποκατάστασης βλαβών εκτελούνται στο φυσικό στρώμα, είναι στην πρώτη γραμμή της έρευνας τα τελευταία χρόνια, ειδικά για εφαρμογές μονοσημειακής σύνδεσης.

Ωστόσο, υπάρχουν αρκετές νέες εφαρμογές (όπως βίντεο κατά απαίτηση, τηλεδιάσκεψη, εξ' αποστάσεως μάθηση, εξ' αποστάσεως εφαρμογές ιατρικής διάγνωσης, κλπ), οι οποίες απαιτούν πολυσημειακή σύνδεση στο φυσικό πεδίο.

Επιπλέον, λόγω του ότι οι οπτικές ίνες μεταφέρουν πολύ μεγάλο όγκο πληροφορίας, το γεγονός ότι στις εφαρμογές πολυσημειακής σύνδεσης η βλάβη μιας μόνο οπτικής ίνας μπορεί να επηρεάσει τη μετάδοση της πληροφορίας σε μεγάλο αριθμό προορισμών, και το επίπεδο ποιότητας υπηρεσίας που προσφέρεται από τους παροχείς στους χρήστες, οδηγούν στην αναγκαιότητα ενσωμάτωσης μηχανισμών γρήγορης αποκατάστασης βλαβών στη λειτουργία των οπτικών δικτύων.

Η παρούσα διατριβή ασχολείται ακριβώς με το πρόβλημα της δρομολόγησης και αποκατάστασης βλαβών σε εφαρμογές πολυσημειακής σύνδεσης, σε οπτικά δίκτυα με πολυπλεξία μήκους κύματος και αυθαίρετη τοπολογία πλέγματος. Η κύρια συνεισφορά της διατριβής είναι η ανάπτυξη καινοτόμων αλγορίθμων δρομολόγησης και προστασίας

πολυσημειακών συνδέσεων, για δίκτυα μη κατευθυνόμενου και μικτού γραφήματος. Ένα μη κατευθυνόμενο γράφημα ορίζεται ως εκείνο που έχει μόνο αμφίδρομες συνδέσεις μεταξύ των κόμβων του, και ένα μικτό γράφημα ως εκείνο που έχει τόσο αμφίδρομες όσο και μονόδρομες συνδέσεις. Η μοντελοποίηση ενός δικτύου ως ένα μικτό γράφημα είναι πιο κοντά στην πραγματικότητα, δεδομένου ότι στην πράξη, ακόμα και αν το δίκτυο έχει σχεδιαστεί ως μη κατευθυνόμενο, οι συνδέσεις που έχουν ήδη υλοποιηθεί δεσμεύουν κάποιο μέρος της χωρητικότητας του, με αποτέλεσμα η εναπομένουσα χωρητικότητα να οδηγεί σε δίκτυο μικτού γραφήματος για τις επικείμενες συνδέσεις.

Διάφορες αρχιτεκτονικές δικτύων ερευνήθηκαν στη παρούσα εργασία, όπως δίκτυα με οπτικούς διαχωριστές σε όλους τους κόμβους, δίκτυα με οπτικούς διαχωριστές σε μερικούς από τους κόμβους, και δίκτυα με κόμβους τερματισμού-και-συνέχειας και τερματισμού-ή-συνέχειας. Για αυτές τις κατηγορίες δικτύων, αναπτύχθηκαν αλγόριθμοι που δίνουν αποτελεσματικές λύσεις στα προβλήματα της κατανομής των οπτικών διαχωριστών στο δίκτυο και της δρομολόγησης πολυσημειακών συνδέσεων.

Επιπλέον, παρουσιάζεται μια τεχνική εξισορρόπησης φορτίου, η οποία λαμβάνει υπόψη τις ήδη εγκατεστημένες εντολές πολυσημειακών συνδέσεων και τη συμφόρηση που προκαλούν σε κάθε σύνδεση του δικτύου, ώστε η δρομολόγηση των επερχόμενων συνδέσεων να γίνει με πιο αποτελεσματικό τρόπο.

Παρά το γεγονός ότι οι προτεινόμενοι αλγόριθμοι αναπτύχθηκαν για την περίπτωση των αδόμητων οπτικών δικτύων, οι περισσότεροι από αυτούς, με ελάχιστες τροποποιήσεις, μπορούν να εφαρμοστούν και σε άλλους τύπους δικτύων.

Όλοι οι προτεινόμενοι αλγόριθμοι αξιολογήθηκαν και συγκρίθηκαν με τους σχετικούς υφιστάμενους. Η βελτιωμένη τους απόδοση φαίνεται μέσα από παραδείγματα και προσομοιώσεις σε πραγματικά και τυχαία δημιουργημένα δίκτυα.

Acknowledgements

My Ph.D. thesis is dedicated to my parents, my sister and my grandparents, for their love and support, as well as to my advisor George Ellinas, for his guidance.

Ευχαριστίες

Η διδακτορική μου διατριβή αφιερώνεται στους γονείς μου, την αδερφή μου, τον παππού και τη γιαγιά μου, για την αγάπη και τη στήριξη τους, καθώς και στον καθηγητή μου Γεώργιο Έλληνα, για την καθοδήγηση του.

Contents

List of Figures

Acronyms

ADT: Arc-Disjoint Trees

ANT: Auxiliary Network Transformation

ASSP: Adaptive Shared Segment Protection

ATM: Asynchronous Transfer Mode

BP: Blocking Probability

BSV: Basic Steiner Vertex

CR: Cost Reduction

DaC: Drop-and-Continue

DNB: Decreased Number of Branches

DNCD: Decreased Number of Children Destinations

DoC: Drop-or-Continue

FDM: Frequency Division Multiplexing

GMPLS: Generalized Multiprotocol Label Switching

GSNH: Generalized Steiner Node Heuristic

HDTV: High Definition TV

ILP: Integer Linear Programming

IP: Internet Protocol

ITU-T: International Telecommunications Union - Telecommunications

kmaxD: k-maximum Degree

kmaxWR: k-maximum Wavelength Reduction

LB-OPP-SDP: Load Balancing Optimal Path-Pair Shared Disjoint Paths

LDT: Link-Disjoint Tree

LURF: Least-Used Removed First

MC: Multicast Capable

MC-RWA: Multicast Routing and Wavelength Assignment

MEMS: Micro-Electro-Mechanical Systems

MG-MPH: Mixed Graph Minimum Path Heuristic

MG-OPP-SDP: Mixed Graph Optimal Path-Pair Shared Disjoint Paths

MG-PPH: Mixed Graph Pruned Prim Heuristic

MG-SNH: Mixed Graph Steiner Node Heuristic

MGST: Mixed Graph Spanning Tree

MI: Multicast Incapable

MPF: Maximum Possible Flow

MPH: Minimum Path Heuristic

MPLS: Multiprotocol Label Switching

MSNF: Most Saturated Node First

MST: Minimum Spanning Tree

MUS: Multicasting Using Splitters

NDT: Node-Disjoint Trees

NMCF: Nearest Multicast-Capable node First

OADM: Optical Add Drop Multiplexer

OEO: Optical-Electrical-Optical

OOO: Optical-Optical-Optical

OPP-SDP: Optimal Path-Pair Shared Disjoint Paths

OPT: OPtimal Tree

OSV: Optional Steiner Vertex

OTMCF: On Tree Multicast-Capable node First

OTN: Optical Transport Network

OXC: Optical Cross-Connect

PCLH: Pruned Chu Liu Heuristic

PPH: Pruned Prim Heuristic

QoS: Quality-of-Service

RSV: Required Steiner Vertex

RWA: Routing and Wavelength Assignment

SaD: Splitter-and-delivery

SLA: Service Level Agreement

SMT: Steiner Minimal Tree

SNH: Steiner Node Heuristic

SONET: Synchronous Optical Networking

SR: Steiner Ratio

SSMRH: Sparse-Splitting Multicast Routing Heuristic

STP: Steiner Tree Problem

TDM: Time Division Multiplexing

TE: Traffic Engineering

WDM: Wavelength Division Multiplexing

WR: Wavelength Reduction

Chapter 1

Introduction

1.1 Background

Over the last few decades, the size and complexity of telecommunications networks have steadily increased. The number of Internet users as well as the average usage time per person are increasing every year (it is estimated that more than two billion people are nowadays Internet users [1]), and this trend will continue for the foreseeable future. Therefore, the usage of a telecommunications medium with huge available amount of capacity and low bit-error rate, that is able to supply high-speed and reliable communications, is a necessity. Optical fiber was the successor to copper-wire in telecommunication transmission systems, since it has all the aforementioned characteristics [2].

Both optical fibers and copper cables are *waveguides* that guide the signal and keep it focused for a reasonable distance without being scattered [3]. The signal arrives at the receiver with enough power so that the message can be successfully decoded. The main physical difference between these two communication mediums is that optical fiber transports *photons*, while in copper cables *electrons* are the carriers of information. Photons have the ability to travel faster and they do not affect one another, since they are not electrically charged [4]. Moreover, the amount of data that can be transmitted through an optical fiber is much greater compared to copper-based systems.

Networks that use optical fibers as the communication medium are called *optical networks*. This kind of networks combine several advantages such as: they offer much higher bandwidth than copper cables, they have low signal attenuation and

low signal distortion, they have very low bit-error rates, they have small space requirements, and they are less susceptible to electromagnetic interference [3]. Optical networks are separated in two categories, in terms of the multiplexing technique they use: *Time Division Multiplexing (TDM)* and *Wavelength Division Multiplexing (WDM)* optical networks [3]. In TDM networks, many lower speed data streams are multiplexed into a higher speed stream at the transmission bit rate. The multiplexer typically interleaves the lower speed streams to obtain the higher speed stream. In WDM networks, the idea is basically the same as frequency division multiplexing (FDM), which has been used in radio systems. Different data are transmitted simultaneously using different wavelengths, that do not interfere. This thesis deals with the latter case, since it is the most widely-used case for optical network deployment.

Fiber-optic communication systems utilizing WDM nowadays comprise of 40-channel or 80-channel 10 Gbps optical communications systems while under development are next-generation optical communications systems with single-channel data rates of 40 and 100 Gbps for long-haul. With the successful commercialization of WDM, the optical transport network (OTN) has been standardized by the ITU-T as the underlying infrastructure of the network, thus allowing the network to transport huge amounts of data that are needed for many current and future communication services and applications, including a host of bandwidth-intensive multicast applications.

In initial optical network deployments, while the transmission medium was optical (fiber-optic links), the rest of the network control and management functionalities took place at higher (electrical) layers such as the SONET, ATM, and IP/MPLS layers. Only recently have these control and management functionalities started to move down to the optical layer, and this was a direct consequence of the development of intelligent optical cross-connects (OXCs) that can support such functionalities. OXCs have functions similar to optical add/drop multiplexers (OADMs) (an optical add/drop multiplexers drops and adds a selective number of wavelengths from a WDM signal, while allowing the remaining wavelengths to pass through), but on a much larger scale in terms of the number of ports and wavelengths involved. OADMs are typically deployed in linear or ring topologies, while OXCs are usually used in mesh networks. These cross-connects that can switch any wavelength from any input port to any output port belong to two categories: opaque (OEO) cross-connects where switching takes place utilizing an electronic switch fabric and

all-optical (OOO) cross-connects where switching is done exclusively in the optical domain (e.g., utilizing a MEMS-based optical switch).

In the OEO case the main advantage is the fact that it allows access to the electrical signal and thus to the overhead bytes, making it very easy to provide all of the network control and management functionality at the optical layer. Specifically, in this case, automated network database creation and maintenance, automated provisioning, fault detection, isolation, and recovery, multiplexing and grooming, as well as unrestricted routing through the network can be very easily provided.

On the other hand, the OOO approach will provide for significant cost savings (achieved by removing all OEO conversions) and transparency in terms of bit rate, protocol format, and modulation format. However, OOO solutions, where the signal stays in the optical domain throughout, make it very difficult (but not impossible) to provide the necessary control and management functionalities, as now there is no access to the overhead bytes until the signal terminates at the destination nodes. Other challenges that also arise in completely transparent (OOO) networks include the accumulation of physical impairments along the entire path, the requirement for end-to-end system engineering, and the fact that fault recovery is practically limited to dedicated protection. All of these challenges can be tackled but with difficulty in transparent optical networks. In this thesis the assumption is that the cross-connect switches can be either opaque or all-optical.

Current and next-generation optical networks not only provide transmission capacities to higher transport levels, but now have the intelligence to support a number of real-time control functionalities previously provided only by the higher (electrical) layers such as IP/MPLS. Thus, control functions such as connection provisioning and fault recovery can now be handled by the optical layer rather by higher (electrical) layers. The emergence of intelligent optical network elements (including optical cross-connects) are instrumental in making such optical architectures a reality today. For example, the Generalized Multiprotocol Label Switching (GMPLS) control protocol was proposed to provide the control functions at the physical layer. GMPLS is a generalization of the Multiprotocol Label Switching (MPLS) protocol, which was designed to provide automated provisioning, connection maintenance, and network resource management, including providing Quality-of-Service (QoS) and Traffic Engineering (TE) for IP/MPLS networks.

Therefore, there has been a steady evolution of optical networks from point-to-

point WDM transmission systems, to ring, and finally to arbitrary connected mesh networks that utilize intelligent OXCs along with control protocols at the physical (optical) layer, in order to support network functionalities such as automated provisioning of optical services and automated fault recovery capabilities. The applications that are supported in these networks range from mainstream point-to-point voice and data services to frontier services such as HDTV, webcasting, and streaming video, to future services such as web agents, metacomputing, 3-D multimedia, telemedicine, etc. Clearly, frontier and future services include not only unicast applications but a host of multicast applications as well. The provisioning and fault recovery of these applications is precisely the topic of this dissertation.

1.2 Motivation

The majority of the traffic carried over an optical mesh network currently corresponds to unicast connections, mostly in the form of point-to-point connections for enterprise customers. However, a number of new customer applications have emerged that are driving the need to support multicast connections, potentially over optical mesh networks. These applications include video distribution for residential customers, video conferencing between telepresence-equipped rooms, video training, e-learning and on-line teaching, emerging grid-computing applications requiring collaborative and interactive high-definition visualization, and telemedicine applications amongst others.

Multicasting in optical networks is achieved by the calculation of light-trees. Since this is equivalent to the Steiner tree problem in graphs, Steiner tree heuristic algorithms are exploited for finding these trees. The calculation of trees of low cost (i.e., consisting of a small number of arcs) is important, since this leads to the minimization of the network resources used. The heuristics that are widely used for multicasting in optical and other networks, are *Minimum Path Heuristic* (MPH) and *Pruned Prim Heuristic* (PPH). If new, more efficient heuristic algorithms are created, the network multicast calls will be established using less network resources, and this will increase the number of the calls the network can serve, thus reducing the blocking probability for the multicast connections to be provisioned.

Furthermore, all the multicast routing heuristic algorithms used in optical and other networks are used under the assumption that the network can be modeled as

an *undirected* graph (a graph where all the connections are bidirectional). However, this is not the practical case. Modeling a network as a *mixed* graph (a graph that consists of both unidirectional and bidirectional connections) is more realistic, as even if the network is designed to be undirected, when some requests arrive and hold resources of the network, some of the arcs have all their wavelengths occupied, and cannot be used for the upcoming requests. Therefore, the resulting network graph is mixed, and the routing for the new requests will be calculated on this mixed graph. Another case where the routing is performed on a mixed graph, is when the challenge is to find a pair of arc-disjoint trees (e.g., for protection against single link failures). In this case, the second tree, after the removal of the first one, will be calculated on a mixed graph. Therefore, new multicast routing heuristic algorithms must be created specifically for mixed-graph networks.

For the realization of light-trees, the node architectures must also be multicast-capable. Since it may not be possible to have multicast-capable architectures at each network node and assuming that only a fraction of the network nodes will be multicast-capable (resulting in *sparse-splitting* networks), efficient multicast routing heuristic algorithms must be created for this category of networks as well, including techniques on the placement of the multicast-capable nodes.

Protection of multicast connections is also a very important aspect for the operation of intelligent optical networks. Optical network architectures can suffer failures, either due to accidental fiber cuts and human mistakes, or due to equipment failures. As any failure in these networks can potentially result in a large amount of information loss if it is not recovered quickly, efficient recovery techniques are extremely important when provisioning the multicast requests. Therefore, it is important to devise protection methods that can find working and protection trees with less cost, that will subsequently lead to the saving of network resources and will increase the possible number of multicast requests that the network can serve. Furthermore, all existing multicast protection techniques ignore the current network status, for the calculation of the light-trees of the upcoming requests. New techniques are required that take into account the load distribution on the network, and the congestion each network link has, which will in turn lead to lower blocking probability, since such techniques will decrease the possibility that the network will be disconnected due to high congestion on specific links.

Clearly, there is fertile ground for a number of research innovations in the area

of multicast routing and protection in mesh optical networks and this is exactly the work performed in this dissertation as explained in detail in the section that follows.

1.3 Thesis Objective/Contribution

The objective of this thesis is to study problems related to multicast routing and protection in optical WDM networks with arbitrary mesh topologies. The main contributions of this dissertation are in the design and implementation of novel multicast provisioning and protection heuristic algorithms for different types of networks including unidirectional and mixed-graph networks, as well as networks with sparse-splitting capabilities. A number of heuristic algorithms that outperform the existing ones are presented, explained, and evaluated through examples and simulations.

Specifically, this dissertation initially develops a new multicast routing heuristic algorithm, called Steiner Node Heuristic (SNH), that is a general algorithm, i.e., it can use any appropriate algorithm as its basis and follow a recursive procedure to improve the solution obtained by this algorithm. The proposed method outperforms the widely-used multicast routing heuristic algorithms for mesh optical networks. The tradeoff for the performance gain achieved by the proposed algorithm is an acceptable increase of its time-complexity (the increase of the time-complexity is considered acceptable since it remains polynomial).

For sparse-splitting networks (i.e., networks where not all nodes are multicast capable (MC)) both problems of multicast routing and multicast-capable node allocation are investigated. A second contribution of this dissertation is the development of heuristic algorithms for multicast routing and MC-node placement that outperform the best of the existing ones. The multicast routing heuristic algorithms proposed in the current thesis locate the multicast-capable nodes, that, if used for the calculation of the multicast tree, will reduce its cost. Furthermore, new splitter allocation techniques are presented that outperform the existing ones in terms of the resulting average tree cost. The routing and placement techniques were developed for architectures where all the multicast-incapable nodes were *Drop-and-Continue* (DaC) nodes as well as for architectures where all the multicast-incapable nodes were *Drop-or-Continue* (DoC) nodes.

A third major contribution of the thesis is the heuristic algorithms developed

for mixed-graph networks. Up-to-date, research work on multicast routing in optical networks was performed according to the assumption that the network can be modelled as an undirected graph (a graph consisting only of bidirectional connections). However, in the thesis, this work is expanded to the more general case of mixed-graph networks (graphs consisting of both bidirectional and unidirectional connections). As it is reasoned in the thesis these types of networks are closer to the practical case that is encountered during the provisioning request of unprotected and protected multicast requests. The performance of the undirected-graph multicast routing heuristic algorithms if applied in mixed graphs is mathematically analyzed and evaluated through simulations. The mathematical analysis shows that the creation of mixed-graph multicast routing heuristic algorithms is important. Consequently, new multicast routing heuristic algorithms were specifically designed for this category of networks, and analyzed via simulations for randomly created networks. Performance results indicated that these new heuristics outperform (in terms of the cost of the calculated trees) the existing ones that were initially designed for undirected graphs but were applied to the mixed-graph case.

Since, as reasoned in the previous section, it is of critical importance to have efficient fault recovery techniques that can provide the required SLAs for the multicast connections, a fourth contribution of this dissertation is the development of new protection techniques for *single link failure* and *single link/node failure* scenarios that utilize the proposed undirected- and mixed-graph multicast routing heuristic algorithms to provide protection against failures as well. Performance results show that when the newly developed routing techniques are used for protection purposes they are more efficient compared to the case where the existing routing approaches were used. Another contribution in terms of multicast protection in mixed-graph networks is the development of an algorithm that provides protection for multicast connections in mixed-graph networks based on finding pairs of working and protection paths (and being able to deal with certain "trap" topologies where other algorithms fail). This algorithm is also shown to outperform its counterpart (in terms of the cost of the calculated trees) that was designed for undirected graph networks.

Another important contribution of the thesis is the development of a "load balancing" technique that decreases the blocking probability of multicast connections. Until now, the existing algorithms were not taking into consideration the state of the network (due to the already deployed requests), for the establishment of upcoming

multicast requests. The method developed in this dissertation modifies the cost of each network arc according to the distribution of the load in the network, in order to achieve a balanced load distribution and thus to minimize the possibility that the network will be disconnected in some areas due to load congestion on certain arcs, which will subsequently lead to higher blocking probability for the multicast connection requests that follow.

It must be stated that, even though the proposed heuristic algorithms were developed for optical networks, most of them are general, i.e., they can be used in other kinds of networks as well. This fact increases the importance of the proposed multicast routing and protection heuristic algorithms and, consequently, the contribution of the work performed in this thesis.

1.4 Thesis Outline

The outline of the rest of the dissertation is as follows:

Chapter 2 describes the existing state-of-the-art for multicasting in optical networks, including the multicast-capable optical switch architectures for opaque and transparent networks and the mathematical formulation for the routing and wavelength assignment of several (protected) multicast groups for networks with full-splitting capability (i.e., where all network nodes are multicast capable). Some of the most notable heuristic algorithms for multicast routing in full-splitting and sparse-splitting (i.e., where only a fraction of the network nodes are multicast capable) networks are presented as well.

In Chapter 3, the focus is on multicast routing techniques for full-splitting undirected-graph networks. Initially, several existing multicast routing techniques are presented, followed by a newly proposed multicast routing technique (and its generalized version). The theoretical analysis of the proposed technique is given as well. An extensive performance evaluation is obtained via simulations on a widely-used network, on random network topologies, and on other categories of networks (regular, small-world, and scale-free).

Chapter 4 considers architectures with sparse splitting where there are only a few multicast-capable (MC) nodes in the network and the rest of the nodes are multicast incapable (MI). The MI nodes are further classified as drop-and-continue or drop-or-continue. For these types of networks multicast routing techniques as

well as MC-capable node placement approaches are developed. For all proposed techniques, performance evaluation is obtained via simulations on random network topologies.

Chapter 5 focuses on the design of multicast routing heuristic algorithms for mixed-graph networks. Initially, in this chapter an evaluation of the widely-used multicast routing heuristic algorithms is performed for the case of mixed-graph networks, as well as the theoretical calculation of their worst-case performance. A modification of the *Chu Liu* algorithm for multicasting applications is also given. Furthermore, novel heuristic algorithms that are generalizations of the existing ones and are more efficient for mixed graphs are presented, in addition to the mixed graph version of the newly developed multicast routing heuristic algorithm presented in Chapter 3. Again, for all techniques developed performance evaluation is obtained via simulations on random network topologies.

Chapter 6 extends the routing discussion of Chapter 3 by investigating the problem of multicast protection in mesh optical networks. It starts with a description of arc-disjoint protection techniques utilizing the existing multicast routing heuristic algorithms, and continues with the description and performance evaluation of arc-disjoint protection techniques for the newly developed multicast routing heuristic algorithms for undirected graphs as described in Chapter 3. The description of a protection method based on Suurballe's algorithm, that jointly finds pairs of working and protection paths, is given as well, followed by a "load balancing" technique that tries to minimize link congestion and in turn establish dynamic multicast connections more efficiently.

Chapter 7 is devoted to protection of multicast connections in networks with mixed-graph topologies. This is an extension of the routing discussion of Chapter 5 that utilizes arc-disjoint protection techniques while using the existing and the newly developed mixed-graph multicast routing heuristic algorithms. Furthermore, it describes a new approach for protection in mixed graphs that jointly finds pairs of working and protection paths in these types of networks, demonstrating its improved efficiency compared to its undirected counterpart.

Chapter 8 concludes the thesis by summarizing the original contributions of this dissertation and presenting possible research directions that were not covered in this thesis. As pointed out in this chapter, the list of future directions presented is not exhaustive but describes some of the interesting problems that warrant further

investigation.

At the end of the dissertation, some important graph theory definitions that are used throughout the thesis are given in Appendix A. Furthermore, a list of publications that resulted from this thesis and related telecommunication systems work is given in Appendix B.

Chapter 2

Optical Multicasting

2.1 Introduction

Multicasting is the transmission of information from a single source to multiple destinations. It is established by the calculation of a tree on the network graph, that has the source of the multicast request as the root, and spans all the destinations. The problem of finding a cost-efficient multicast tree is equivalent to the classical mathematical *Steiner Tree Problem* (STP) on graphs. Therefore, Steiner tree heuristics are exploited in network applications for the derivation of multicast trees. The calculation of the multicast tree is called *Multicast Routing*. For the transmission of information between the source and every destination, a wavelength must also be selected for each tree arc, for the case of an optical Wavelength Division Multiplexing (WDM) network. This procedure is called *Wavelength Assignment*. Note that in the case of a network that has wavelength converters at each node, different wavelengths can be used in the tree arcs. Otherwise, if no wavelength converters exist, the same wavelength must be used for the entire tree [6].

Therefore, for the establishment of a multicast request, the *Multicast Routing and Wavelength Assignment (MC-RWA)* problem must be solved. When the multicast routing and wavelength assignment is completed, the resulting tree is referred to as a *light-tree* in optical networks [5].

For the implementation of a light-tree, optical splitters are required in the network nodes. An optical splitter is a passive device that optically splits the input signal into multiple identical output signals. The nodes that have the ability of optical splitting, are called *multicast-capable* (MC) nodes, otherwise they are called *multicast-incapable*

(MI) nodes. If all network nodes are equipped with an optical splitter, the network is called a *full-splitting network*. On the other hand, if only a fraction of the nodes has this ability, it is called a *sparse-splitting network*. A summary of optical node architectures that can be used as MC and MI nodes is presented in Section 2.2 that follows.

2.2 Multicasting Optical Node Architectures

2.2.1 Multicast-Capable Network Nodes

There are two approaches to design multicasting switches [8] for the network multicast-capable nodes. The first one is called the *opaque* approach where the optical bit streams are converted into electronic data, switched (and split via duplication) using an electronic cross-connect, and then the electronic bit streams are converted back to the optical domain (the opaque switch architecture is shown in Figure 2.1). The second approach is called the *transparent* (all-optical) approach, where the signal remains in the optical domain throughout and splitting is performed utilizing optical splitters as described previously. An example of a transparent multicast-capable switch architecture is shown in Figure 2.2. This switch design is called a splitter-and delivery (SaD) switch and can be found in [7].

2.2.2 Multicast-Incapable Network Nodes

The multicast-incapable network nodes can be either Drop-or-Continue (DoC) or Drop-and-Continue (DaC) [11]. Optical DoC cross-connects can either terminate the incoming lightpath or allow the incoming lightpath to optically bypass the node. In such architectures, lightpaths are established between source-destination node pairs and cannot be directly utilized by intermediate nodes (Figure 2.3).

In a DaC optical cross-connect, incoming optical signals can be split unequally. Consequently, a small portion of the optical signal can be dropped and processed electronically, while the remaining portion continues to travel optically to the next node with negligible degradation (Figure 2.4). The main advantage of the DaC optical cross-connect architecture is that it allows an available lightpath to be shared by connection requests whose destinations are intermediate or end-nodes of the

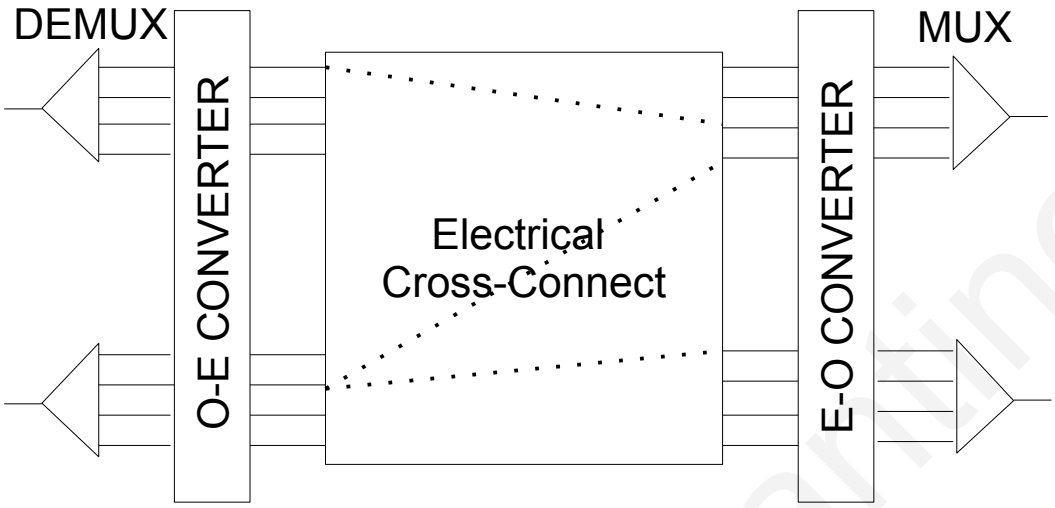


Figure 2.1: Opaque MC node architecture.

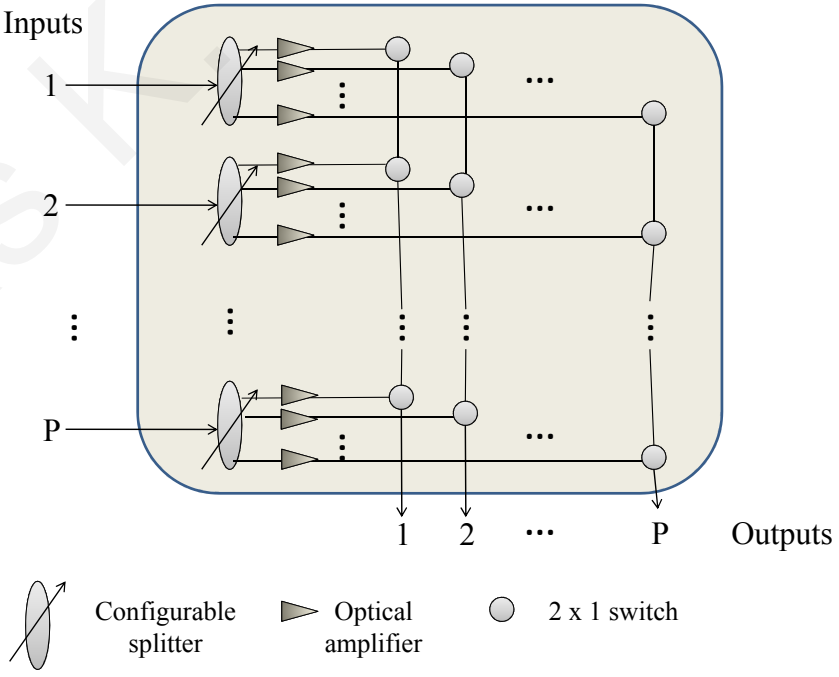


Figure 2.2: Transparent MC node architecture.

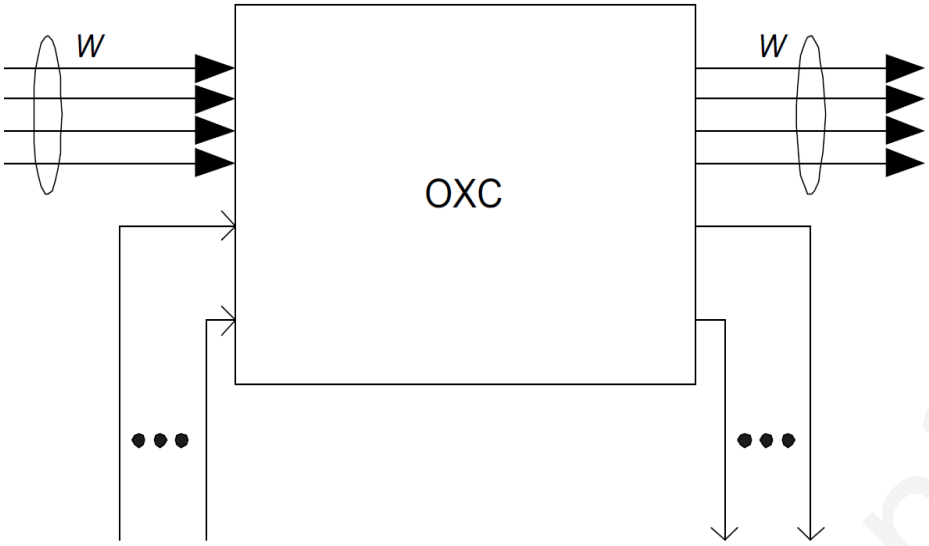


Figure 2.3: DoC MI network node.

lightpath. However, a potential drawback of this architecture is the excessive power loss that a lightpath may experience as it passes through many nodes.

2.3 Multicast Routing and Wavelength Assignment (MC-RWA)

Routing is the procedure of selecting paths (or trees) in a network along which the information will be sent. It is a general engineering problem that is performed for any kind of network, such as telephone, computer, or transportation networks. The node that sends the message is considered as the *source* and the receiving nodes are the *destinations*. In terms of the number of destinations that the information is sent to, there exist three categories of routing:

- unicast routing, where the information is sent to one destination,
- broadcast routing, where all network nodes are considered as destinations, and
- multicast routing, where only a set of nodes will receive the information.

The above three cases are, in fact, three fundamental combinatorial optimization problems: *Shortest Path* (SP), *Minimum Spanning Tree* (MST), and *Steiner Minimal*

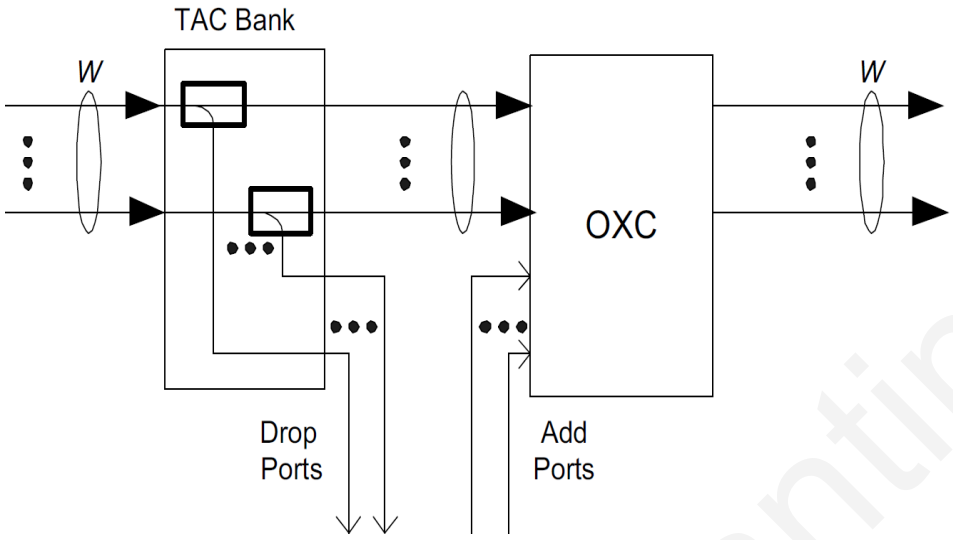


Figure 2.4: DaC MI network node.

Tree (SMT) respectively. Polynomial-time algorithms that give the optimal solution exist only for the SP and MST problems, since SMT is considered an *NP-complete* problem [12] and no optimal polynomial-time algorithms are likely to exist. The optimal solution for this problems can be calculated using *exhaustive search*, i.e., checking all the possible solutions and selecting the one with the least cost. This method is not useful in practice however, due to the huge amount of time and computational power needed, which makes it unrealistic for dynamically provisioned networks. Therefore, approximate algorithms that have efficient average- and worst-case performance have been proposed for the SMT problem. These heuristics have polynomial time complexity i.e., the time needed for the calculation of the solution is a polynomial function of the input of the problem. *Integer Linear Programming* (ILP) can be also used for solving this problem. This method does not have polynomial time complexity, but in practice it is acceptably fast for small size networks. It is not used for real dynamic provisioning applications, but it is exploited in simulations, to be compared with a heuristic algorithm, in order to evaluate the performance of the latter. ILPs are also commonly used in design (static) problems where the computation is performed offline and the time required to obtain the results is not as critical.

Routing is performed according to a specific routing metric. This can be for example the number of hops from source to destination, physical distance from source to destination, monetary cost of each hop, latency, or a combination of them.

Each optical fiber consists of a number of wavelengths. Therefore, for the transmission of information in an optical network, except for the selection of the path or tree along which the information will be sent, a wavelength must be selected for each link that will be used. This procedure is called *wavelength assignment* (WA). The main constraint in assigning wavelengths to different calls in the network is that there cannot exist two calls on the same wavelength traversing the same fiber (color clash constraint). The various WA algorithms depend on the number of wavelength converters [13] that exist in the network. In the absence of them, the same wavelength must be assigned for the whole route (wavelength continuity constraint). If every network node has wavelength conversion capability, a different wavelength can be assigned for every link. A more difficult and complicated case, is the one where the network has sparse placement of wavelength converters. In this case, research focuses not only on the WA algorithms, but also on methods for efficient placement of the limited number of converters as well.

The terms of path (route between the source and a single destination) and tree (route between the source and a group of destinations), in optical networks (when used in combination with the wavelength assignment for that path or tree) are now expressed as *lightpath* and *light-tree* [5] respectively.

Solving the RWA problem jointly (even though more efficient) has been proven to be an NP-complete problem even for the case of unicast connections [3]. Thus, the RWA problem is commonly handled by solving the routing and wavelength sub-problems independently. Specifically, in the case of multicasting, first the tree is calculated and then a wavelength is assigned to each link of it in order to create the light-tree.

Since the majority of today's optical networks allow full wavelength conversion, in that case the solution to the problem of wavelength assignment is trivial; if any free wavelengths exist in a fiber, it can be used for information transmission. if not, that link is considered not to exist in the network. On the other hand, the problem of multicast routing still remains a hard and challenging problem. It is even more difficult for the case of sparse-splitting networks. The main heuristics that are used in optical networks are the Minimum Path Heuristic (MPH) [15] and the Pruned

Prim Heuristic (PPH) (utilizing Prim's MST algorithm [18]) that are described in detail in Chapter 3 that follows.

Furthermore, additional heuristics have been developed for routing and for splitter placement in the case of sparse-splitting networks. The state-of-the-art for the routing heuristics for these types of networks include the Member-Only approach (for DaC networks) [19], the On-Tree MC Node First (OTMCF) and Nearest MC Node First (NMCF) techniques [20], and the Multicasting Using Splitters (MUS) approach [21]. These are described in detail in Chapter 4 of the thesis, together with current approaches in the literature for efficient splitter placement.

2.3.1 Mathematical Formulation

The mathematical formulation for the routing and wavelength assignment of several (protected) multicast groups for networks with full-splitting capability (i.e., where all nodes are multicast capable), is detailed below in order to describe the multicasting RWA problem as an optimization problem [14]:

Input:

1. A physical topology $G_p = (V, E_p)$ is given consisting of a weighted undirected graph, where V is a set of network nodes and E_p is the set of links connecting the nodes. Each link is assigned a weight to represent the cost (number of hops or equipment operating cost) of moving traffic from one end to the other.

2. A group of k primary multicast sessions are given along with a binary digit P_i, $(i = 1, 2, ..., k)$ associated with each of them to indicate whether they require protection or not.

The goal is to set up (if possible) all $2k$ primary and backup multicast sessions on the given physical topology while minimizing the total cost. The cost of a multicast session is the sum of the weights on the physical links occupied by it. If network resources available are insufficient, select as many sessions as possible (preferably the ones yielding maximum revenue) and establish them optimally to minimize the operating cost and, hence, maximize the net profit. In the case of a network with full wavelength conversion capability, the problem is formulated as follows:

1. s and d refer to source node and destination nodes, respectively, in a multicast session.

2. m and n denote endpoints of a physical link that might occur in a light-tree.

3. i is used as an index for session number, where $i = 1, 2, ..., 2k$. Indexes 1 through k are used for the primary trees and indexes $k + 1$ through $2k$ are used for the backup trees. If a primary session i requires protection, then its corresponding backup tree index is $i+k$. Otherwise, the index is left unused, and the variables corresponding to that backup tree are ignored.

Given:

- Number of nodes in the network is equal to N.
- The maximum number of wavelengths per fiber is equal to W.
- The physical topology $P_{m,n}$, where $P_{m,n} = P_{n,m}$ (i.e., fiber links are bidirectional), indicates whether there is a direct physical fiber link between nodes m and n, where $m, n = 1, 2, ..., N$. If there is no fiber link between nodes m and n, $P_{m,n} = P_{n,m} = 0$.
- Every physical link between nodes m and n is associated with a weight $w_{m,n}$.
- The capacity of each channel is equal to C.
- A group of k multicast sessions S_i is given for $i = 1, ..., k$ and $P_i = 1$ or is specified as 0 for each session signifying whether session i requires protection or not, respectively.
- Each session S_1 has a source node and a set of destination nodes denoted by $\{s_i, d_{i,1}, d_{i,2}, ...\}$. The cardinality of a multicast session i is denoted by L_i, which is equal to the number of source and destination nodes in that session.
- Every multicast session is at full capacity of a channel, i.e., at C bps.
- Every node is equipped with wavelength converters capable of converting a wavelength to any other wavelength among the W channels.

Variables:

- A Boolean variable $B^i_{m,n}$, which is equal to 1 if the link between nodes m and n is occupied by the multicast session i; otherwise, $B^i_{m,n} = 0$.

- A Boolean variable V_p^i, which is equal to 1 if node p belongs to multicast session i; otherwise, $V_p^i = 0$. A node belongs to a session if it is either the source or the destination or an intermediate node in the light-tree for the multicast session, e.g., $V_{s_i}^i = 1$ and $V_{d_{i_1}}^i = 1$.

- An integer commodity-flow variable $F_{m,n}^i$. Each destination node for a session needs one unit of commodity. Therefore, there are L_i units of commodity flowing out of source s_i for session i. $F_{m,n}^i$ is the number of units of commodity flowing on the link from node m to node n for session i. $F_{m,n}^i$ is also the number of destination nodes in session i downstream of the link between nodes m and n.

Optimize:

Minimize total cost of all multicast sessions:

$$\text{Minimize} \sum_{i=1}^{2k} \sum_{m,n} w_{m,n} \cdot M_{m,n}^i \tag{2.1}$$

Constraints:

- Tree-creation constraints are:

$$\forall i, \forall n \neq s_i : \sum_{m,n} M_{m,n}^i = V_n^i \tag{2.2}$$

$$\forall i : \sum_{m} M_{m,s_i}^i = 0 \tag{2.3}$$

$$\forall i, \forall j \in S_i : V_j^i = 1 \tag{2.4}$$

$$\forall i, \forall m \neq d_{i_j}, j \geq 1 : \sum_{n} M_{m,n}^i \geq V_m^i \tag{2.5}$$

$$\forall i, m : \sum_{n} M_{m,n}^i \leq D_p(m) \cdot V_m^i \tag{2.6}$$

$$\forall m, n : \sum_{i} M_{m,n}^i \leq P_{m,n} \cdot W \tag{2.7}$$

- The commodity-flow constraints are:

$$\forall i, \forall m, \notin S_i : \sum_n F^i_{m,n} = \sum_n F^i_{n,m} \tag{2.8}$$

$$\forall i, \forall m = s_i : \sum_n F^i_{s_i,n} = L_i \tag{2.9}$$

$$\forall i, \forall m = s_i : \sum_n F^i_{n,s_i} = 0 \tag{2.10}$$

$$\forall i, \forall m = d_{i_j}, j \geq 1 : \sum_n F^i_{n,m} = \sum_n F^i_{m,n} + 1 \tag{2.11}$$

$$\forall i, m, n : M^i_{m,n} \leq F^i_{m,n} \tag{2.12}$$

$$\forall i, m, n : F^i_{m,n} \leq N \cdot M^i_{m,n} \tag{2.13}$$

- The directed-link disjointness is:

$$\forall i = 1, ..., k \forall m, n : M^i_{m,n} + M^{i+k}_{m,n} \leq 1 \tag{2.14}$$

Equation (2) ensures that every node that belongs to a multicast session (except the source) has at least one incoming edge. Equation (3) says that the source node has no incoming edge, as it is the root of the tree. Equation (4) ensures that every source node and the destination node of a multicast session belong to the tree. Equation (5) ensures that every node (except the destination nodes) belonging to the tree has at least one outgoing edge. Equation (6) ensures that every node with at least one outgoing edge belongs to the tree. Equation (7) restricts the number of lightpaths between nodes m and n by $P_{m,n} \cdot W$ in each direction.

Equations (8)−(13) are flow-conservation equations to create a connected tree with the source having a lightpath to every destination in the session. Equation (8) ensures that at any intermediate node (which is neither a source nor a destination), the incoming flow is the same as the outgoing flow. However, outgoing flow at the source node for a session is the number of destinations in the session, and the incoming flow is zero. These are achieved by Equations (9) and (10), respectively. Equation (11) ensures that the total outgoing flow is one less than the incoming flow for destination nodes. Equations (12) and (13) ensure that links occupied by a session have a positive flow and that links not occupied by the session have no flow. In Equation (13), N can be replaced by L_i without altering its meaning. A flow on any link for a multicast session is limited by the number of destinations in that session. Finally, Equation (14) ensures that the primary and backup tree share a link, if any, only in opposite directions.

2.4 Optical Multicast Protection

Since multicasting applications (such as video-on-demand, teleconferencing, distance-learning, remote medical diagnostic applications, etc.) are highly sensitive to network failures, due to the fact that a single link failure may potentially disrupt traffic to a large number of destinations, efficient and reliable recovery from failures is an essential part of optical network operation. Several protection schemes have been proposed in the literature in order to protect the network against equipment, link, and node failures. These include link-based, path-based, segment-based, and cycle-based protection techniques amongst others. The majority of them focuses on protection against single link failures, since this is the predominant case of network malfunction.

2.4.1 Path-Based Dedicated Protection

In this case, a pair of disjoint trees is calculated; one is the working (or primary) tree and the other one is the protection (or secondary) tree. Examples of path-based (in this case tree-based) protection techniques include:

1. **Dual-Tree (DT) Protection:** The most basic approach for dedicated protection of a multicast tree is to build a second tree that does not share any links or nodes (except the source and destination nodes) with the working tree [22]. The two trees are fully disjoint. If a failure occurs on the primary tree, the affected destinations are automatically reconfigured receiving the signal from the backup tree. Thus, the failure is restored automatically and very fast. DT protection consists of the following steps : (i) Calculate the primary light-tree using a multicast routing heuristic algorithm, (ii) Remove all nodes and all links along the primary tree (except for the source and destination nodes) and (iii) Calculate the secondary tree on the resulting graph using a multicast routing heuristic algorithm.

2. **Link-Disjoint Tree (LDT) Protection:** Another approach to protect multicast traffic is to calculate primary and link-disjoint backup multicast trees [23]. This approach can provide (1+1) dedicated protection against link failures. Pitfalls of this approach include excessive use of resources (although less than with DT protection). Also, unlike the DT technique, this scheme cannot protect

against intermediate node failures. LDT consists of the following steps: (i) Calculate the primary light-tree using a multicast routing heuristic algorithm, (ii) Remove all links along the primary tree and (iii) Calculate the secondary tree on the resulting graph using a multicast routing heuristic algorithm.

3. **Arc-Disjoint Tree (ADT) Protection:** An improvement over the link-disjoint approach is arc-disjoint tree protection [23], which can significantly reduce network resources reserved for protection from a link failure. Arcs are defined as unidirectional edges and links are defined as bidirectional edges with an edge in each direction. In arc-disjoint tree protection, a backup tree is constructed which is arc-disjoint from the primary tree. ADT consists of the following steps: (i) Calculate the primary light-tree using a multicast routing heuristic algorithm., (ii) Remove all arcs along the primary tree and (iii) Calculate the secondary tree on the resulting graph using a multicast routing heuristic algorithm.

Path-based dedicated protection approaches can provide very fast protection against failures against the expense of large redundant capacity. Segment-based approaches described below on the other hand try to minimize the usage of redundant capacity while also keeping the recovery time low.

2.4.2 Segment-Based Protection

The working tree is divided into segments and a protection path is calculated for each segment. Examples of segment-based protection techniques include:

1. **Shared Disjoint Segments Protection (SDS):** . In this scheme [23] the authors define a segment as a sequence of arcs from the source or from any splitting point on a tree to a leaf node or to a downstream splitting point. A destination node is always considered as a segment end-node because it is either a leaf node in a tree or is a splitting point where a portion of a signal is dropped locally and the remainder continues downstream (drop-and-continue). SDS consists of the following steps : (i)Compute a primary tree from network graph, (ii) Identify the primary segments on the primary tree, (iii) Reset cost on all arcs used in the primary tree to 0, (iv) For every primary segment repeat Step 5 to 8, (v) Remove links along the primary segment, (vi) Create an arc-disjoint backup

segment, (vii) Reset cost of links on backup segment to 0 and (viii) Replace the links along the primary segment.

2. **Adaptive Shared Segment Protection:** This method [24] exploits two characteristics of multicast trees: (i) a tree does not contain any cycles, and (ii) a tree must contain at least two destination nodes. The authors consider relevant only the segments formed by two destination nodes and an upstream splitting node, which is a common ancestor of both destination nodes. They claim that any such segment is protected if a path could be established between the two destination nodes that forms a cycle in the graph defined by the segment. Therefore, the ASSP algorithm builds the primary multicast tree, creates a set of all destination nodes, and tries to find a shortest path between any two destination nodes that is link-disjoint from the primary multicast tree. The cost of the links in an already-found shortest path is set to 0 in order to maximize intra-tree sharing. The algorithm then selects as backup paths the shortest paths that protect the maximum possible number of relevant segments with the least possible amount of resources.

3. **Dynamic Segment Shared Protection:** This technique is presented in [25]. Each link is dynamically adjusted according to the current network state with proper setting of the link costs to encourage load balancing and sharing. A segment is defined as a sequence of directed wavelength links, from the source or a multicast splitting node on the light-tree to a downstream multicast splitting node or a destination node.

4. **Optimized Collapsed Rings (OCR) Protection:** The OCR scheme [26] is significant because it does not compute a backup tree but traverses the primary tree backwards instead. The primary tree is computed by starting from the source node and visiting all destinations in an optimized order. The first destination visited after the source is the one closest to it. The algorithm computes the shortest path between the source and the destination. All links along the path are removed and the destination becomes the new source. After that, the path is expanded to include the second closest destination and so on until all destination nodes are included in the primary tree. For the backup path, the OCR algorithm computes the shortest path between the original source and the

last destination visited and then traverses the primary tree backwards until the first destination is reached to create a backup tree. The OCR technique consists of the following steps: (i) Given a source s and a set of multicast destinations $\{d_1, d_2, ..., d_n\}$, form a ring set of vertices $\{v_1, v_2, ..., v_n\}$, where v_1 is the source s and $v_2, ..., v_n$ are the multicast destinations, (ii) For primary path, find the closed destination to the source using the shortest path algorithm (e.g., node d_i). Remove all the links along the path. The new source is set to be node d_i, (iii) Repeat Step 2 until no more vertices are included in the ring set and (iv) For backup path, find the shortest path from node s to the last destination node in the primary path and traverse the primary path backwards until the first destination is reached.

5. **Optimal Path Pair-based Shared Disjoint Paths (OPP-SDP) Protection:** This scheme [23] finds a pair of paths between the source and each destination of the multicast request. It consists of the following steps: (i) For every destination node repeat Steps 2 and 3, (ii) Find optimal path-pair between source and destination nodes (using Suurballe's algorithm [27]) and (iii) Reset cost for already found optimal path-pairs to 0.

2.4.3 Cycle-Based Protection

Cycle-based protection techniques also try to minimize the usage of redundant capacity while also keeping the recovery time low by breaking the network into cycles and utilizing these cycles for protection purposes. An example of a p-cycle based approach for multicast protection in optical networks is described in this section. This technique is called the *intelligent p-cycle protection* technique and it was first presented in [28]. In this approach, when a multicast request arrives, a multicast tree is computed for it (using any known multicast routing algorithm) and then the intelligent p-cycle scheme is used to compute a set of efficient p-cycles on demand to protect the multicast tree. p-Cycles are a cycle decomposition of the network so that all network links are either on the cycles or are chords of these cycles and were first introduced in [29]. The proposed scheme has the following features: (i) it provides fast restoration since preconfigured p-cycles are used to protect the multicast tree links; (ii) it makes efficient use of spare capacity since a set of high-efficiency p-cycles are computed on demand to protect the multicast tree

links; (iii) both intra-session sharing and inter-session sharing are achieved since a p-cycle can provide protection to links belonging to not only the same multicast tree, but also different multicast trees; (iv) the capacity efficiency is further improved by combining the existing p-cycles whenever possible; (v) assuming sufficient capacity is available in the network, a set of p-cycles can always be found to protect any multicast tree as long as the network is two-edge-connected.

Clearly, there are a multitude of techniques that can be used for protecting multicast connections in WDM optical networks with arbitrary mesh topologies. In this thesis we focus on dedicated tree-based solutions as they provide the fastest recovery speed, trying to limit the average cost of the working and protection trees as well as the blocking probability that the connections may experience (assuming that for a connection to be established, both the working and protection trees must be provisioned).

Chapter 3

Full-Splitting Undirected-Graph Networks: Multicast Routing Heuristic Algorithms

3.1 Introduction

The Steiner Tree Problem (STP) is a classical combinatorial optimization problem that tries to find the shortest interconnection for a given finite set of points. It has applications in circuit design, network design (multicast routing), computational biology, transportation networks, and other fields. There are several variations of it: Euclidean Steiner Tree, Rectilinear Steiner Tree, and Steiner Tree in graphs [30]. The work in this dissertation is related to the latter case.

STP in graphs is defined as follows: Given a connected, undirected, simple, weighted graph $G(V, E)$ with a positive cost function for every edge, and a subset Z of V, find the connected acyclic subgraph $T^*(V^*, E^*)$ with the minimum cost among all connected subgraphs that contain set Z. An acyclic subgraph $T^*(V^*, E^*)$ of graph $G(V, E)$ is an acyclic graph such that $V^* \subset V$ and $E^* \subset E$. T^* is a tree, called *Steiner Minimal Tree* (SMT). Except of the required vertices of set Z, T^* possibly contains some other vertices $S \subset (V - Z)$, which are called *Steiner vertices*.

Let $|V| = n$ and $|Z| = k$. For $k = 2$, the problem is reduced to the extensively studied "shortest path problem". For this problem, there are several algorithms that give an optimal solution in polynomial time. The most widely used are Dijkstra's [31], Bellman-Ford's [32, 33], and Floyd-Warshall's [16, 17]. For $k = n$, the problem is

reduced to the "minimum spanning tree problem". This is also a classical problem which has polynomial-time algorithms that solve it optimally. The most commonly used are Prim's [18], Kruskal's [34], and Boruvka's [35].

For the rest of the cases the problem is NP-complete, hence it cannot be solved exactly in polynomial time [12]. In practice, heuristic algorithms are used. When this problem is applied to multicast routing, one of the vertices in Z is considered as the source s and the rest as the destinations $d_i, 1 \leq i \leq k-1$. Regardless of the selection of the source, the resulting tree is the same.

The *worst* case performance of a Steiner tree heuristic algorithm is given by the *Steiner Ratio* (*SR*) (called as *Approximation Ratio* as well). *SR* is equal to the upper bound on the ratio of the cost of the found solution over the cost of an optimal solution [36]. It is calculated theoretically.

Clearly, the best heuristic algorithms are the ones that have an SR ratio close to 1. A number of heuristic algorithms have been defined theoretically that give $SR < 2$ (for example in [36] a heuristic that has $SR = 1.55$ is presented). The *SR* of a heuristic algorithm gives the worst-case performance of it. However, the *average-case* performance for the cost of the trees compared to the optimal solution is what is interesting in network applications and it is something that cannot be calculated formally but can only be derived using simulations.

3.2 Existing SMT Heuristic Algorithms

The straightforward method for establishing a multicast connection is to calculate a unicast path from the source node to every destination node. It is obvious from Figure 3.1 that this method is not cost-effective.

A more effective approach to solve the problem is to find a tree that connects the source to the set of destinations. This tree is called a "Steiner tree". There are various approaches in the literature for calculating the Steiner tree in graphs [36-44]. A simple way to do this is to calculate the minimum spanning tree (MST) of the graph using one of the classical algorithms, and then delete the unnecessary edges. Assuming Prim's algorithm [18] is used to find the MST, the heuristic algorithm is called the Pruned Prim's Heuristic (PPH) and it has time complexity $O(elogn)$ ($e = |E|$ and $n = |V|$) [45]. Another heuristic algorithm widely utilized in the literature to solve the STP, with better performance than the PPH, is the "Minimum Path Heuristic"

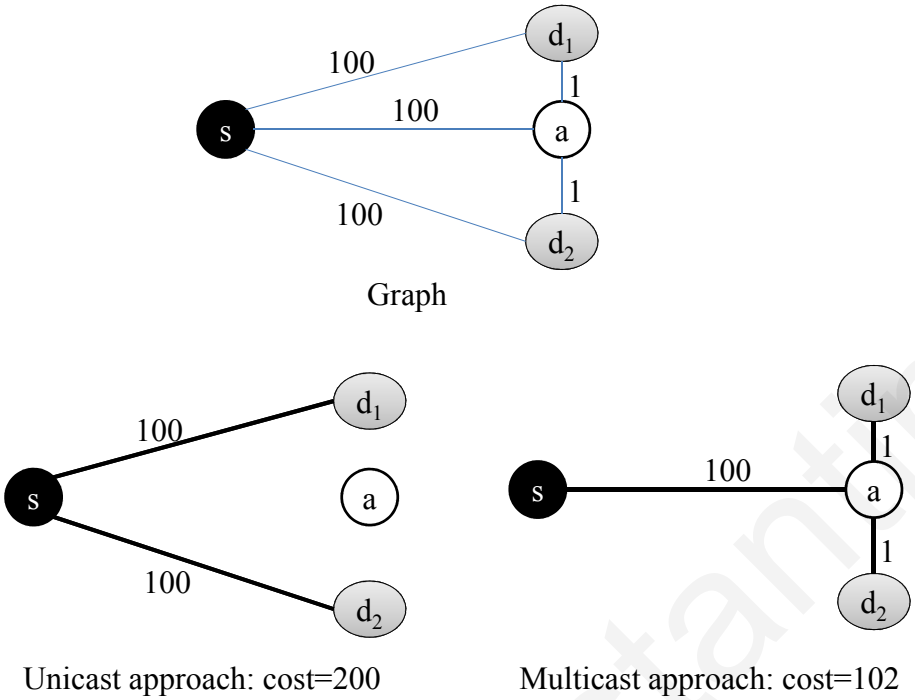


Figure 3.1: Comparison of unicast-based and multicast-based routing.

(MPH) [15]. MPH consists of the following steps:

Input: Graph $G(V, E)$, set Z of vertices that must be connected.

Output: Steiner Tree $T^*(V^*, E^*)$.

1. Choose an arbitrary vertex v in Z and define a tree T_1 consisting of v only. $i = 1$.
2. Determine a vertex u in $Z - V(T_i)$, closest to T_i. Construct a tree T_{i+1} by adding the minimum cost path joining u to T_i. $i = i + 1$.
3. Repeat Step 2. STOP when all vertices of Z are connected to the tree.

where the shortest paths (and the corresponding distances) between every pair of graph nodes, are calculated using Floyd-Warshall's all-pair shortest path algorithm [16, 17].

MPH works as follows: In the beginning, the tree consists of only one vertex that is chosen arbitrarily from set Z. In each step, the algorithm finds the shortest path that connects the current tree and one of the unconnected vertices of set Z. The algorithm terminates when all vertices of Z have been connected to the tree. While MPH is a very popular STP heuristic algorithm, it has a weakness; it cannot

find the vertices that, if added in the tree, will decrease its cost. To explain exactly the weakness of MPH and in fact the difficulty of this interesting problem, Steiner vertices are divided in two categories: the Required Steiner Vertices *(RSV)* and the Optional Steiner Vertices *(OSV)* for a destination. An *RSV* is every vertex in $(V - Z)$ that belongs to the tree calculated by MPH, and an *OSV* is every vertex not belonging either in MPH-tree or in Z. A Basic Steiner Vertex *(BSV)* is an *OSV* which if it is used (i.e., added in Z), it will give a Steiner Tree with less cost.

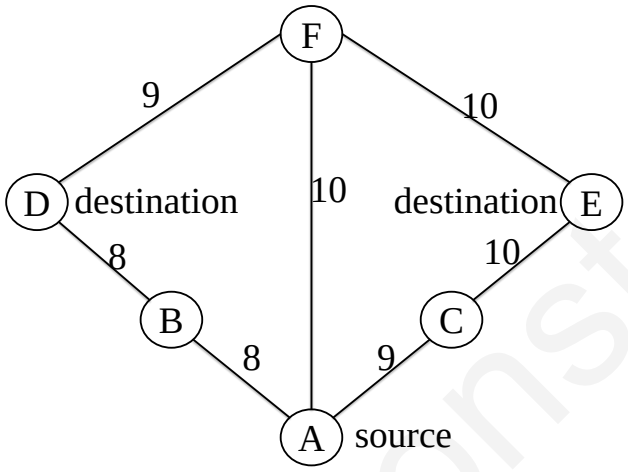


Figure 3.2: Example test network demonstrating the weakness of MPH.

Consider the example network shown in Figure 3.2. Vertex A is considered as the source node and vertices D, E as the destinations d_1 and d_2 of the multicast call. If MPH is implemented, it will connect vertex D through path *A-B-D* (path cost=16 units) and E through *A-C-E* (path cost=19 units) and the total tree cost will be 35 units. Therefore, according to the aforementioned definitions, vertices B and C are *RSVs* and vertex F is an *OSV*. An alternate path for D is *A-F-D* (path cost=19 units) and for E is *A-F-E* (path cost=20 units). If these paths are used, the total tree cost will be 29 units. This result makes vertex F a *BSV*. It is obvious that, if vertex F is added in Z, MPH will calculate these paths and, consequently, will give a tree with less cost. If an algorithm has the ability to find these *BSVs* (that MPH fails to find), it will be able to find Steiner trees with less cost. In the following section, a novel heuristic algorithm is described, called *Steiner Node Heuristic* (SNH), that is capable of finding these critical vertices and, consequently, of finding a Steiner tree with less cost.

Examples of other heuristic algorithms that deal with the Steiner tree problem in

graphs are *Zelikovsky's* algorithm [38] and *Robins-Zelikovsky's* algorithm [36]. Other heuristic algorithms can also be found in [37, 39-44].

Zelikovsky's algorithm finds, as long as possible, the best reduction of the cost of the initially calculated Steiner tree, by recursively adding to it three edges with a common end, and removing the longest edges from each resulting cycle. After the triple contraction the next Steiner tree is obtained, until no further cost reduction is possible. Robins-Zelikovsky algorithm repeatedly chooses appropriate full components and then contracts them to form the overall solution. (A Steiner tree over a subset S^* of the set S of nodes to be connected, in which all nodes of set S^* (called *terminals*) are leaves is called a full component). It discards an already accepted full component, if later a better full component, that conflicts with this previously accepted component, is found out (two components conflict if they share at least two terminals). The main idea is to contract as little as possible so that a chosen full component may still participate in the overall solution, but not many other full components would be rejected. It iteratively modifies the initially calculated Steiner tree, by incorporating into this, loss-contracted full components greedily chosen. *Loss* serves as an upper bound on the optimal solution cost increase during the algorithm's execution. As soon as the algorithm selects a full component K it contracts its $Loss(K)$, i.e., collapses each connected component of $Loss(K)$ into a single node. More details can be found in the corresponding paper [36].

It must be stated that due to their implementation complexity and their running time, none of the aforementioned heuristic algorithms has been exploited in the literature for multicasting applications in optical or other telecommunication networks. Their theoretically calculated improved performance describes their worst-case performance. In practical applications, however, it is the average case performance that counts. The aforementioned heuristic algorithms have improved performance only under certain trap topologies, that are not met in real networks.

In optical networks, the widely-used Steiner tree heuristic algorithms are MPH and PPH. Examples of papers that utilize them are [10, 20, 23, 46-51].

3.3 Steiner Node Heuristic Algorithm

The Steiner Node Heuristic (SNH) uses MPH as its basis, but manages to find all Basic Steiner Vertices (BSVs). It consists of the following steps:

Input: Graph $G(V, E)$, set Z of vertices that must be connected.

Output: Steiner Tree $T^*(V^*, E^*)$.

1. Calculate Steiner Tree $T^*(V^*, E^*)$ using MPH and find its cost, called C.

2. For every vertex $v_i \in \{V - V^*\}$ do the following:

   (a) $\{Z'\} \leftarrow \{Z\} + v_i$.

   (b) Calculate Steiner Tree $T_i(V_i, E_i)$ on $G(V, E)$ for set $\{Z'\}$, using MPH, and find its cost, c_i.

   (c) $\{Z'\} \leftarrow \{Z\}$.

3. Find v_j and $T_j : c_j = min_i\{c_i\}$.

4. If $c_j \geq C$, the solution is $T^*(V^*, E^*)$, STOP.

   (a) If $c_j < C$, replace $\{Z\}$ with $\{Z\} + v_j$, C with c_j, and $T^*(V^*, E^*)$ with $T_j(V_j, E_j)$.

   (b) If set $\{V - V^*\} \neq 0$, return to Step 2. Else STOP.

Explanation of SNH: The key function of the heuristic is the discovery of those vertices that the addition of them in set Z gives a less-cost tree. In the first step, the Steiner Tree using MPH is calculated (called "TREE" (TR) here, for explanation purposes). Then, in Step 2, for every vertex in graph G but not in TR, MPH is utilized again for the calculation of the corresponding tree, if the specific vertex was temporarily in set Z. After the comparison of these trees, the one with the least cost (called "New Tree" (NT) here, for explanation purposes) is kept. If $\text{cost}_{NT} \geq \text{cost}_{TR}$, the heuristic terminates and tree TR is the final solution, whereas if $\text{cost}_{NT} < \text{cost}_{TR}$, tree TR is replaced with NT, the corresponding vertex is added permanently in set Z and Step 2 is repeated. The heuristic terminates, with tree TR as the final solution, either if, after the procedure of Step 2 NT has more than or equal cost to TR, or if all vertices of the graph belong to TR.

Example of SNH: In the following example a more detailed explanation of SNH is given, as well as a comparison with MPH. The network graph is given in Figure 3.3a. Node A is considered as the source and nodes B, C, F, G as the destinations.

When MPH is used, the following steps take place:

- Step 1: A is chosen arbitrarily as the source.

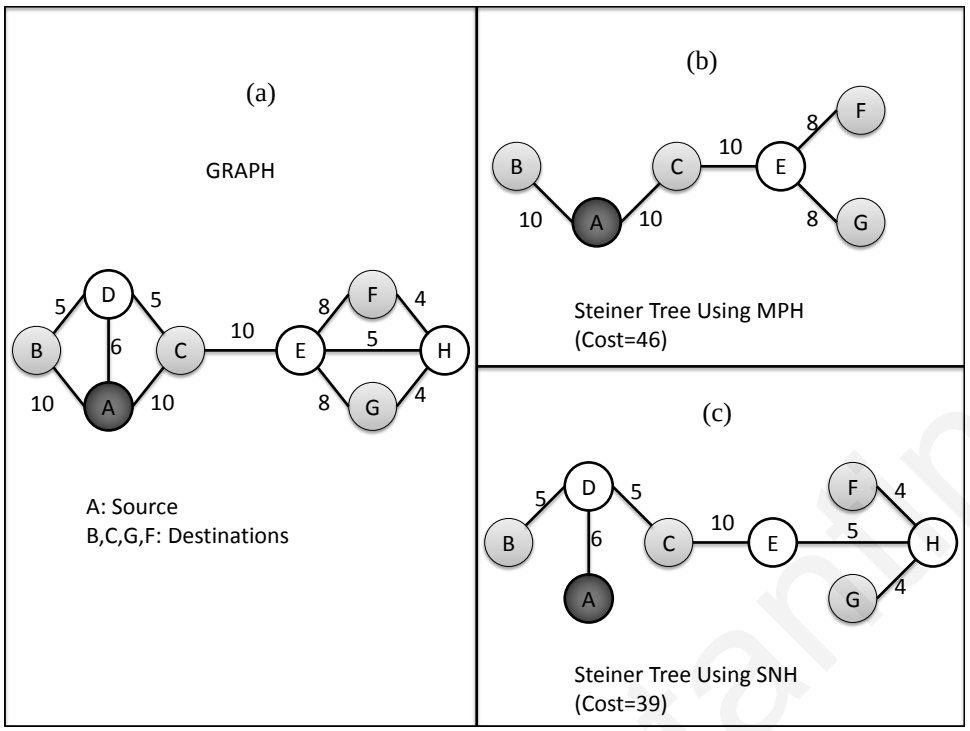


Figure 3.3: SNH example.

- Step 2: The destination that is closest to the source A is B (or C). The selected path is $A - B$.

- Step 3: The procedure is continued and the connected vertices are C (path $A - C$), F (path $C - E - F$), and G (path $C - E - G$). The heuristic terminates and the resulting tree has cost=46 as shown in Figure 3.3b.

When SNH is used, the process is as follows:

- Step 1: Start with the calculated MPH tree $T^*(V^*, E^*)$, with cost equal to $C = 46$.

- Step 2: The set $\{V - V^*\}$ consists of vertices D, H. If D is added (temporarily) in set Z, the resulting tree has cost equal to 42. After the removal of D and (temporary) addition of H in Z, the resulting tree has cost equal to 43.

- Step 3: From the vertices in set $\{V - V^*\}$, D gives the minimum cost if added in Z.

- Step 4: Vertex D is added (permanently) in set Z, $C = 42$, $T^*(V^*, E^*)$ is replaced with the tree derived after the addition of D in Z, and Step 2 is visited again. Now the only vertex in $\{V - V^*\}$ is H. The resulting tree, if H is added in Z (considering that now D belongs to Z as well), has cost equal to 39. Hence, H

is added permanently in set Z, $C = 39$ and $T^*(V^*, E^*)$ is replaced with the tree derived after the addition of D and H in Z. The set $\{V - V^*\}$ now is empty and the heuristic terminates. The resulting tree is presented in Figure 3.3c with cost=39.

According to the aforementioned terminology, vertex E is an *RSV* and vertices D, H are *OSVs* and, eventually, *BSVs*. A theoretical analysis of SNH is given in the following section.

3.4 Theoretical Analysis

3.4.1 Performance of SNH for Unicasting and Broadcasting

Theorem 3.1. *SNH is optimal for unicasting and broadcasting.*

Proof. MPH gives optimal results for the case of $k = |Z| = 2$ and $k = n$. Since SNH uses MPH as its basis and applies a recursive procedure to improve the solution obtained by the latter, it always performs at least as efficiently as MPH. Therefore, for the cases of $k = |Z| = 2$ (unicasting) and $k = n$ (broadcasting), it has optimal results. □

3.4.2 Maximum Possible Number of BSVs in a Graph

Theorem 3.2. *Every Basic Steiner Vertex (BSV) connects at least two destinations with the source.*

Proof. If a node x connects only one destination with the source, this path is calculated by MPH, and x is not a *BSV*, according to the definition of the latter. Therefore, each *BSV* connects at least two destinations with the source. □

Lemma 3.1. *The maximum possible number of BSVs that may be added in Z is $\lfloor \frac{D}{2} \rfloor$, where $D = k - 1$ is the number of destinations.*

Proof. The maximum possible number of *BSVs* that may be added in Z appears in the case where each *BSV* connects the least possible number of destinations. i.e., two destinations, according to Theorem 3.2. Therefore, the maximum possible number of *BSVs* that may be added in Z is equal to

$$|BSV|_{max} = \lfloor \frac{D}{2} \rfloor \tag{3.1}$$

□

3.4.3 Maximum Possible Cost Reduction of SNH Compared to MPH

The *Cost Reduction* (CR) of SNH compared to MPH is defined as the cost of the tree obtained by MPH over the cost of the tree obtained by SNH:

$$CR = \frac{MPH_{cost}}{SNH_{cost}} \tag{3.2}$$

Theorem 3.3. *The maximum possible value of CR is 2.*

Proof. The worst-case performance of MPH is the calculation of a tree that has cost twice the cost of the optimal tree [15], i.e., $max\{\frac{MPH_{cost}}{OPT_{cost}}\} = 2$, where $\frac{MPH_{cost}}{OPT_{cost}}$ is calculated over all possible graphs, for the derivation of its maximum value (OPT_{cost} stands for the cost of the optimal tree).

The best case performance of SNH is the calculation of the optimal tree i.e., $min\{\frac{SNH_{cost}}{OPT_{cost}}\} = 1$.

The maximum value of CR is equal to:

$$max\{CR\} = max\{\frac{MPH_{cost}}{SNH_{cost}}\} = max\{\frac{\frac{MPH_{cost}}{OPT_{cost}}}{\frac{SNH_{cost}}{OPT_{cost}}}\} = \frac{max\{\frac{MPH_{cost}}{OPT_{cost}}\}}{min\{\frac{SNH_{cost}}{OPT_{cost}}\}} \tag{3.3}$$

If a graph exists where $\frac{MPH_{cost}}{OPT_{cost}}$ takes its maximum value and $\frac{SNH_{cost}}{OPT_{cost}}$ takes its minimum value, Theorem 3.3 is proven. This graph is given in Figure 3.4:

In this figure, s is the source, d_1, d_2, ..., d_n are the destinations, and $n \gg \alpha \gg \varepsilon$. MPH will give tree $s - d_1$, $s - d_2$, ..., $s - d_n$ that has a cost equal to $2\alpha n$, while the tree derived by SNH (which is the optimal one as well), is $s - x$, $x - d_1$, $x - d_2$, ..., $x - d_n$, with a cost equal to $\alpha + \varepsilon + \alpha n$. Therefore, for this case,

$$\frac{MPH_{cost}}{OPT_{cost}} = \frac{2\alpha n}{\alpha + \varepsilon + \alpha n} = \frac{2\alpha n}{\alpha(n+1) + \varepsilon} \approx \frac{2\alpha n}{\alpha n} = 2 \tag{3.4}$$

$$\frac{SNH_{cost}}{OPT_{cost}} = 1 \tag{3.5}$$

$$(3.4), (3.5) \Rightarrow CR = 2 \tag{3.6}$$

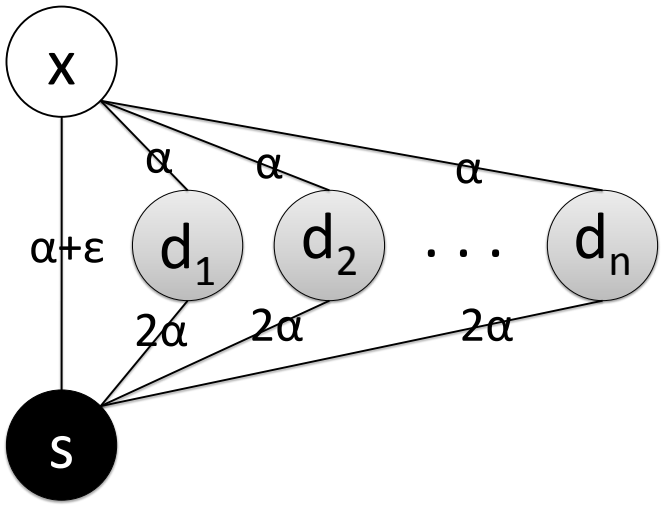


Figure 3.4: Graph topology for CR=2.

□

3.4.4 Complexity of SNH

The derivation of the time complexity for SNH is as follows:

- The MPH heuristic, which is the basis for SNH, has time complexity $O(kn^2)$ [15], where k and n were previously defined.
- The set $\{V - V^*\}$ has at most n elements. Therefore, each iteration of Step 2 of the SNH algorithm, multiplies the order of time-complexity by n.
- According to Equation 3.1, Step 2 is repeated no more than $\lfloor (k-1)/2 \rfloor + 1$ times. Hence, the order of time-complexity is multiplied by k.

The result of the three previous statements is that the time-complexity of SNH is $O(k^2n^3)$.

3.5 Performance Evaluation

The performance of SNH was evaluated via simulations, using the sample network (USNet) shown in Figure 3.5, as well other randomly created networks.

The network of Figure 3.5 consists of 24 vertices and 43 links (*bidirectional* connections) between vertex pairs. The numbers shown for every link in the network graph represent the link cost (in terms of physical distance, monetary cost, etc). Let

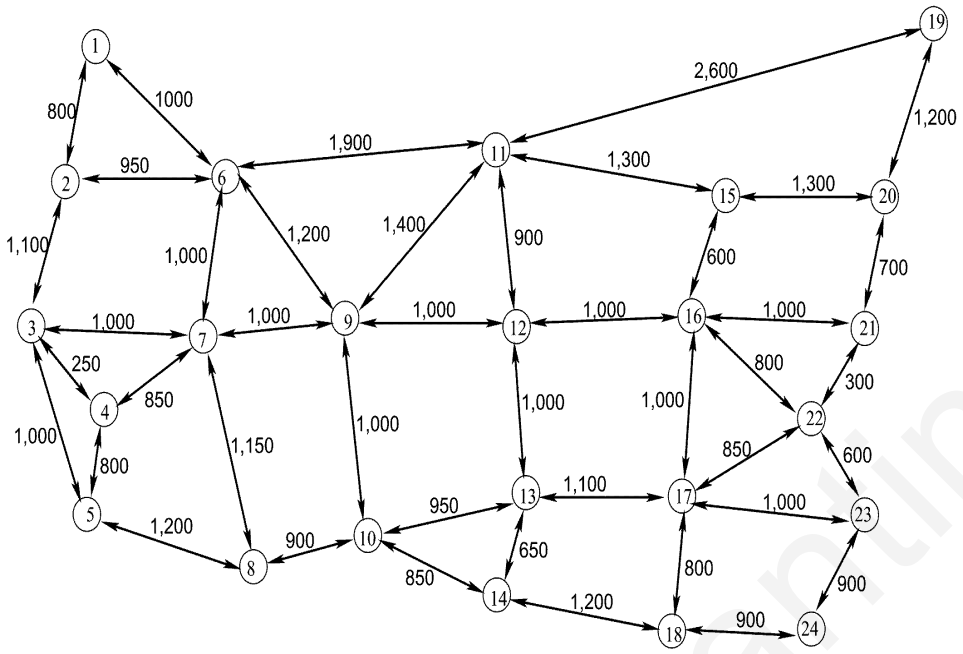


Figure 3.5: Sample network (USNet) used in the simulations.

D be the number of destinations of the multicast group that must be connected. The experiment was repeated for all possible multicast groups (from $D = 1$ (unicasting) to $D = 23$ (broadcasting)).

The following procedure was repeated for every possible D: Fifty thousand (50000) multicast calls were randomly created, with the vertices of each connection uniformly distributed across the network. Each multicast call was assumed to depart from the network before the arrival of the following ones as in this case the interest is in the cost of the trees generated, without any blocking in the network. The multicast routing tree was calculated with both the MPH and SNH heuristic algorithms, for each one of the 50000 calls. The average cost of the trees derived by MPH and by SNH, over the 50000 calls, is denoted by MPH_{cost} and SNH_{cost} respectively. The $\%Gain$ of the performance of SNH compared to MPH is calculated as follows:

$$\%Gain = 100 * \frac{MPH_{cost} - SNH_{cost}}{MPH_{cost}} \tag{3.7}$$

The results of the simulation for the USNet are presented in Figure 3.6. The performance of the SNH was also evaluated through simulations on randomly created undirected network graphs as well, with varying number of nodes and links. Let the *nominal distance* (d^{ij}_{nom}) between two nodes i and j of the created graphs be defined

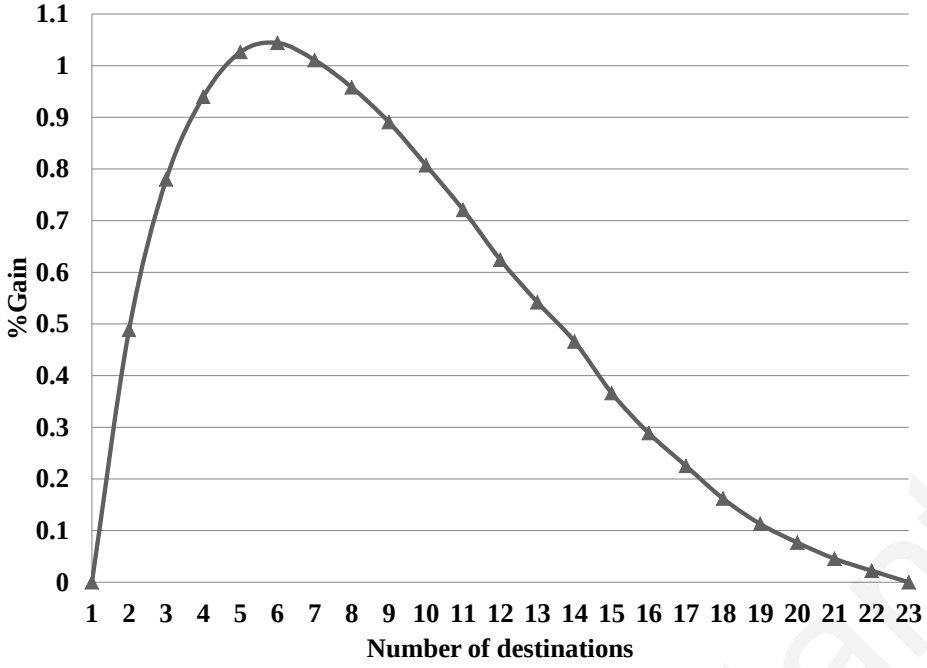


Figure 3.6: Performance gain of the SNH heuristic compared to the MPH approach.

as $d_{nom}^{ij} = |i - j|$. The constraint that every arc that was added in the graph had to connect nodes that satisfy $d_{nom}^{ij} \leq x \; \forall i, j, (1 \leq x \leq k)$ was used for some of the created graphs. The reason is that the graph must simulate a real network, where nodes that are connected belong to the same neighborhood. Simulations were performed with and without taking this constraint into account. A randomly calculated cost varying from 1 to 1000 was also assigned to each network link.

The simulation was repeated for various possible multicast group sizes. The experiment was executed 50000 times for every multicast group size, as described previously. The results are presented in Figures 3.7-3.14 (the characteristics of each graph derived from the simulation are given in each figure's caption).

Simulations were performed on other categories of networks as well. Figures 3.15, 3.16, and 3.17 give the comparison of MPH and SNH heuristics for regular, small-world, and scale-free networks respectively.

Analysis of the Results

For unicast and broadcast calls, MPH and SNH give trees with equal cost. This was expected, because for these two cases MPH is optimal. For every other case (i.e., multicast calls), for all types of networks, SNH always gives better results. SNH has improved performance regardless of the characteristics and the size of the network graphs (up to 3.5% reduction to average tree cost). The maximum gain appears

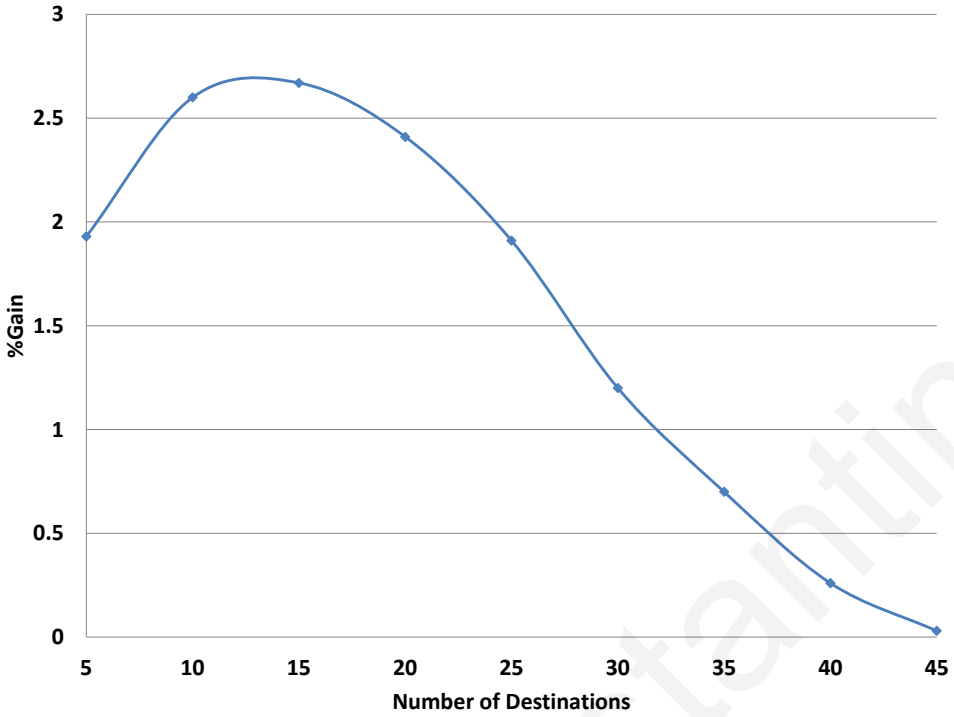


Figure 3.7: Performance gain of the SNH heuristic compared to the MPH approach: Graph with 50 nodes, 100 links, $d_{nom} \leq 5$.

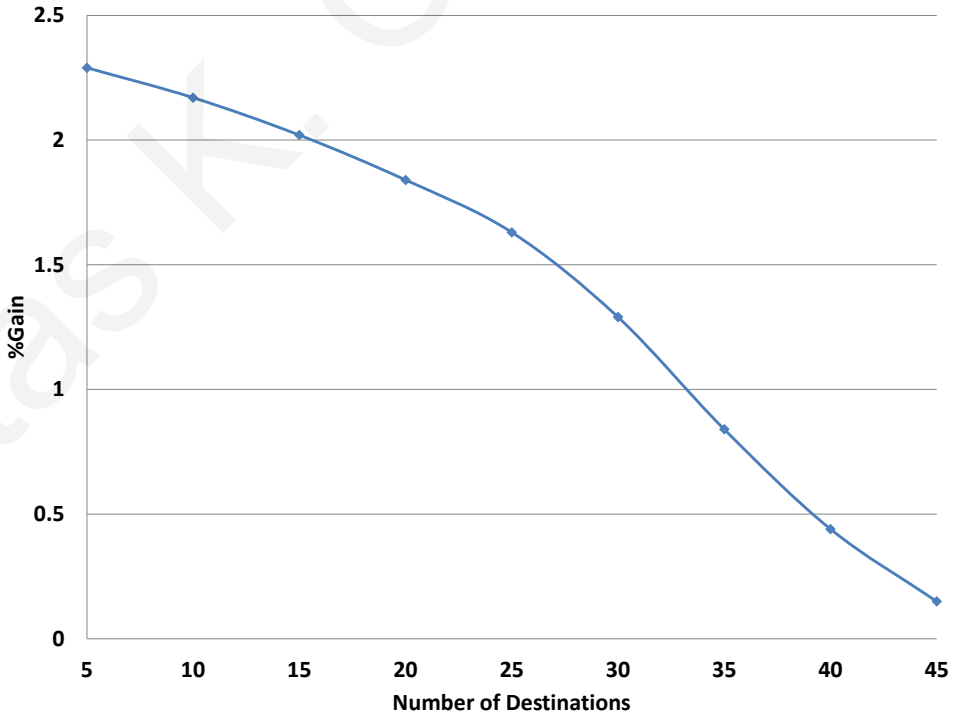


Figure 3.8: Performance gain of the SNH heuristic compared to the MPH approach: Graph with 50 nodes, 100 links, $d_{nom} \leq 50$.

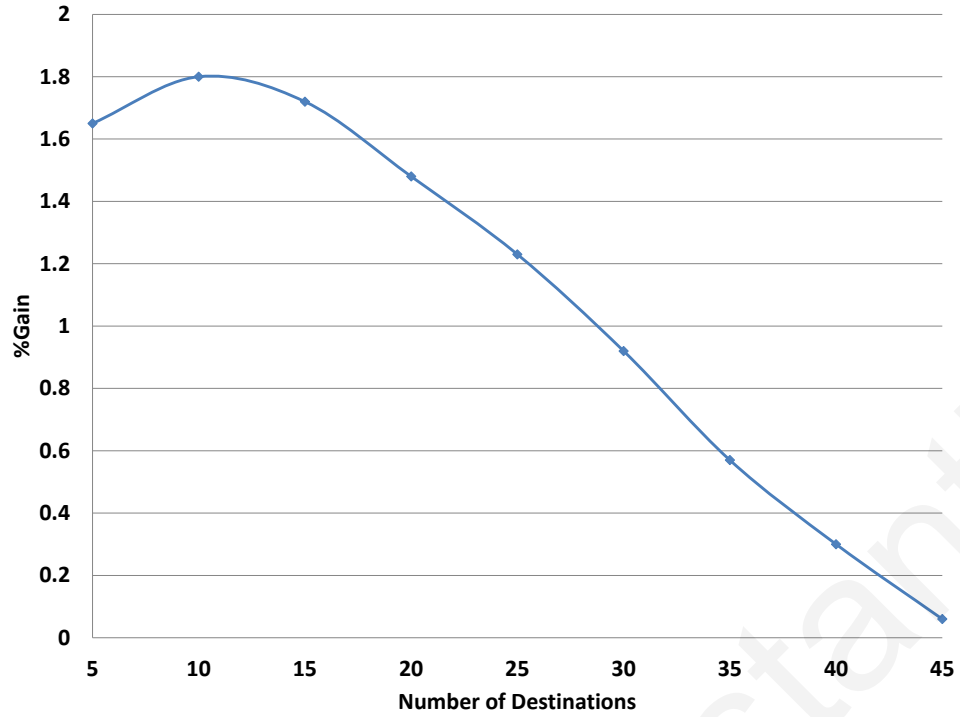


Figure 3.9: Performance gain of the SNH heuristic compared to the MPH approach: Graph with 50 nodes, 200 links, $d_{nom} \leq 5$.

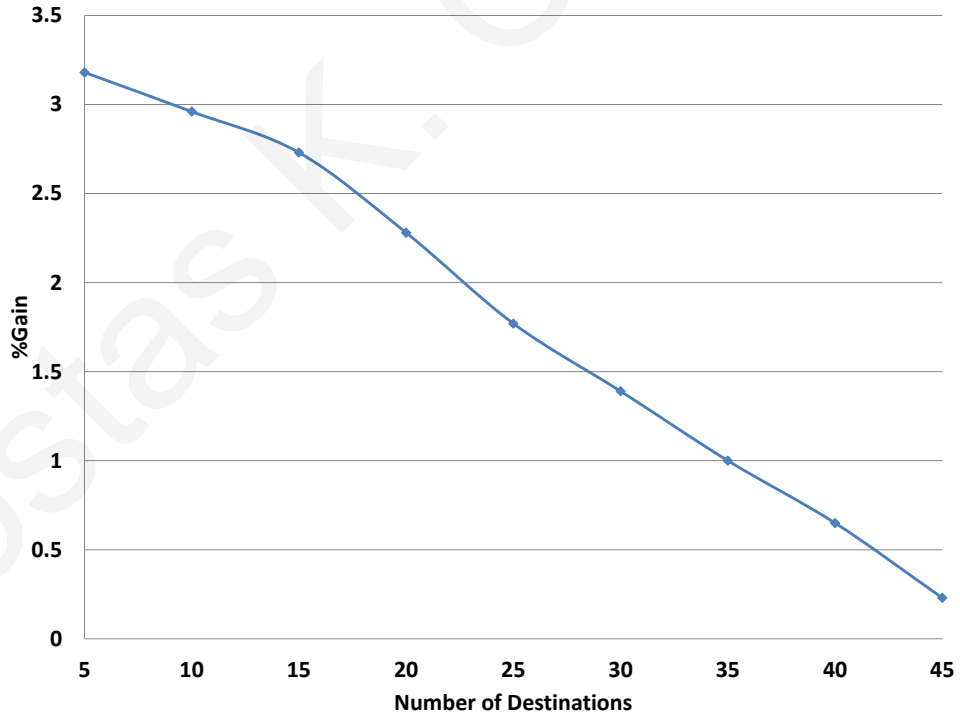


Figure 3.10: Performance gain of the SNH heuristic compared to the MPH approach: Graph with 50 nodes, 200 links, $d_{nom} \leq 50$.

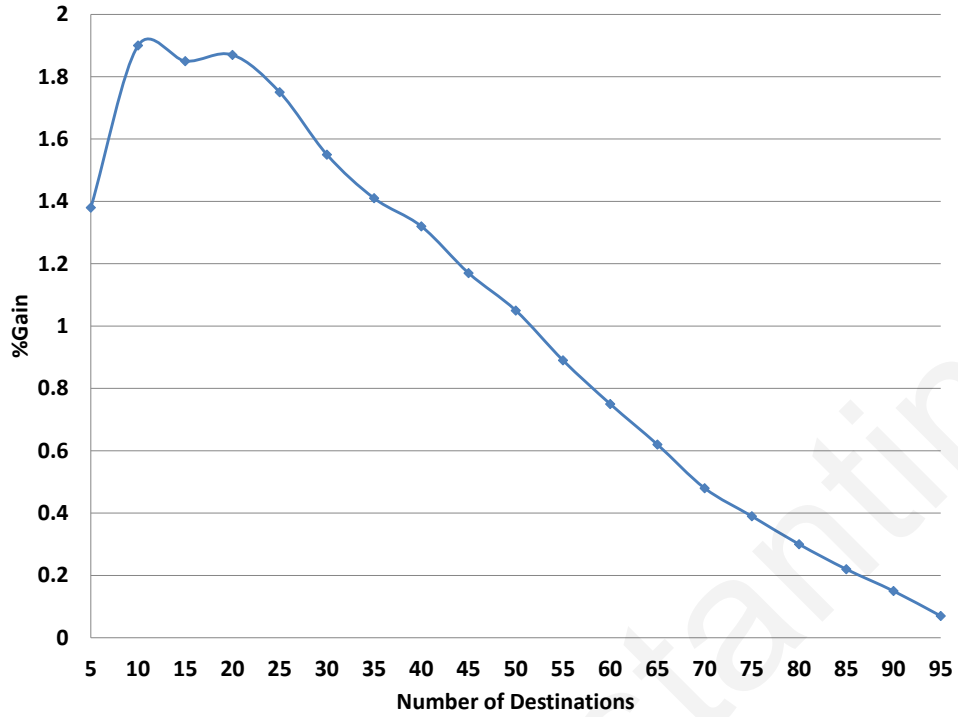


Figure 3.11: Performance gain of the SNH heuristic compared to the MPH approach: Graph with 100 nodes, 200 links, $d_{nom} \leq 10$.

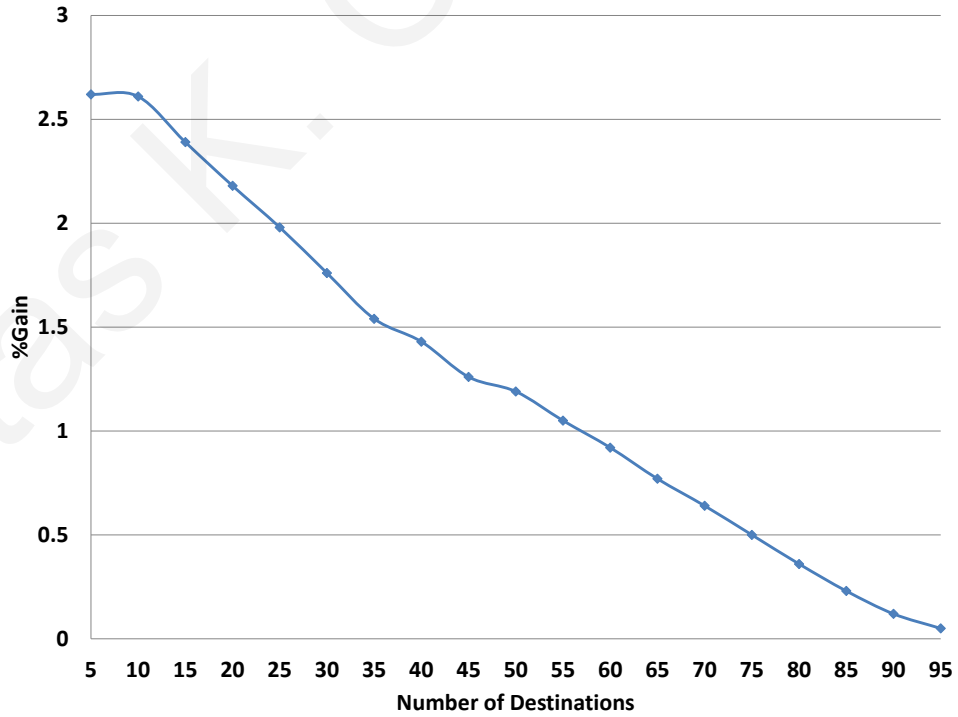


Figure 3.12: Performance gain of the SNH heuristic compared to the MPH approach: Graph with 100 nodes, 200 links, $d_{nom} \leq 100$.

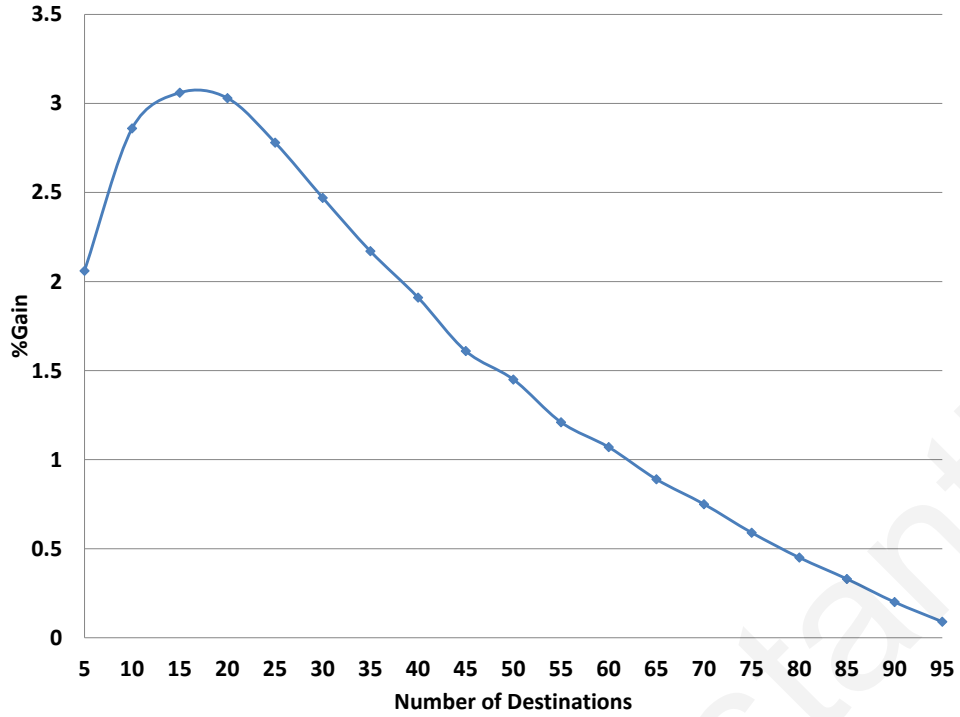


Figure 3.13: Performance gain of the SNH heuristic compared to the MPH approach: Graph with 100 nodes, 400 links, $d_{nom} \leq 10$.

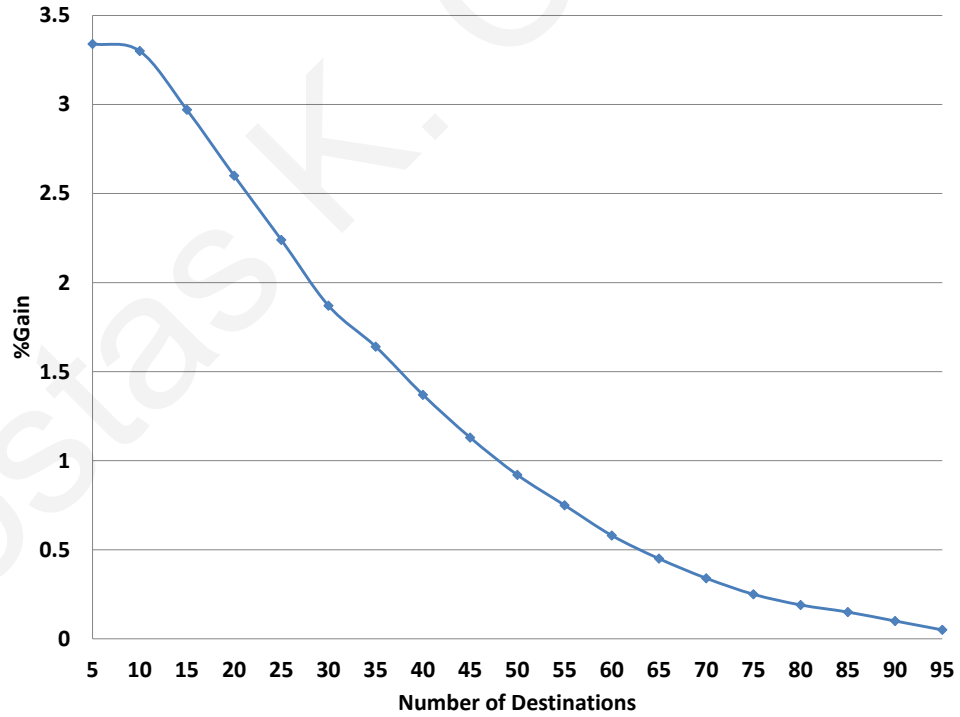


Figure 3.14: Performance gain of the SNH heuristic compared to the MPH approach: Graph with 100 nodes, 400 links, $d_{nom} \leq 100$.

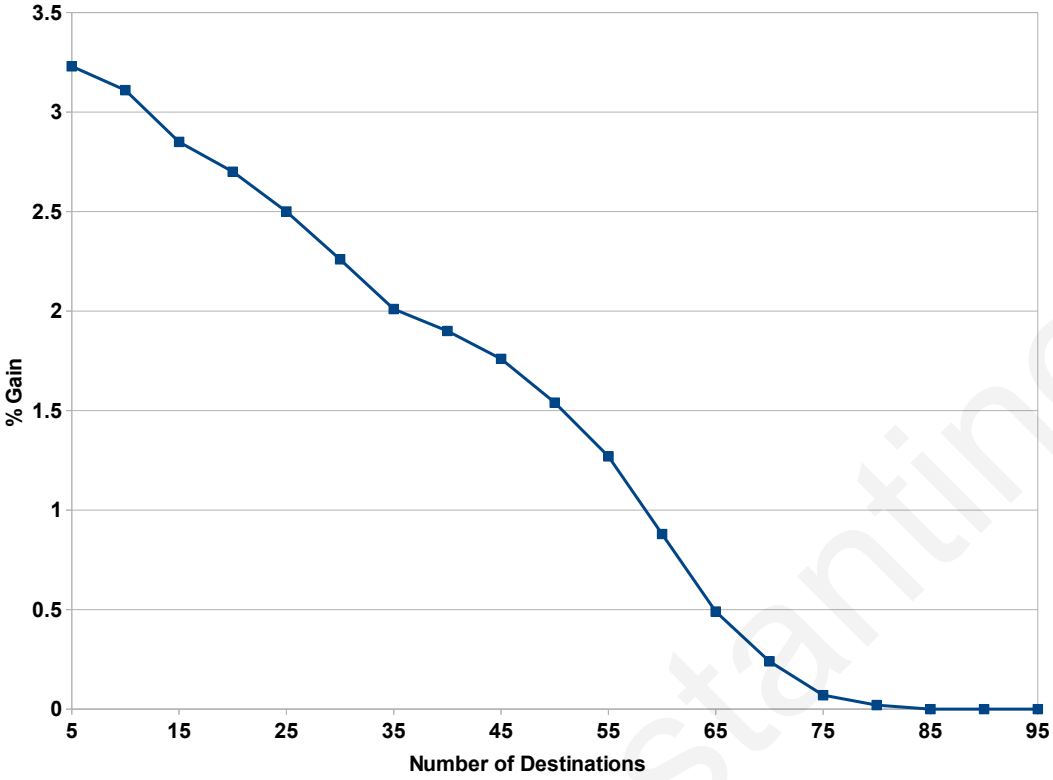


Figure 3.15: Performance gain of the SNH heuristic compared to the MPH approach: Regular graph with 100 nodes and 180 links.

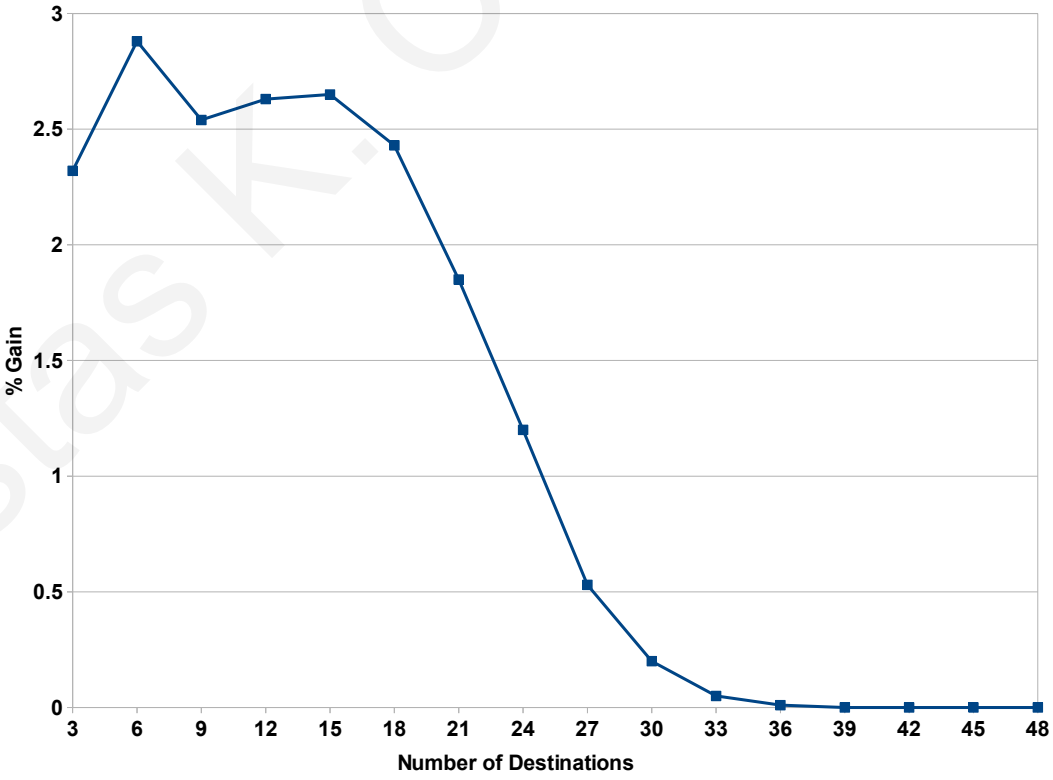


Figure 3.16: Performance gain of the SNH heuristic compared to the MPH approach: Small-world graph with 50 nodes and 112 links.

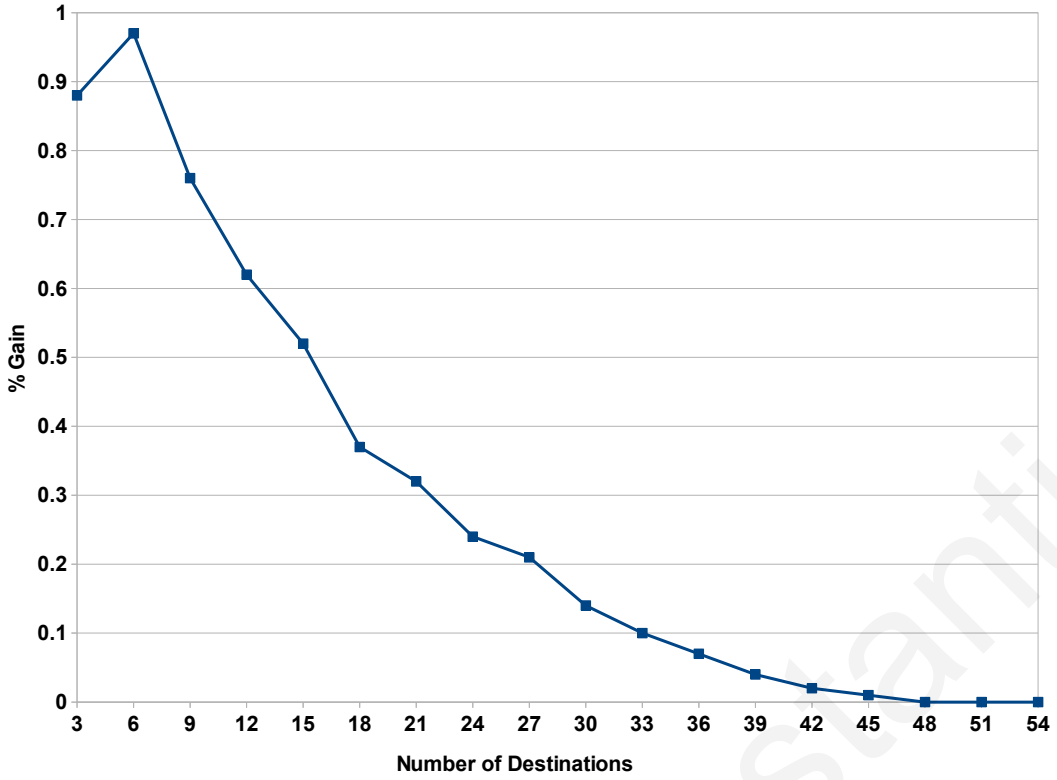


Figure 3.17: Performance gain of the SNH heuristic compared to the MPH approach: Scale-free graph with 100 nodes and 180 links.

for relatively small multicast sessions. This is important, since this is the case that appears more in practical applications for mesh optical networks.

3.6 Kou's Heuristic Algorithm Used as Basis for SNH

Kou's heuristic algorithm [37] is another heuristic that can be used as basis for SNH. It consists of the following steps:
Input: An undirected distance graph $G(V, E)$ and a set of Steiner nodes $Z \subset V$.
Output: A Steiner tree T^* for G and Z.

1. Construct the complete undirected graph $G_1 = (V_1, E_1)$ from G and Z.
2. Find the minimal spanning tree, T_1 of G_1.
3. Construct the subgraph, G_Z, of G by replacing each edge in T_1 by its corresponding shortest path in G.
4. Find the minimal spanning tree, T_Z, of G_Z.

5. Construct a Steiner tree, T^*, from T_Z by deleting edges in T_Z, if necessary, so that all the leaves in T^* are Steiner nodes.

The shortest paths between every pair of graph nodes in Kou's heuristic algorithm are calculated using Floyd-Warshall's all-pair shortest path algorithm [16, 17], and the minimal spanning tree can be calculated using Prim's algorithm [18].

SNH is compared with Kou's heuristic algorithm, via simulations similar to the comparison between MPH and SNH in Section 3.5. The results are presented in Figures 3.18-3.21. Simulations have shown that once again SNH outperforms Kou's algorithm, that was used as its basis.

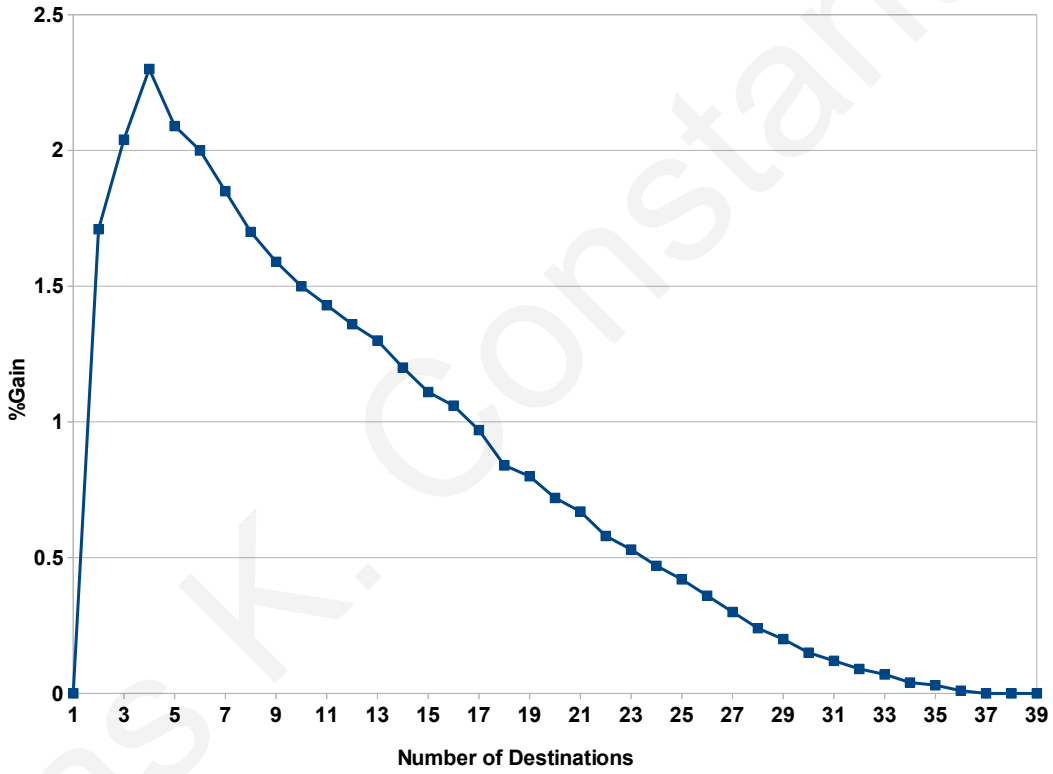


Figure 3.18: Performance gain of the SNH heuristic compared to Kou's heuristic algorithm: Graph with 40 nodes, 100 links, $d_{nom} \leq 5$.

In the following section, a generalized version of SNH is presented.

3.7 Generalized SNH

In the explanation of the SNH heuristic given in Section 3.3, Step 2 is repeated for every vertex not belonging to the current tree. This step can be generalized, leading to the *Generalized SNH* (GSNH), as follows:

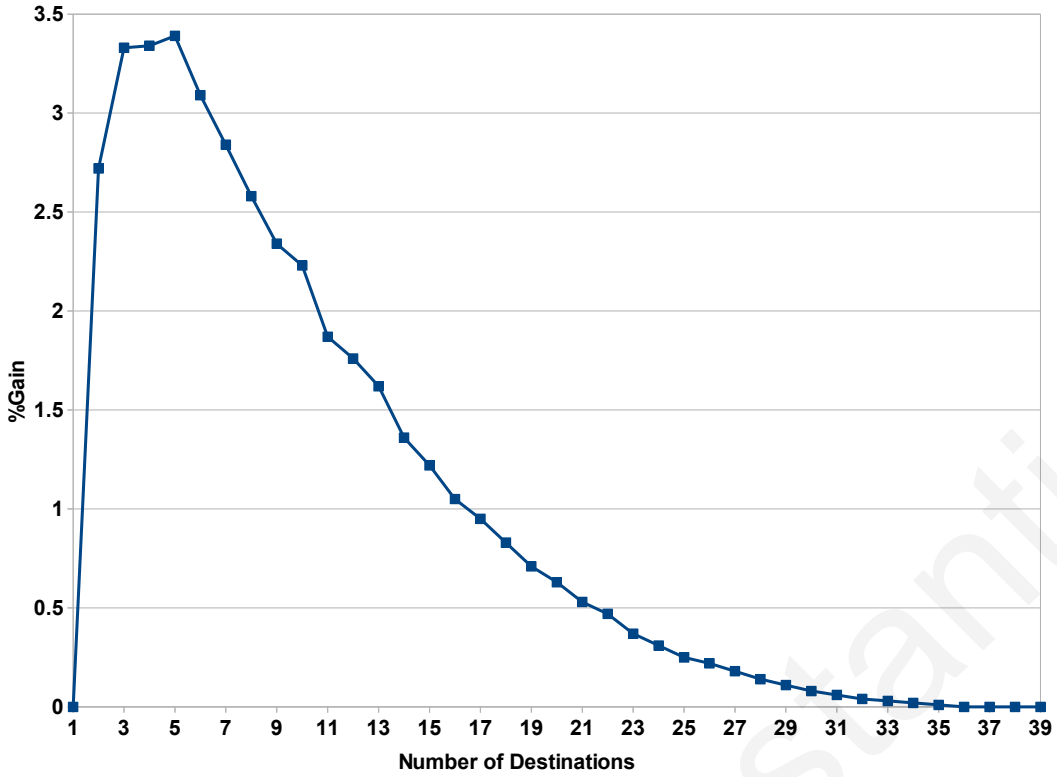


Figure 3.19: Performance gain of the SNH heuristic compared to Kou's heuristic algorithm: Graph with 40 nodes, 100 links, $d_{nom} \leq 40$.

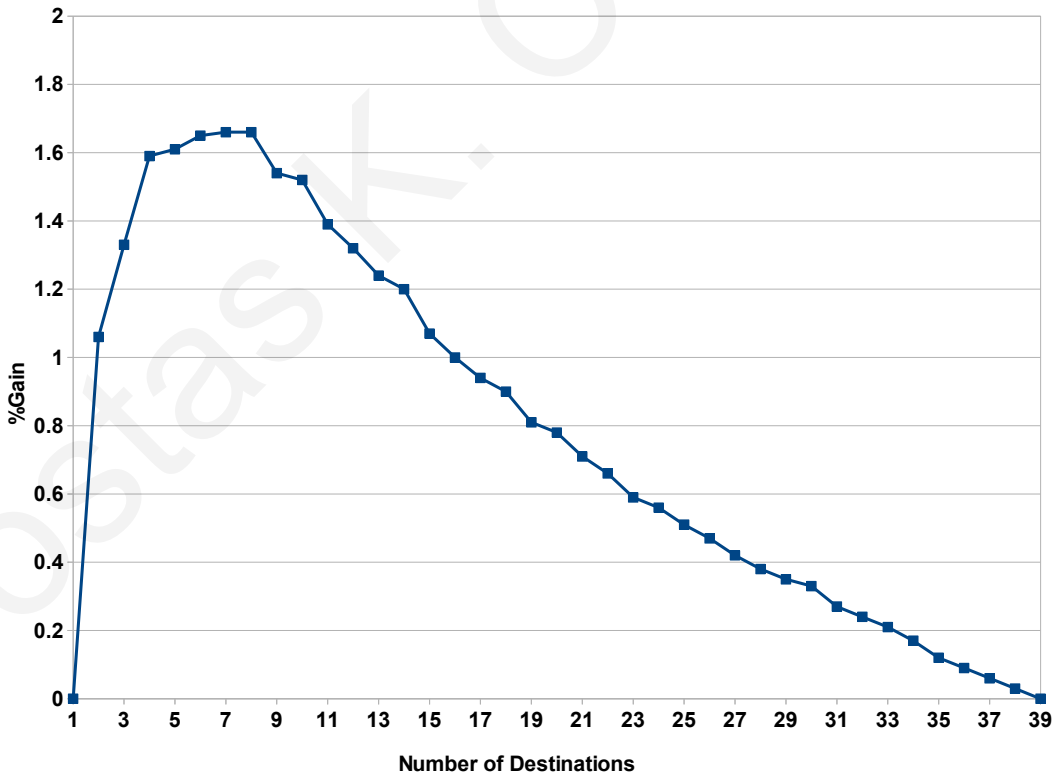


Figure 3.20: Performance gain of the SNH heuristic compared to Kou's heuristic algorithm: Graph with 40 nodes, 150 links, $d_{nom} \leq 5$.

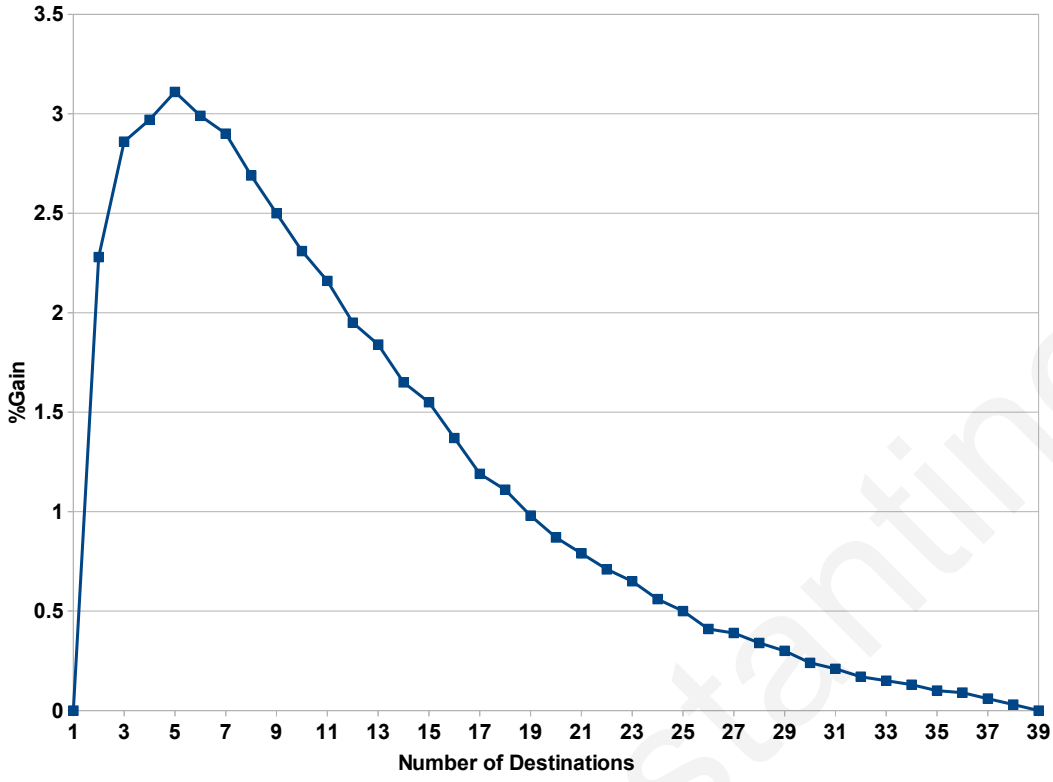


Figure 3.21: Performance gain of the SNH heuristic compared to Kou's heuristic algorithm: Graph with 40 nodes, 150 links, $d_{nom} \leq 40$.

1. Calculate Steiner Tree $T^*(V^*, E^*)$ using MPH and find its cost, called C.
2. For every possible set i of vertices $v \in \{V - V^*\}$, consisting from 1 up to Q vertices, do the following:
   (a) $\{Z'\} \leftarrow \{Z\} + v_i$.
   (b) Calculate Steiner Tree $T_i(V_i, E_i)$ on $G(V, E)$ for set $\{Z'\}$, using MPH, and find its cost, c_i.
   (c) $\{Z'\} \leftarrow \{Z\}$.
3. Find v_j and $T_j : c_j = min_i\{c_i\}$.
4. If $c_j \geq C$, the solution is $T^*(V^*, E^*)$, STOP.
   (a) If $c_j < C$, replace $\{Z\}$ with $\{Z\} + v_j$, C with c_j, and $T^*(V^*, E^*)$ with $T_j(V_j, E_j)$.
   (b) If set $\{V - V^*\} \neq 0$, return to Step 2. Else STOP.

In this case, Q is determined before the application of the algorithm. For example, if $Q = 2$, the check will be performed for all possible single vertices and all possible

pairs of vertices, that belong to $\{V - V^*\}$. GSNH will have better performance compared to SNH, but higher complexity. Its complexity is $O((^nP_1 +^n P_2 + ... +^n P_Q))k^2n^2)$, where ^aP_b are the permutations of a elements from the total number of b elements of a set. Obviously, as Q increases, the performance of the heuristic is improved, but the time-complexity is increased.

It must be noted that although GSNH has theoretical interest, in practice it will give only a slight improvement in the cost of the multicast tree compared to SNH, and the tradeoff will be high time-complexity and, consequently, unacceptable running time especially for dynamically provisioned multicast connections.

3.8 Conclusion

A new heuristic algorithm, called *Steiner Node Heuristic* (SNH) was presented for solving the Steiner Tree Problem in graphs. The proposed heuristic has polynomial time complexity of order $O(k^2n^3)$, where n is the number of network nodes and k is the subset of them to be connected. For the special cases of $k = 2$ and $k = n$, it has the same (optimal) results as the widely-used MPH algorithm. For all other cases, its results are, in general, better.

The major importance of the new heuristic algorithm is that it is a general method for improving the solution obtained by a certain multicast routing heuristic algorithm. In this thesis, MPH and Kou's heuristic were used as the basis for the proposed algorithm. However, any other appropriate algorithm can be used, in order to have improved results and this approach can also be applied to multicasting in other categories of networks, apart from optical networks. The tradeoff for the performance gain achieved by the proposed algorithm, is an acceptable increase of the time-complexity.

Chapter 4

Sparse-Splitting Undirected-Graph Networks: Heuristic Algorithms for Multicast Routing and Efficient Placement of Multicast-Capable Nodes

4.1 Introduction

Multicasting in optical networks is based on the calculation of light-trees, utilizing optical splitters in the network nodes [5]. The information through an optical fiber is sent using a specific wavelength for each multicast request. If no wavelength converters exist at the network nodes, the same wavelength must be used in all fibers [2]. In this section it is assumed that each network node has full wavelength conversion, therefore the information can be sent in different wavelengths through the light-tree fibers.

An optical splitter [5] is a passive device that splits the input optical signal into multiple identical output signals. The nodes that have this splitting capability are called *Multicast-Capable* (MC) nodes, otherwise they are called *Multicast-Incapable* (MI) nodes. Assuming an optical splitter with n outputs, optical splitting reduces the optical power at each splitter output port to $(1/n)^{th}$ of the input power. Thus, since the optical signal power at the photoreceiver needs to be higher than a threshold to be detected, an optical network with a large number of multicast-capable nodes may cause the signal to experience a significant power loss, limiting the reach of

the optical signal. To combat this effect, a larger number of optical amplifiers will be required in the network, further adding amplified spontaneous emission (ASE) noise in the system and requiring a worst-case network engineering and design [76]. To limit the impact of optical splitters in the network we can only place them at some of the network nodes (multicast-capable (MC) nodes), resulting in a *sparse-splitting* network [**??**, 77]. As previously discussed, multicast-capable nodes can also be opaque nodes where the signal is electrically duplicated at the opaque switch before it is sent out of the node on the same (or different) wavelength. This is an architecture that is also exploited in some of the heuristic algorithms presented in this section.

This section focuses on the following open problems that are related to this category of networks: development of efficient multicast routing heuristic algorithms under the sparse-splitting constraint, efficient allocation of the multicast-capable nodes, and calculation of the optimal percentage of network nodes that must be multicast capable. It is assumed that the splitting factor of MC nodes is not limited i.e., an MC node is able to split (duplicate) the incoming signal as many times as required.

The problem of multicast routing in sparse-splitting networks is NP-hard, since this is a special case of the Steiner problem in graphs which is NP-hard [12]. The problem of efficient allocation of MC nodes in the network was also proven to be NP-complete [52]. Therefore, polynomial-time heuristics are used in practice for both problems.

The MI nodes of the network may be *Drop-and-Continue* (DaC) or *Drop-or-Continue* (DoC) nodes (Figures 2.4 and 2.3 respectively). A DaC node can transmit the signal to the next node and can receive it locally as well, while a DoC node can either transmit the signal to the next node or receive it locally. Since networks of both cases are referred to in the literature (e.g., [19] for DaC case and [53] for DoC), this section deals with both DaC and DoC networks. It is noted that although the placement of DaC nodes in the network leads to cost-efficient routing, it is more costly compared to a DoC solution. Thus, the study of both cases will give a comparison between them, i.e., for the same performance of two networks, one with DaC and the other with DoC, it can be calculated how many MC nodes are needed for each one, and comparing the cost of MC nodes and DaC nodes, the network designer can apply the most economical solution.

4.2 Multicast Routing in Networks with Sparse-Splitting Capability: Existing Heuristic Algorithms

There are a number of approaches in the literature on the problem of multicast routing in sparse-splitting mesh optical networks that can be found in [20, 21, 19, 53, 46, 54, 55, 56], as well other sources. As it is not possible to compare the proposed solution with all the heuristics in the literature, a few of the heuristics were identified that produce the most efficient solutions amongst the existing works. These heuristics are presented in this section and will be the ones that will be compared with the proposed heuristic algorithm for multicast routing in sparse-splitting networks.

4.2.1 Member-Only

The Member-Only approach was created for DaC networks and this heuristic is presented in [19]. The basic idea of the Member-Only technique is similar to that of the shortest-path heuristic for constructing a near-minimum multicast tree, with the main feature being that, as long as an MI node y on a tree is a non-leaf node, other members will not join the tree at y.

4.2.2 On-Tree MC Node First (OTMCF), Nearest MC Node First (NMCF)

The On-Tree MC Node First (OTMCF) and Nearest MC Node First (NMCF) heuristics algorithms that were created for DoC networks are presented in [20]. For both heuristics the following preprocessing takes place.

Initially, a multicast-capable (MC) network M_G is derived from the original network G as follows:

1. All the MC nodes in network graph G are included as its vertex set, say W.
2. If there is a path between two MC nodes in G, these two MC nodes are connected in M_G.
3. The link cost from node i to node j in graph M_G is set to the cost of the minimum-weight path from node i to node j in G, for all $i, j \in W$.

Subsequently, the *Auxiliary Network Transformation (ANT)* is performed as follows:

1. The MC network graph M_G of the original network graph G is determined.

2. A Steiner tree heuristic is applied to graph M_G to generate a minimum-cost tree T_R for multicast session R.

3. The resulting light-tree T is obtained by substituting each link, say (i, j), in T_R with the corresponding minimum-cost path from node i to node j in G.

Finally, the multicast tree can be derived by the following two heuristic algorithms that utilize *ANT* and were created for DoC networks:

On-Tree MC node First (OTMCF):

This approach attempts to minimize the cost of an MC tree. An MC tree is constructed by first including all the MC nodes to which group members are directly connected, and then expanded as follows: the MI nodes to which the remaining members are directly connected join the tree through the nearest on-tree MC nodes.

Nearest MC node First (NMCF):

This approach attempts to minimize the costs of MI nodes which join the MC tree. The set of MC nodes used to construct an MC tree consists of all the MC nodes directly connecting to the destinations of the group and the MC nodes which are the nearest to the MI nodes connecting to the remaining destinations. NMCF expands on-tree MC nodes in such a way that each MC node nearest to each MI node in the group must also be on-tree.

4.2.3 Cost-Effective Multicasting Using Splitters (MUS)

This heuristic is presented in [21] and was also created for DoC networks. According to a given graph G, an auxiliary graph $G'(M, E')$ is created that consists of MC nodes only (set M), where the cost of a link in E' is the cost of the corresponding shortest path in the original graph. An initial Steiner tree T is then created that consists of the source node and the destination nodes that are MC. Let M_{onTree} be the set of MC destinations, and M_{remain} the set of the rest of the destinations in a given session. The latter are MI nodes and are not as of yet connected to the tree. The procedure presented next is then followed for the connection of the M_{remain} nodes on the initial tree T.

1. For each node in M_{remain}, find the MC node in M_{onTree} that it can connect to via the shortest path.

2. Select the shortest path among the corresponding paths and add it to tree T.

3. If an MC node exists on the shortest path, then the MC node is added to M_{onTree}.

4. The MI node connected to T is excluded from the set M_{remain}.

5. Repeat the above steps until M_{remain} is an empty set.

MUS is similar to NMCF with two basic improvements that make it more efficient. The first one is that, after the addition of a path that connects a MI destination to the tree, the unconnected MI destinations are checked whether they can be connected efficiently (i.e., with low cost) through any MC nodes belonging to this path (whereas NMCF ignores these nodes). The second one is that, using MUS, the MI destinations are connected in increasing order according to the cost of the shortest path between them and the tree, while in NMCF this is not taken into account.

For DoC networks, OTMCF and NMCF outperform Member-Only, and MUS outperforms OTMCF and NMCF, according to simulations in [20] and [21] respectively. Other sparse-splitting multicast routing heuristic algorithms can be found in the literature (e.g., [46, 53-56]), but the ones discussed in this section are shown to have the best performance results.

The proposed algorithms for multicast routing in sparse-splitting networks are presented in the following section.

4.3 Multicast Routing in Networks with Sparse-Splitting Capability: Proposed Heuristic Algorithms

4.3.1 *MPH** Heuristic Algorithm

A heuristic algorithm that is used extensively in the literature for multicast routing is the Minimum Path Heuristic (MPH) [15]. It consists of the following steps (the shortest paths (and the corresponding distances) between every pair of graph nodes are calculated using Floyd-Warshall's all-pair shortest path algorithm [16, 17]):

1. Tree T_1 consists only of the source s. Set $V_1 = \{s\}$. $i = 1$.

2. Determine a node u in $\{D - V_1\}$ (D is the destination set), that is closest to T_i. Construct a tree T_{i+1} by adding the minimum cost path joining u and T_i. Add the nodes of this path in set $\{V_1\}$. $i = i + 1$.

3. Repeat Step 2. STOP when all nodes of D are connected to the tree.

The MPH heuristic algorithm can be applied only in networks with full-splitting capability, since the minimum cost path of Step 2 can originate from any network node. In order to be applicable to sparse-splitting networks without DaC nodes, this path must originate either from the source or from a multicast-capable node that belongs to the current tree. In the case that all multicast-incapable nodes are DaCs, the path can also originate from destinations that are part of the current tree and have outdegree equal to zero. The modified heuristic algorithm, called *MPH**, that works for sparse-splitting networks as well, consists of the following steps:

1. Tree T_1 consists only of the source s. Set $V_1 = \{s\}$. $i = 1$.

2. (i) *Non-DaC networks:* Determine a node u in $\{D - V_1\}$, that is closest to T_i, which is connected through a path originating either from the source, or from a multicast-capable node that belongs to T_i.
   (ii) *DaC networks:* Determine a node u in $\{D - V_1\}$, that is closest to T_i, which is connected through a path originating either from the source, or from a multicast-capable node that belongs to T_i, or from a destination that belongs to T_i and has outdegree equal to zero.

3. Construct a tree T_{i+1} by adding the minimum-cost path joining u and T_i. Add the nodes of this path in set $\{V_1\}$. $i = i + 1$.

4. Repeat Step 2. STOP when all nodes of D are connected to the tree.

The *MPH** heuristic algorithm has the same complexity as MPH, i.e., $O(kn^2)$ [15], where n is the number of nodes of the graph and k is the number of nodes that must be connected.

Although, in general, the *MPH** heuristic algorithm is efficient, Figure 4.1 shows an example demonstrating that it does not perform well in all cases. In this example, node S is the source of the multicast request, nodes d_1, d_2 are the destinations, and node b is a multicast-capable node. In the first network all nodes are DaC nodes, while in the second all are non-DaC nodes. For both cases the *MPH** heuristic finds paths $S-a-d_1$ and $S-a-d_2$, with a total tree cost equal to 40. The optimal solution for both cases are paths $S-a-b-d_1$ and $b-d_2$, with a tree cost equal to 33 for the first case and 22 for the second. Therefore, an algorithm that will have the ability to use

MC node b for the calculation of the two paths, will calculate the optimal solution for both cases. This can be achieved if node b is added in the destination set. A new heuristic algorithm that was created to take advantage of the MC nodes, is presented in the following section.

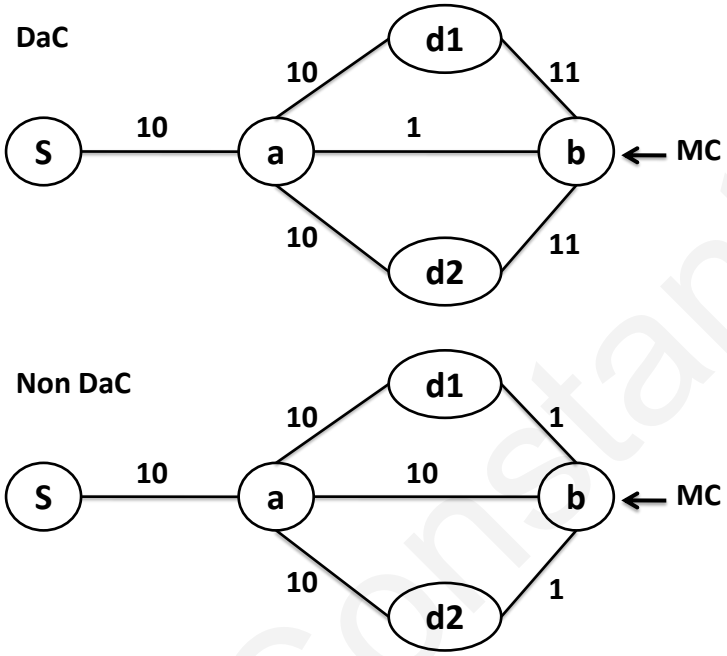


Figure 4.1: Example demonstrating drawbacks of the MPH^* heuristic algorithm.

4.3.2 Sparse-Splitting Multicast Routing Heuristic Algorithm

The *Sparse-Splitting Multicast Routing Heuristic (SSMRH)* consists of the following steps (D is the destination set):

1. Calculate the multicast tree, using the MPH^* heuristic algorithm.
2. Calculate the cost c of the resulting tree.
3. For each multicast-capable node MC_i not belonging to the current tree:
   (a) Add it to D.
   (b) Calculate the multicast tree, using MPH^*.
   (c) Calculate the cost c_i of the resulting tree.
   (d) Remove MC_i from D.

4. Find c_i^{min}. If $c_i^{min} \geq c$, go to Step 5. Else,

   (a) $c = c_i^{min}$.

   (b) Add MC_i permanently to D and return to Step 3.

5. Calculate the multicast tree for the new D, using MPH^*.

The SSMRH heuristic algorithm works as follows: the MC nodes that are not part of the multicast tree are added and removed from the destination set and the new tree is calculated every time using MPH^*. The MC node that gives the tree with the least cost is added permanently into the destination set. The procedure is repeated until no further cost reduction can be succeeded. It can be easily verified that for the networks of Figure 4.1, the proposed heuristic gives the optimal solution. It must be noted that, for every case, SSMRH gives a tree at least as good as the one derived from MPH^*, since it is a generalization of the latter.

It is important to note that even though Step 3 of SSMRH can find all MC nodes that give a tree with less cost if added into D, Step 3 is not executed only once and all MC nodes that give a tree with less cost are added permanently in D. The reason for doing this is explained using Figure 4.2. Here, S is the source node, $d_1 - d_4$ are the destination nodes, and b, c, y, z are the MC nodes. If MPH^* is used, the resulting tree is $S - a - d_1$, $S - a - d_2$, $S - x - d_3$, and $S - x - d_4$. After the execution of Step 3 in SSMRH, it is found that the addition of each one of the nodes b, c, y, z will give a tree with less cost, with b giving the best improvement. If node b is added, destinations d_1 and d_2 come closer to the source, and the use of MC node c as well, will not give any improvement (the opposite will happen), since this node "serves" the same destinations as MC node b. Therefore, it is not correct to add in D *all* MC nodes that Step 3 finds. After the addition of b, Step 3 must be repeated, to find the next MC node to be added in D, node y in this case, and in the third iteration the procedure stops, since no further improvement can be achieved.

SSMRH was presented using MPH^* as its basis. Other sparse-splitting multicast routing heuristic algorithms can be used as well, for example OTMCF, NMCF, or MUS. If algorithm X is used as basis, SSMRH will outperform it, since it starts from the tree derived by X and recursively tries to improve the initial solution. Therefore, in the worst case, it will find a tree with the same cost as the one found by X. On the average, the resulting trees obtained by SSMRH will have less cost compared to the

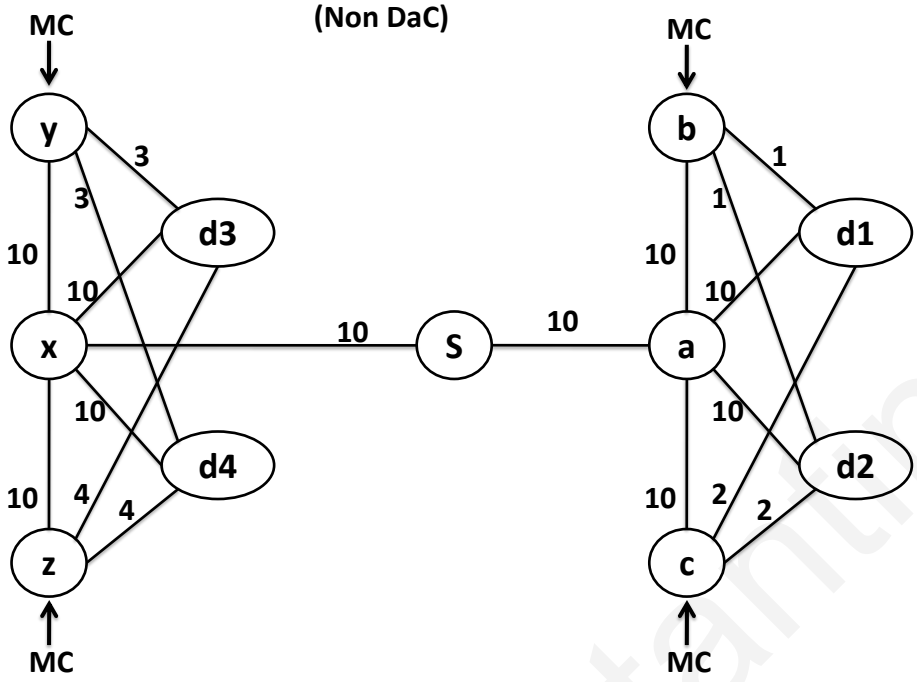


Figure 4.2: Example of the *SSMRH* heuristic algorithm.

results found by technique *X*.

An example where the SSMRH heuristic algorithm has improved performance compared to the algorithm used as its basis, is shown in Figures 4.3-4.7. Here, the network is considered as DoC, s is the source node, d_1, d_2, d_3 are the destinations, and square-shaped nodes are the MC nodes.

It can easily be verified that for every case, SSMRH is more efficient compared to the algorithm used as its basis. The lowest cost tree is obtained by SSMRH using either *MPH**, NMCF, or MUS. For these three cases, the tree has cost equal to 132.

Complexity of SSMRH: The complexity of the proposed heuristic for a network that has x MC nodes, with an algorithm having complexity $O(y)$ used as basis, is derived as follows:

- Complexity of Step 1: $O(y)$
- Complexity of Step 3: $O(xy)$
- Step 3 is repeated at most x times

Therefore SSMRH has complexity $O(y+x^2y) = O(y(1+x^2)) = O(yx^2)$. For example, its complexity is $O(kn^2x^2)$, if *MPH** ($O(kn^2)$) is used as its basis (n is the number of the network nodes and k is the number of the nodes that must be connected).

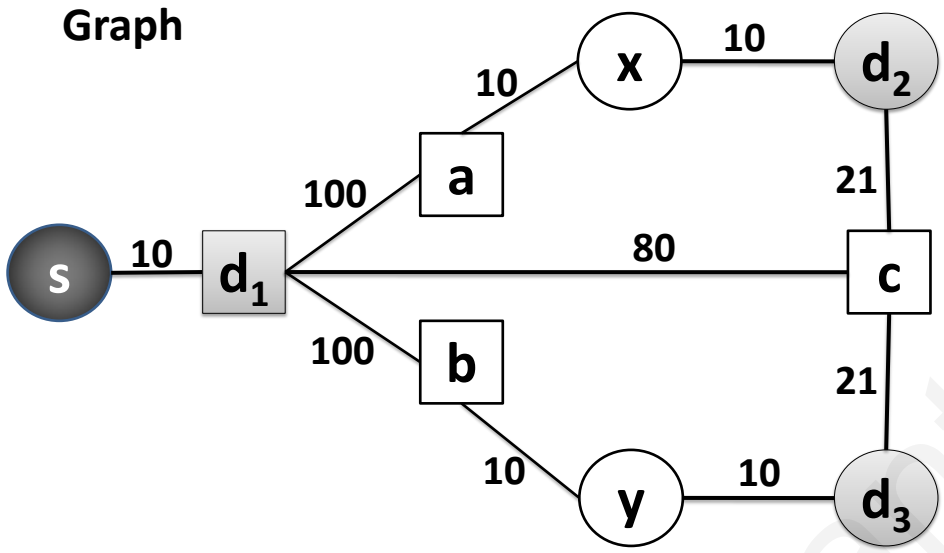


Figure 4.3: Network graph used for explanation and comparison of existing and proposed heuristic algorithms.

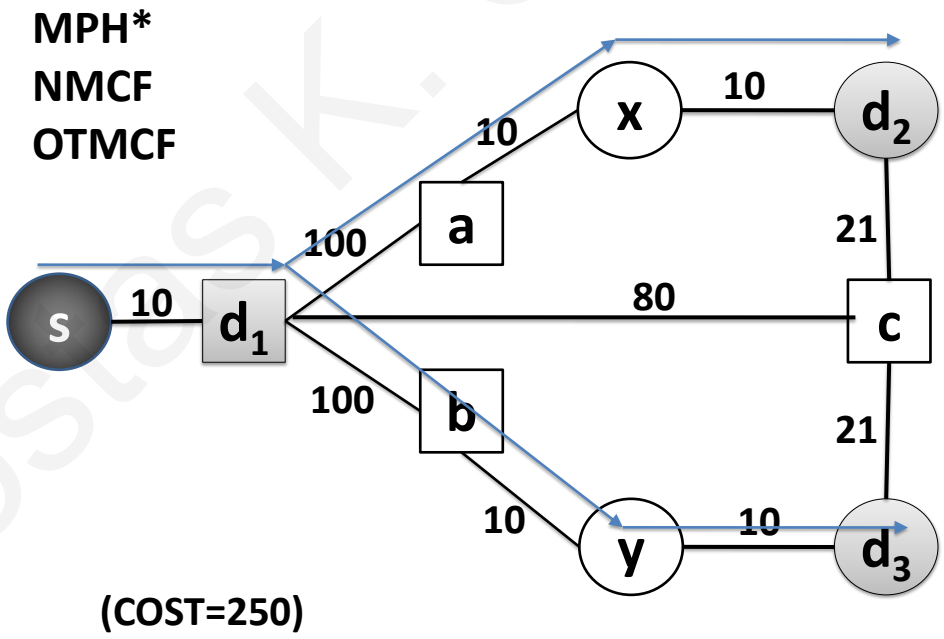


Figure 4.4: Tree obtained by *MPH**, NMCF, and OTMCF.

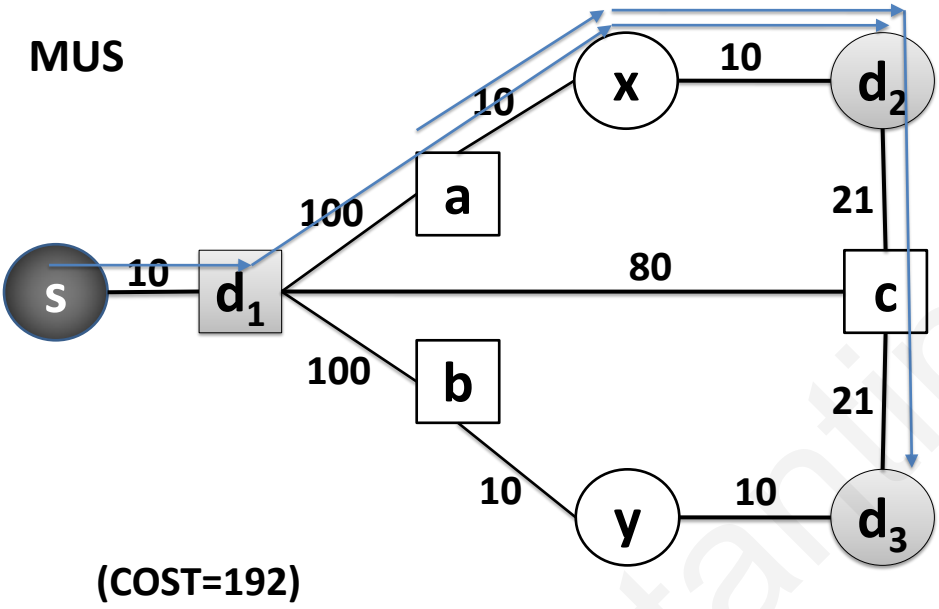


Figure 4.5: Tree obtained by MUS.

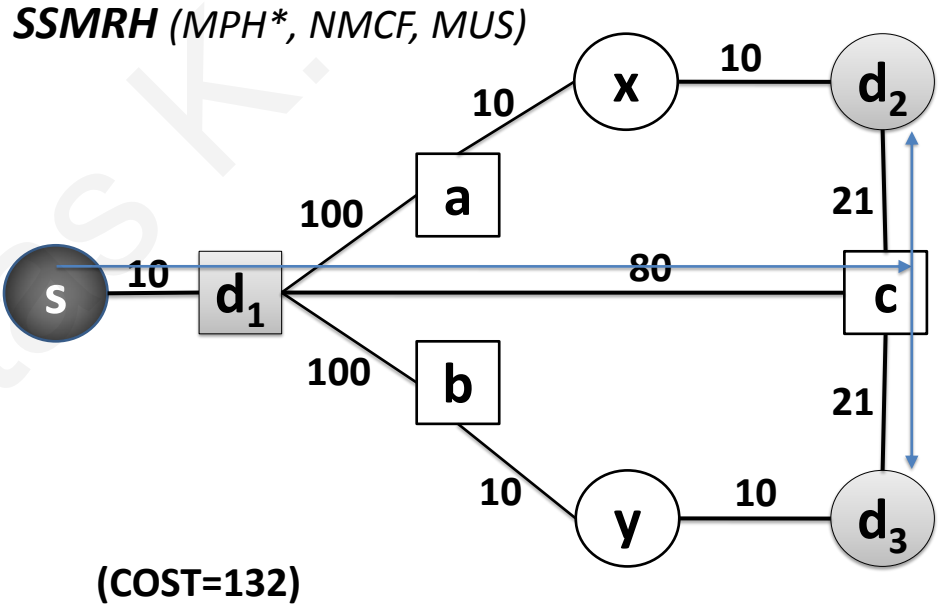


Figure 4.6: Tree obtained by SSMRH, using *MPH**, NMCF, or MUS as basis.

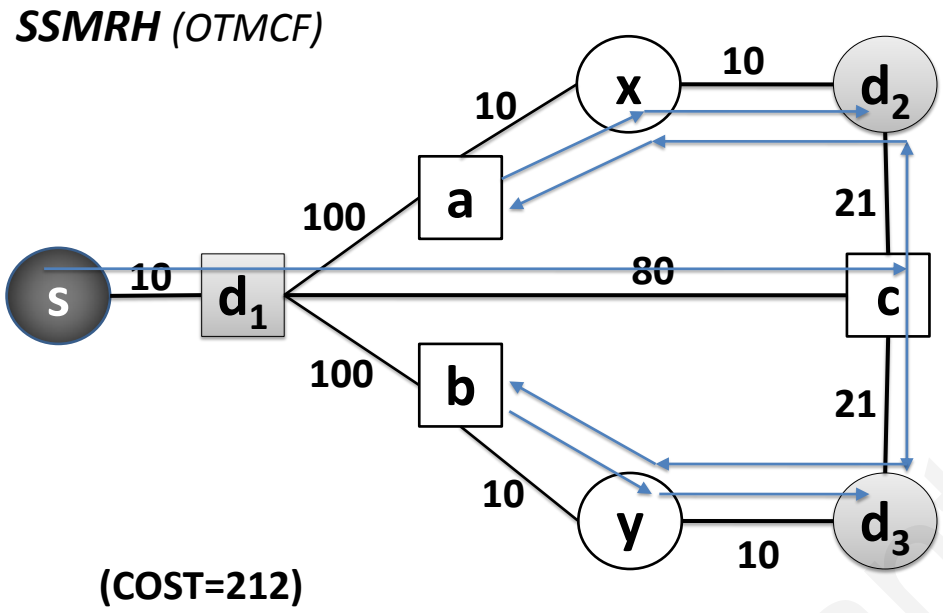


Figure 4.7: Tree obtained by SSMRH, using OTMCF as basis.

In the following section, some important existing heuristic algorithms for the allocation of optical splitters are presented, followed by new proposed techniques for the placement of the optical splitters in sparse-splitting networks.

4.4 Allocation of Splitters: Existing Heuristic Algorithms

4.4.1 k-Maximum Degree Method

The idea of this method [57] is that a node with a large number of neighboring nodes (high outdegree) is more likely to become a branch node of a multicast tree. Hence, placing a splitter on that node is expected to be effective in reducing the wavelength resource usage of the multicast connections. Therefore, this method sorts the nodes in the network in descending order according to their degree and selects the first k nodes to place the splitters. The computation time of this method includes the time for checking the degree of each of the nodes and the time for sorting the nodes according to their degree.

4.4.2 k-Maximum WR (Wavelength Reduction) Method

This method works as follows [57]: if the placement of a splitter at a node yields more reduction on the wavelength resource usage, it is more beneficial to place a splitter at that node.

Let s_i be the shortest path spanning tree rooted at node i. Let $h(s_i)$ denote the wavelength resource usage of s_i when no splitter is placed in the network. Note that when no splitter is placed in the network, each multicast connection is realized by multiple lightpaths. The wavelength resource usage $h(s_i)$ can be calculated by traversing the shortest path spanning tree s_i in a depth-first manner.

Let h be the total wavelength usage of the set of all shortest path spanning trees rooted at each of the nodes when no splitter is placed in the network. Then, $h = \sum_{i \in V} h(s_i)$.

Let $f(i, s_j)$ be the wavelength resource usage of s_j when a splitter is placed at node i. Let $f(i)$ denote the wavelength resource usage of the set of all shortest path spanning trees rooted at each of the nodes when a splitter is placed at node i. Then, $f(i) = \sum_{j \in V} f(i, s_j)$.

Let $R(i)$ be the amount of reduction in wavelength resource usage when a splitter is placed at node i compared to the case where no splitter is placed in the network. Then, $R(i) = h - f(i)$.

The k-maximum WR method sorts the values of $R(i)$, $i = 1, 2, \ldots, |V|$, in descending order and selects the first k nodes to place the splitters.

4.4.3 Most-Saturated Node First (MSNF)

This heuristic is presented in [52]. The main idea behind this greedy heuristic is to allocate the MC nodes to those nodes in the network which are used the most by routed sessions as branching nodes. The MSNF heuristic starts with solving the multicast routing and wavelength assignment problem using some available heuristic. The cross-connects are then sorted in decreasing order of $|\Delta(t)|$, where $\Delta(t)$ denotes the set of trees requiring cross-connect t as a branching node. The fist K cross-connects in the sorted order become MC cross-connects and the rest become MI cross-connects. All sessions affected by this assignment are then deallocated and re-established taking into consideration now the real allocation of the MC nodes.

4.4.4 Simulated Annealing

This heuristic is presented in [52] as well. It is a general optimization technique motivated by a physical process related to the cooling of a material in a heat bath. It employs a variation of the well-known search technique called neighborhood search (also known as local search). It starts with an initial solution created by randomly assigning the splitters and deallocating all routed sessions not supported by the initial assignment. The initial state (called *temperature*), τ, is assigned. For a number of iterations $Iter^{max}$, a new solution in the neighborhood is selected. If the new solution is better than the current solution, the new solution replaces the current solution. Otherwise, depending on the current solution, the new solution, and the current value of τ, a function is used that decides whether the new solution should replace the current solution. The probability of accepting the new solution is given by $P^{Accept} = e^{-\frac{\delta}{\tau}}$, where δ is the difference in terms of the objective function value between the two solutions. After $Iter^{max}$, the current value of τ is decreased, thus making it difficult to accept bad solutions as the search process progresses in time. The reduction of τ is controlled by a geometric reduction function $\tau_{k+1} = a \cdot \tau_k$, where a is the reduction factor.

Other splitter allocation heuristics can be found in [58-61], as well other sources. Section 4.5 that follows presents the proposed splitter allocation techniques.

4.5 Allocation of Splitters: Proposed Heuristic Algorithms

Since in sparse-splitting networks the number of MC nodes is limited, the performance of the sparse-splitting multicast routing heuristic algorithms depends on the position of these nodes. Three proposed heuristic algorithms for efficient placement of MC nodes are presented: Decreased Number of Branches (DNB), Decreased Number of Children-Destinations (DNCD), and Least Used Removed First (LURF). The idea behind them is that the optical splitters must be allocated at the nodes that will exploit them the most. Three different heuristics were developed and all of them are presented and compared through simulations. For a large number of multicast requests randomly created, DNB counts the total number of branches each network node had to serve, and allocates the optical splitters to the nodes that had

the largest number of branches, while DNCD allocates the optical splitters to the nodes that serve the largest number of destinations. On the other hand, the opposite procedure is followed for LURF. All the network nodes are assumed to be equipped with an optical splitter, and the ones that use them the least are converted to multicast-incapable nodes; the procedure stops according to the available number of optical splitters. All three heuristics that are described below use the following definitions:

- $\{MC\}$: set of MC network nodes.
- M: number of network nodes.
- P: number of nodes that can be MC.
- I: number of iterations of the heuristics.

4.5.1 DNB Heuristic Algorithm

The *Decreased Number of Branches (DNB)* heuristic algorithm consists of the following steps:

1. Assume that every network node is an MC node.
2. Repeat for all K possible multicast group sizes ($1 \leq k \leq K$):
   - Repeat I times ($1 \leq i \leq I$):
     (a) Select randomly the source and destinations, according to the multicast group size.
     (b) Calculate the multicast tree using a multicast routing heuristic algorithm.
     (c) For each tree node x that has outdegree > 1, calculate the number ($N_x^{k,i}$) of its branches.
3. For each network node x that has outdegree > 1, calculate $C_x = \sum_{k=1}^{M} \sum_{i=1}^{I} N_x^{k,i}$.
4. Arrange network nodes in decreasing order according to their C values.
5. If the network can have P nodes with splitting capability, select the P ones with the highest C values.

4.5.2 DNCD Heuristic Algorithm

The *Decreased Number of Children-Destinations (DNCD)* heuristic algorithm consists of the following steps:

1. Assume that every network node is an MC node.
2. Repeat for all K possible multicast group sizes ($1 \leq k \leq K$):
   - Repeat I times ($1 \leq i \leq I$):
     (a) Select randomly the source and destinations, according to the multicast group size.
     (b) Calculate the multicast tree using a multicast routing heuristic algorithm.
     (c) For each tree node x that has outdegree > 1, calculate the number ($N_x^{k,i}$) of its descendants that are destinations.
3. For each network node x that has outdegree > 1, calculate $C_x = \sum_{k=1}^{K} \sum_{i=1}^{I} N_x^{k,i}$.
4. Arrange network nodes in decreasing order according to their C values.
5. If the network can have P nodes with splitting capability, select the P ones with the highest C values.

4.5.3 LURF Heuristic Algorithm

The *Least Used Removed First (LURF)* heuristic algorithm consists of the following steps:

1. Add all network nodes in $\{MC\}$.
2. Repeat $M - P$ times:
   - Find the minimum spanning tree and calculate its cost c.
   - Repeat for each network node x that is an MC node:
     (a) Remove it temporarily from $\{MC\}$.
     (b) Find the minimum spanning tree and calculate its cost c_x.
     (c) Calculate $A_x = c_x - c$.
     (d) Add it in $\{MC\}$.
   - Find node x that has A_x^{min} and remove it permanently from $\{MC\}$.

Explanation of the Heuristics

Heuristic algorithms DNB and DNCD work as follows: the Steiner tree for a random multicast session is calculated and, for each network node that is a branch node on the tree (i.e., that has outdegree > 1), the number of branches, for DNB, and the number of destinations that are descendants of it, for DNCD, is counted. The procedure is repeated I times for each possible multicast group size. If the graph has M nodes for example, the procedure is repeated $I \cdot (M-1)$ times. The total number of branches for, DNB, that each tree branch node had, and the total number of destinations, for DNCD, that each tree branch node had as descendants, over the $I \cdot (M-1)$ iterations, is calculated for each network node that was a branch point at least once. Then, the nodes are selected to have multicasting capability according to this number. The idea behind these heuristics is that a node has more value as an MC node the more branches, for DNB, and the more number of destinations, for DNCD, it serves. If the heuristics are executed for a large number of iterations (i.e., I is large), many more trees for different multicast group sizes will be calculated, thus making a more efficient selection of the MC nodes.

Heuristic LURF works as follows: All network nodes are considered to be MC, and the cost of the minimum spanning tree is calculated using a sparse-splitting routing heuristic algorithm (The source of the tree is selected arbitrarily). The multicasting capability is removed temporarily from a network node, and the minimum spanning tree cost is calculated again. The node that, if converted from MC to MI, gives the least cost increase, is removed permanently from the MC set. If the network consists of M nodes and P of them can be MC the procedure is repeated $M-P$ times, i.e., until $M-P$ nodes will be converted from MC to MI.

It must be stated that a different version of the LURF heuristic was also tested, where the average cost over all possible minimum spanning trees was calculated at Step 2, instead of just one with its source selected arbitrarily. Simulations showed that this heuristic did not give better results compared to the one stated here, though. Therefore, the one presented here was selected for comparison with the other allocation heuristic algorithms, since it is simple and with lower complexity.

Complexity of the proposed splitter allocation heuristic algorithms:

DNB, DNCD: $O(KIY)$, where $O(y)$ is the complexity of the multicast routing heuristic algorithm, K is the number of the possible multicast groups and I is the

number of iterations.

LURF: $O((M-P)(M+1)y) = O((M-P)My)$, where M is the number of network nodes, P in the number of MC nodes, and $O(y)$ is the complexity of the algorithm used for the calculation of the minimum spanning tree.

High complexity of the splitter allocation heuristic algorithms is acceptable (as long as it is polynomial in time), since the splitter allocation problem is a *network design problem*, and these algorithms will not be applied in real time.

4.6 Performance Evaluation

The performance of the proposed heuristics for efficient multicast routing in sparse-splitting networks, and for efficient placement of a limited number of MC nodes, was evaluated through simulations. The network graph was randomly created, it consisted of 50 nodes and 200 links, and it was undirected (i.e., every connection was bidirectional). Let the *nominal distance* (d^{ij}_{nom}) between two nodes i and j be defined as $d^{ij}_{nom} = |i - j|$. The constraint that every arc that was added in the graph had to connect nodes that satisfy $d^{ij}_{nom} \leq 5\ \forall i, j$, was used for the random graph creation. The reason is that the graph must simulate a real network, where nodes that are connected belong to the same neighborhood. Each link had cost randomly selected from 1 to 100. The simulation was repeated for various possible multicast group sizes, from $D = 5$ to $D = 25$ (D stands for the number of destinations), with a step equal to 5. The experiment was executed 5000 times for every multicast group size, while the source and destinations of the multicast connections were distributed uniformly across the network. The cost of the multicast tree was calculated using the existing heuristics OTMCF, NMCF, and MUS. Member-only was omitted since it underperforms the aforementioned. MPH^* and SSMRH were also used for the calculation of the multicast tree for every case. MPH^*, MUS, and OTMCF were exploited as the basis for SSMRH and the resulting heuristics were named SSMRH1, SSMRH2, and SSMRH3 respectively. NMCF was not used, since MUS is a generalization of it that performs at least as good for every case. The average cost over the 5000 iterations was extracted for each multicast group size. The procedure was repeated for the following splitter allocation heuristic algorithms: kmaxD, kmaxWR, DNB, DNCD, and LURF. All the above scenarios were repeated for both DoC and DaC networks. For the case of DaC networks, MUS was modified in order to be applicable to this

type of networks: the MI destinations could be connected to the tree either through the source, or through an MC node, or through an already connected MI destination x, as long as x was a leaf node. The other existing routing heuristic algorithms remained the same for DaC networks, since they cannot be modified in order to be more efficient for this category of networks.

The results of heuristics OTMCF, NMCF, and SSMRH3 for DaC networks are not presented, since their performance was very poor compared to the other algorithms, i.e. they gave trees with cost much higher compared to the cost obtained by MPH^*, MUS, SSMRH1, and SSMRH2.

The performance of the routing heuristic algorithms was averaged over all multicast group sizes used during the simulation ($D = 5$ to $D = 25$, with step equal to 5) (defined as $\overline{Cost}$). The number of MC nodes varied from 5 to 25 with a step equal to 5, i.e. from 10% to 50% of the total number of network nodes, since in sparse-splitting networks the MC nodes rarely exceed half the number of network nodes. The results of the simulations are given in Figures 4.8-4.17.

4.6.1 Comparison of the Multicast Routing Heuristic Algorithms

In Figures 4.8-4.17, the x-axis gives the number of MC nodes and the y-axis gives the $\overline{Cost}$. The description under each graph gives the heuristic algorithm that was used for MC node placement and whether the MI nodes of the network were DoC or DaC.

Analysis of the Results

From the simulation results, it is clear that the proposed heuristics outperform the existing ones (i.e., the average cost of the resulting trees is less), regardless of the selection of the splitter allocation method. They are more efficient for both cases of DoC and DaC networks. For the case of DoC networks, SSMRH2 (i.e., using MUS as its basis) is the most efficient. For the DaC case, SSMRH1 (i.e., using MPH^* as its basis) outperforms the rest of the heuristics. An example of the comparison between MPH^* and MUS is given in Figure 4.18. In this example it is shown that for the same network graph, MPH^* is more efficient for the case that the MI nodes are DaC, and MUS performs better for the DoC case.

Simulations have also shown that:

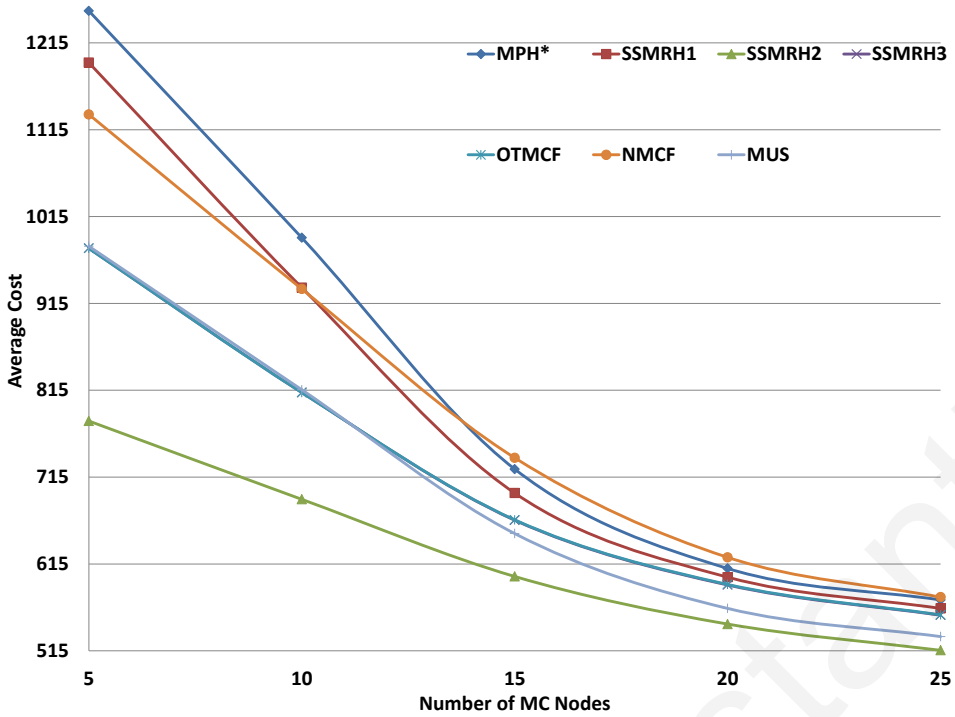


Figure 4.8: Comparison of the multicast routing heuristic algorithms using kmaxD for splitter allocation (DoC network).

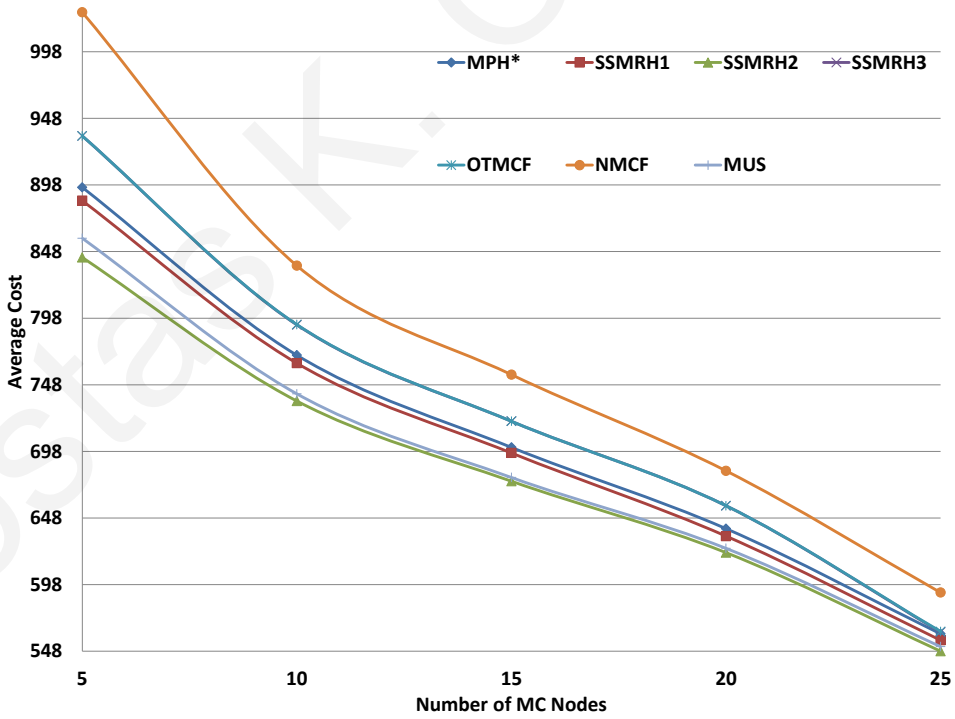


Figure 4.9: Comparison of the multicast routing heuristic algorithms using kmaxWR for splitter allocation (DoC network).

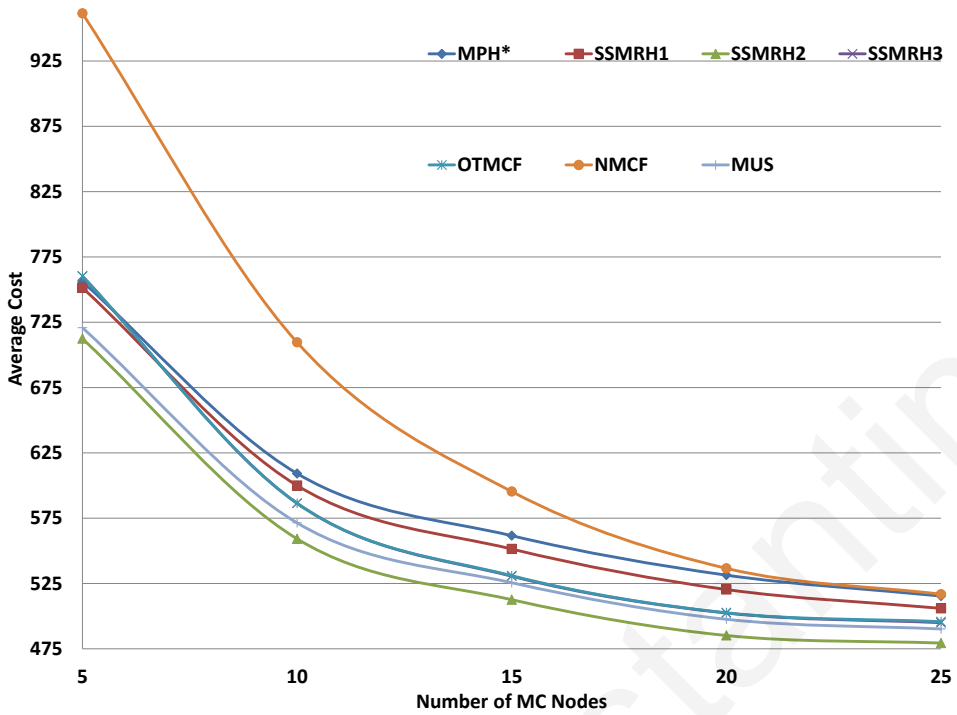


Figure 4.10: Comparison of the multicast routing heuristic algorithms using DNB for splitter allocation (DoC network).

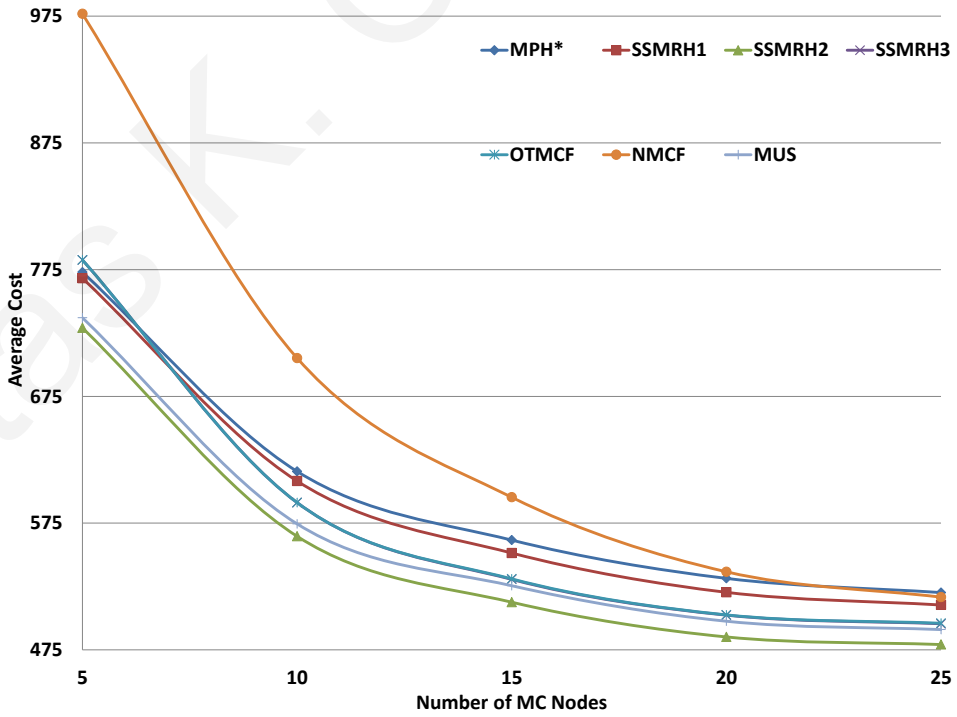


Figure 4.11: Comparison of the multicast routing heuristic algorithms using DNCD for splitter allocation (DoC network).

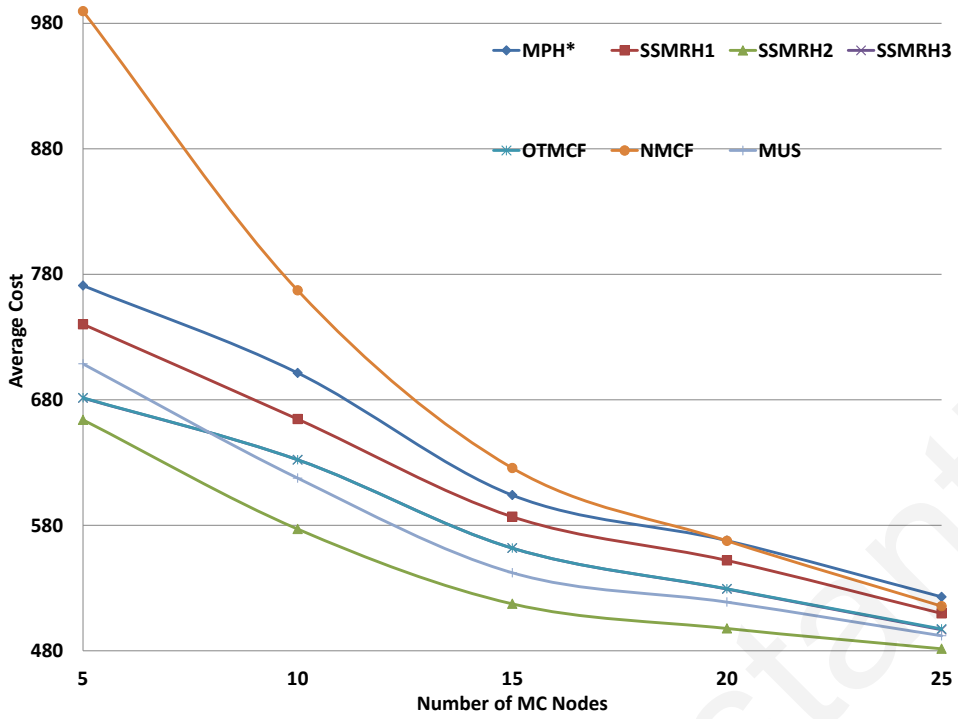


Figure 4.12: Comparison of the multicast routing heuristic algorithms using LURF for splitter allocation (DoC network).

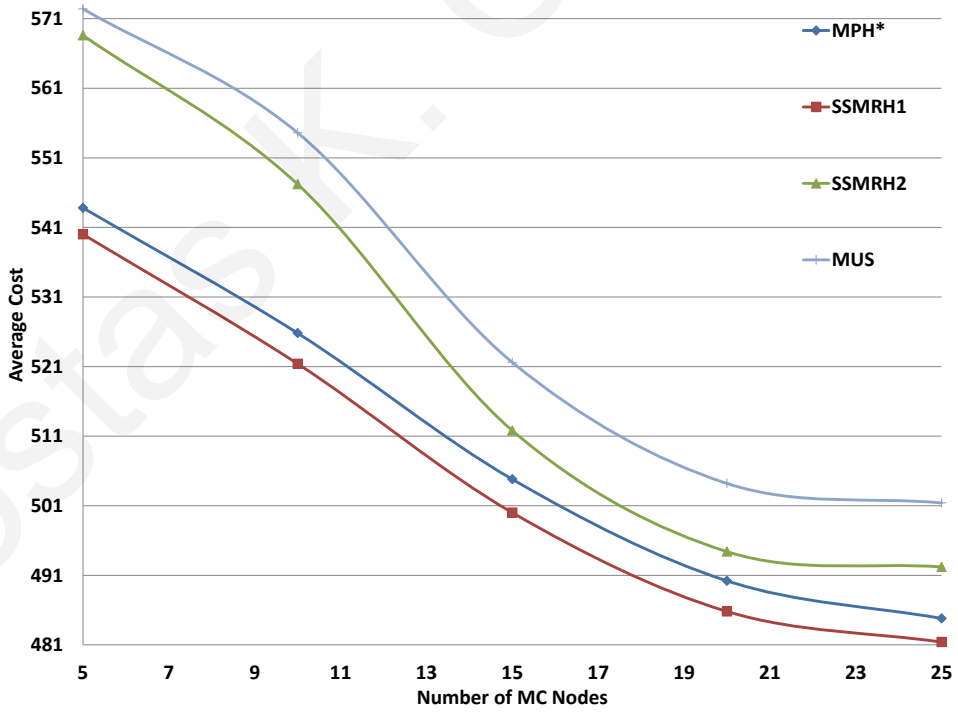


Figure 4.13: Comparison of the multicast routing heuristic algorithms using kmaxD for splitter allocation (DaC network).

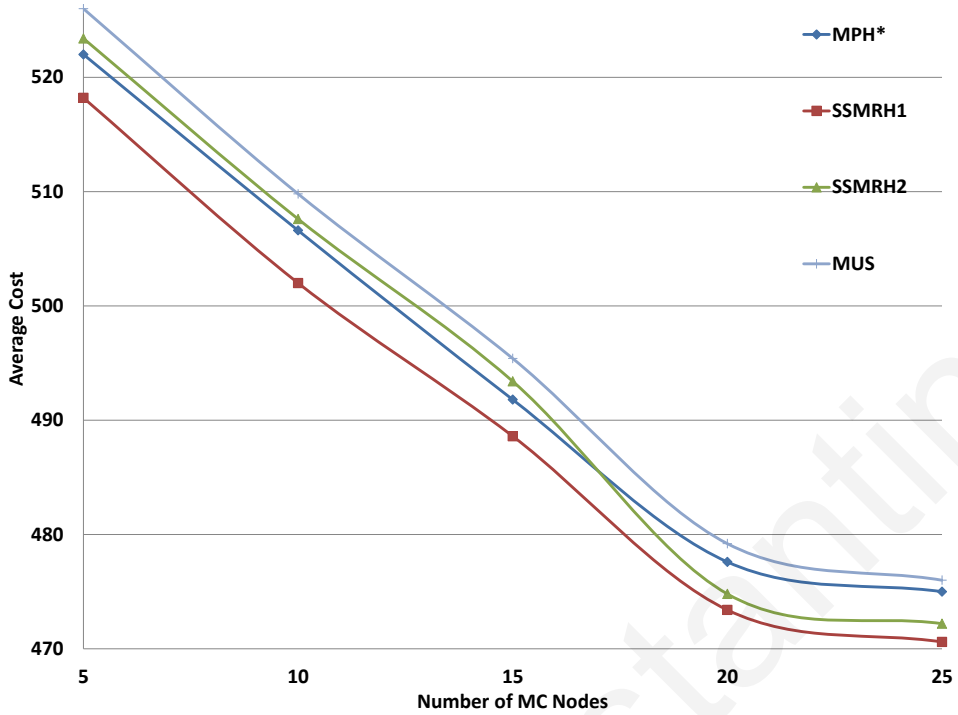


Figure 4.14: Comparison of the multicast routing heuristic algorithms using kmaxWR for splitter allocation (DaC network).

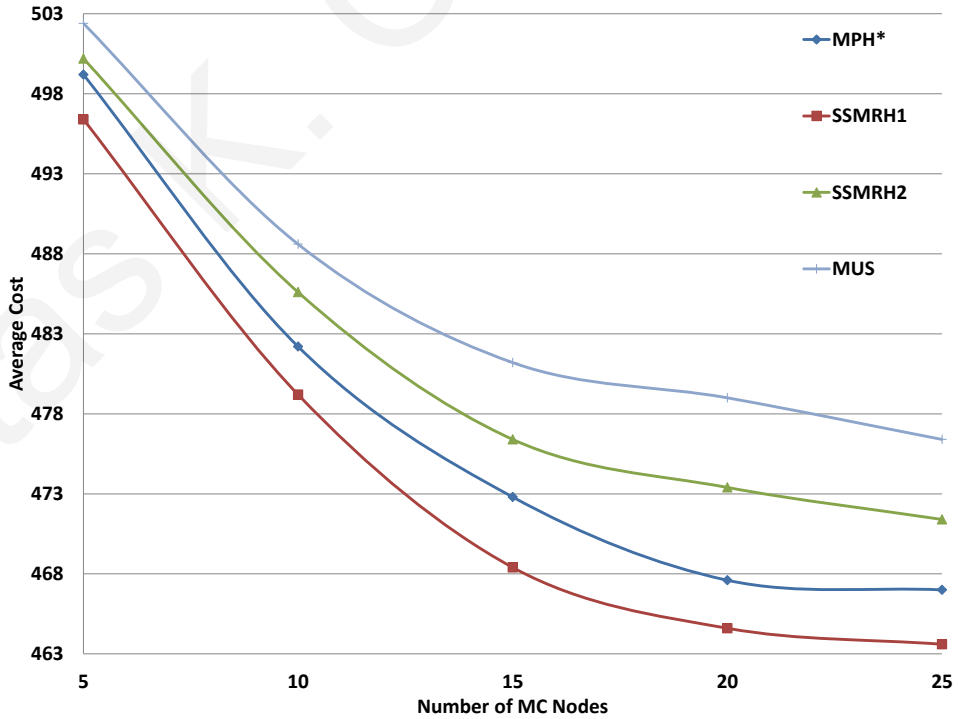


Figure 4.15: Comparison of the multicast routing heuristic algorithms using DNB for splitter allocation (DaC network).

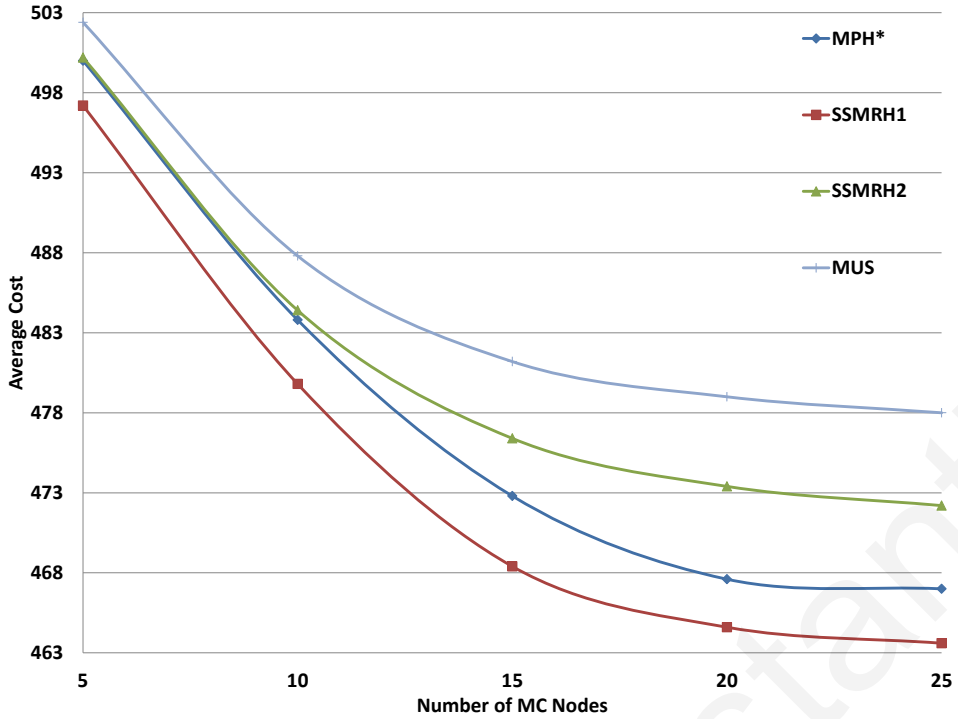


Figure 4.16: Comparison of the multicast routing heuristic algorithms using DNCD for splitter allocation (DaC network).

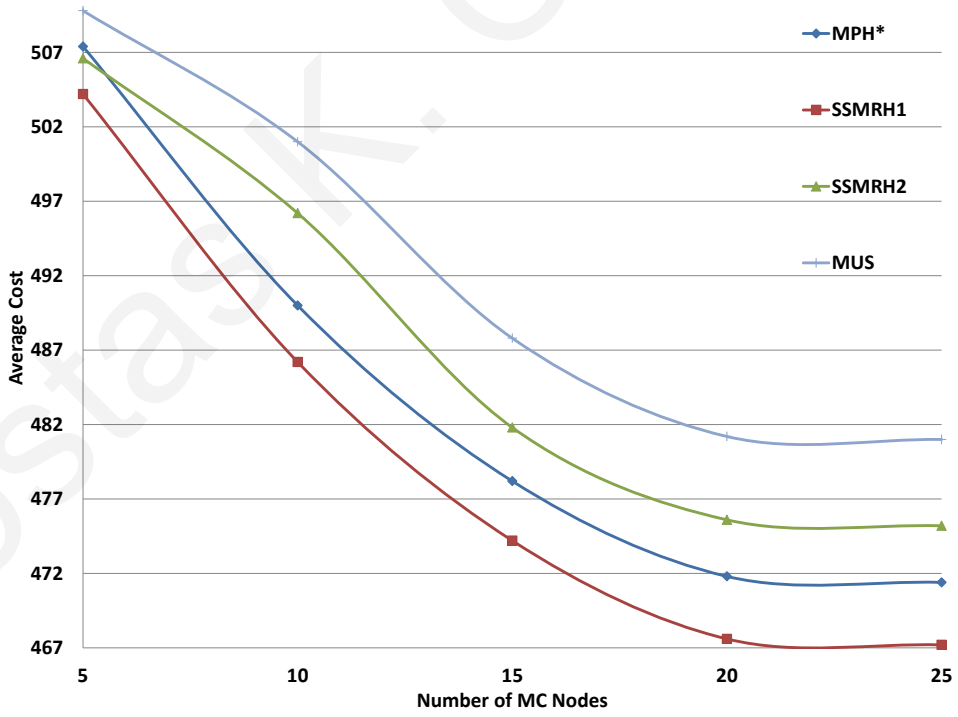


Figure 4.17: Comparison of the multicast routing heuristic algorithms using LURF for splitter allocation (DaC network).

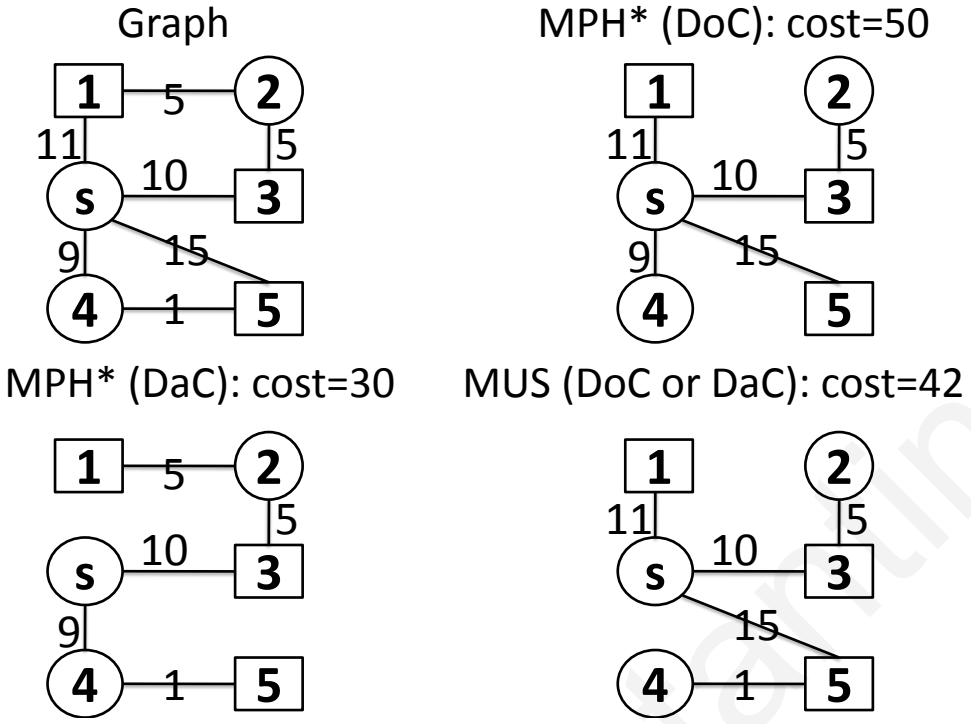


Figure 4.18: Comparison of MPH^* and MUS for DoC and DaC networks.

- SSMRH3 (i.e., using OTMCF as its basis) has poor performance; it performs the same as OTMCF.
- SSMRH1 and SSMRH2 gave better results compared to the heuristic algorithms used as their basis (i.e., MPH^* and MUS respectively) for all cases, regardless of the splitter allocation heuristic algorithm used, or whether the network was DoC or DaC.

4.6.2 Comparison of the MC Node Placement Heuristics

The performance of the proposed heuristics for the placement of the MC nodes in the network were evaluated and compared. The network graph was randomly created. It consisted of 50 nodes and 200 links and it was undirected (i.e., every connection was bidirectional). Again, the constraint that every arc that was added in the graph had to connect nodes that satisfy $d_{nom}^{ij} \leq 5\ \forall i, j$, was used for the random graph creation. Each link had cost randomly selected from 1 to 100. The following procedure was repeated for each different number of MC nodes:

The simulation was repeated for various possible multicast group sizes, from $D = 5$ to $D = 25$ (D stands for the number of destinations), with a step equal to 5. The experiment was executed 5000 times for every multicast group size, while

the source and destinations of the multicast connections were distributed uniformly across the network. The average tree cost was calculated for each multicast group size, over the 5000 repetitions. This was averaged again, over the different multicast group sizes (defined as $\overline{Cost}$).

The multicast tree was calculated by the SSMRH2 heuristic for the case of DoC MI nodes, and by the SSMRH1 heuristic for the case of DaC MI nodes. The simulation was performed for MC nodes up to the total number of network nodes (i.e., up to the case of a full-splitting network). The procedure was executed for both DoC and DaC networks and Figures 4.19 and 4.20 give the respective results. The x-axis gives the number of MC nodes and the y-axis gives the $\overline{Cost}$.

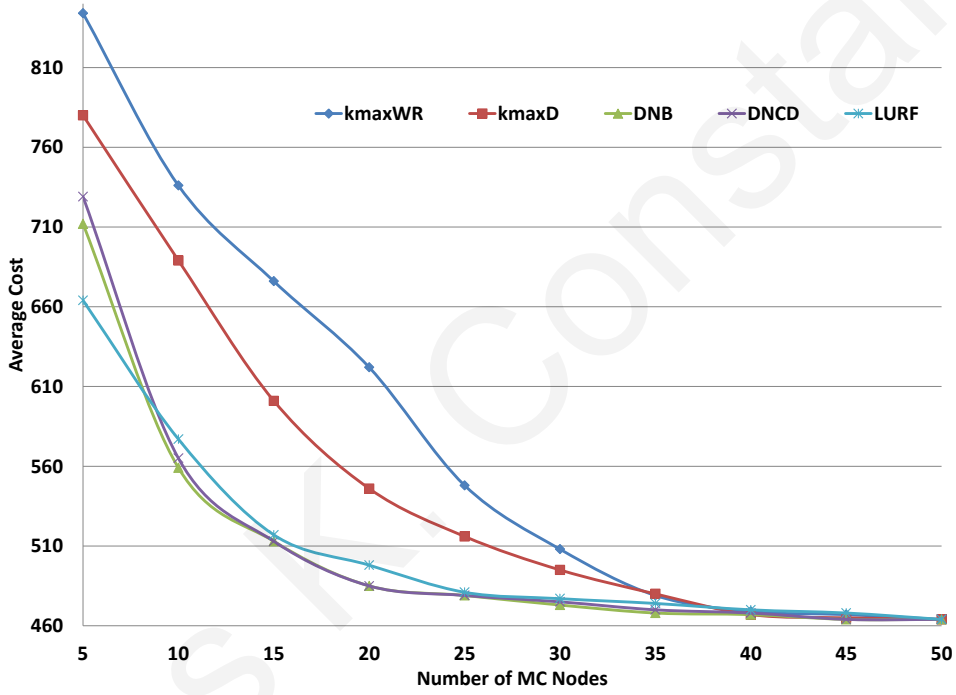


Figure 4.19: Comparison of the MC node placement heuristics for DoC networks, using SSMRH2 as the multicast routing heuristic algorithm.

Analysis of the Results

For the case of DoC networks, LURF is the most efficient approach for small number of MC nodes: up to 9 MC nodes over a 50-node network, i.e. 18%. For larger numbers of MC nodes, the DNB and DNCD splitter allocation heuristic algorithms were the most efficient, having approximately the same performance (DNB is slightly better). It must be noted, though, that in sparse-splitting networks a small number of MC

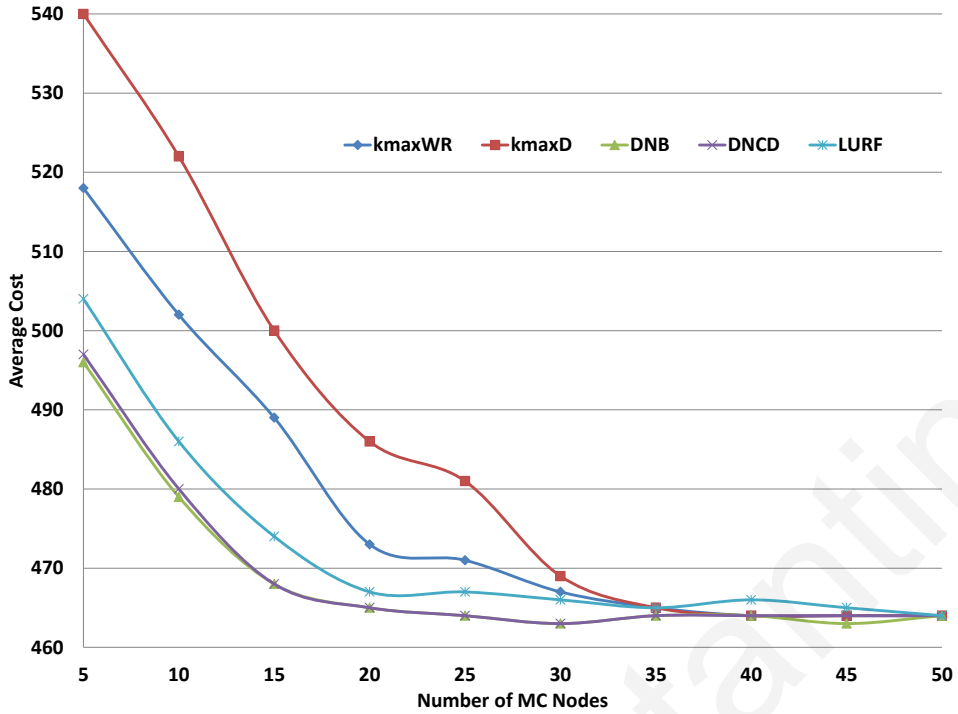


Figure 4.20: Comparison of the MC node placement heuristics for DaC networks, using SSMRH1 as the multicast routing heuristic algorithm.

nodes is usually used, in order to avoid possible multiple attenuation of the optical signal due to splitting.

For the case of DaC networks, DNB and DNCD clearly outperform the rest, for all possible numbers of MC nodes, with all three proposed methods outperforming the existing solutions by a wide margin.

4.6.3 Optimal Percentage of MC Nodes

The study of the results of heuristic algorithm DNB (since this is the most efficient) of Figures 4.19 and 4.20, provides the percentage of MC nodes a network must have. The percentage of the MC nodes depends on their cost. If more of them exist in the network, the average multicast tree cost will be less, but the cost of the network deployment will be more (the assumption is that MC nodes are more expensive than MI nodes). Figures 4.19 and 4.20 show that for both DoC and DaC networks, utilizing up to 20 MC nodes (i.e., up to 40% of the total number of network nodes) provides a significant decrease of the average cost. More than this percentage gives little improvement, showing that most of the benefit of a network with full-splitting capability is achieved with less than half the number of MC nodes. Their exact

percentage from 5% to 40% depends on the available network deployment budget.

4.6.4 Comparison Between DaC and DoC Networks

The importance of the existence of DaC nodes is given in Figure 4.21, by calculating the average performance for both DaC and DoC networks. The multicast tree was calculated by the SSMRH2 heuristic for the case of DoC MI nodes, and by the SSMRH1 heuristic for the case of DaC MI nodes. Heuristic DNB was used for the placement of the MC nodes. The results were averaged over the different multicast groups, for each different number of MC nodes. The simulations were performed for MC nodes up to the total number of network nodes.

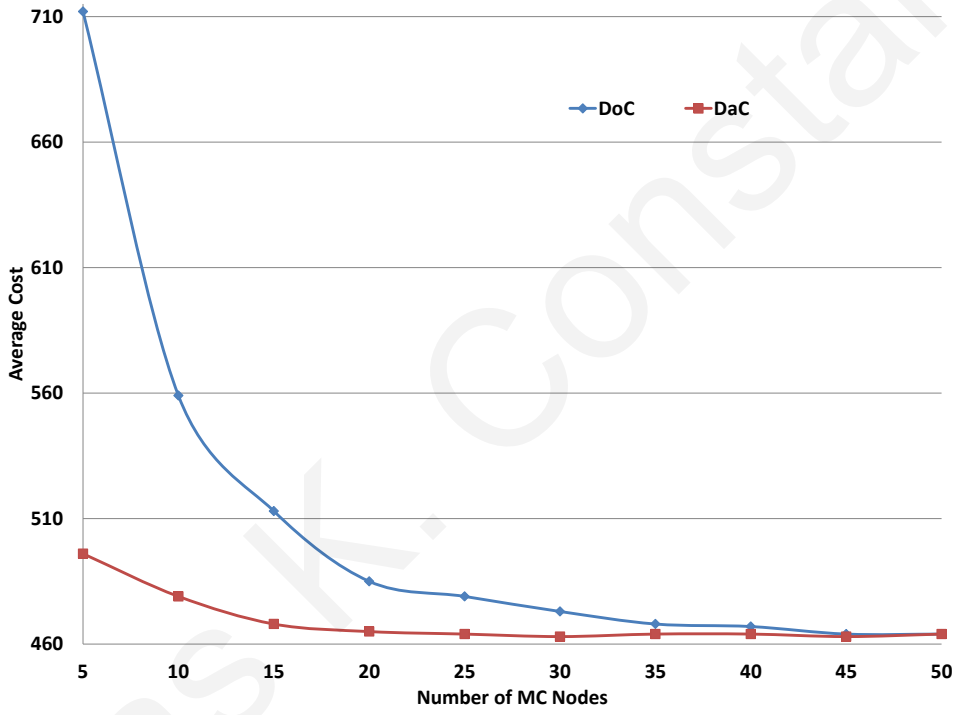


Figure 4.21: Comparison between DaC and DoC networks, using the DNB splitter allocation heuristic algorithm.

Analysis of the Results

DaC networks perform better compared to DoC, as it was expected. The average cost difference is large for a small percentage of MC nodes, and decreases for networks with more MC nodes. For networks where 70% or more of the network nodes are MC, the existence of DaC nodes is unnecessary. However, for networks with a small number of MC nodes (e.g., 5 – 10%), DaCs are essential.

Another result that is obtained from this graph, is the equivalence between the two types of networks. The equivalent of a DaC network with 5 MC nodes (10%) is a DoC network with 17 MC nodes (34%). These two cases have approximately the same average cost performance. Therefore, the selection between them depends on the cost of DaC nodes compared to the cost of MC nodes during network deployment. For the current example (where the network consisted of 50 nodes), if DaC_{cost} is the cost of a single DaC node and MC_{cost} is the cost of a single MC node, the total cost of a DaC network with 5 MC nodes would be $45 \cdot DaC_{cost} + 5 \cdot MC_{cost}$, and the cost of the alternative solution would be $17 \cdot MC_{cost}$. Since these two networks perform the same, the better solution will then be the one that provides the smallest total cost for network deployment.

4.7 Conclusions

In the current chapter, the problems of multicast routing and MC node placement for networks with sparse-splitting capabilities were studied. New heuristic algorithms for both problems were presented, explained, and evaluated through simulations. Both cases of DaC and DoC networks were simulated. It was shown through simulations that the proposed heuristics outperform the existing ones, in terms of average cost of the calculated trees.

The best multicast routing heuristic (for the network graphs used for the simulations) is SSMRH, regardless of the technique used for the allocation of the MC nodes, the number of the MC nodes, or the size of the multicast group. More specifically, SSMRH using MPH^* as its basis is the most efficient heuristic for DaC networks and SSMRH using *MUS* as its basis is the most efficient for DoC networks.

The best splitter allocation technique for DoC networks is LURF for small percentage of MC nodes (up to 18%) and DNB for networks consisting of a larger number of MC nodes. For DaC networks, DNB is the most efficient for any percentage of MC nodes. Finally, it was shown that DaC networks perform better than DoC, especially for networks with a small number of MC nodes.

Chapter 5

Full-Splitting Mixed-Graph Networks: Multicast Routing Heuristic Algorithms

5.1 Introduction

The calculation of least-cost multicast trees is a fundamental problem in graph theory called the "Steiner Minimal Tree" (*SMT*) problem and it was proven to be an NP-complete problem for undirected, mixed, and directed graphs [12]. Therefore, algorithms with polynomial-time complexity that find the optimal solution in every case likely do not exist. The heuristics that are used extensively give satisfactory solutions for undirected graphs, but have poor performance for mixed graphs, defined as graphs consisting of both bidirectional and unidirectional connections between their nodes. The SMT problem for mixed graphs, theoretically, is the most fundamental problem to be solved, since the extensively studied SMT problem for undirected graphs is a special case of it. This case is very important in practice as well; even if the network is designed to be undirected, when some demands arrive and hold some of the resources of the network, the resulting network graph is mixed, therefore the routing for subsequent demands will be calculated on a mixed-graph network. Another case where mixed-graph routing is required is when two arc-disjoint trees must be found on a graph (e.g., for protection against a single-link failure scenario). If the calculation of the primary and secondary trees is done sequentially, the secondary tree, after the removal of the primary one, will be calculated on a mixed

graph. Furthermore, it is possible that certain network nodes can receive information from other nodes, but they are not able to transmit information to them due to physical layer impairments. All the aforementioned cases lead to a network that must be modelled as a mixed graph.

This chapter addresses the problem of multicast routing in mixed-graph mesh optical networks. New heuristic algorithms that are designed for mixed graphs, are presented and evaluated. These heuristics are also compared with multicast routing heuristic algorithms widely used in mesh optical networks. It is shown through simulations that the proposed heuristics have enhanced performance compared to the existing ones in terms of the cost of the resulting multicast trees for different multicast group sizes and different percentages of directionality for the mixed-graph networks. The worst-case performance of multicast routing heuristic algorithms widely used in optical networks is also theoretically calculated and analyzed in order to motivate the design of algorithms specifically for mixed-graph networks.

5.2 Definitions

5.2.1 Undirected, Mixed, and Directed Graphs

Prior to the description and evaluation of the existing and proposed multicast routing heuristic algorithms for mixed-graph networks, it is essential to provide some important definitions that will be used in the rest of the chapter. These can be found in [62, 63], as well as other sources.

Unidirectional edges of a graph are defined as *arcs* and bidirectional edges are defined as *links*. Therefore, every link consists of a pair of opposite arcs (arcs with opposite orientation and equal weight in this work). A graph is considered as *undirected* if all edges are links, *directed* if all edges are arcs, and *mixed* if it has both links and arcs.

An example of these definitions is given in Figure 5.1; in the first graph *all* connections are bidirectional, therefore this graph is considered to be *undirected*, while in the second one the connections $2-3$, $2-5$, $7-5$, and $8-7$ are unidirectional (i.e., the opposite arcs do not exist) and the rest are bidirectional. This graph is thus *mixed*.

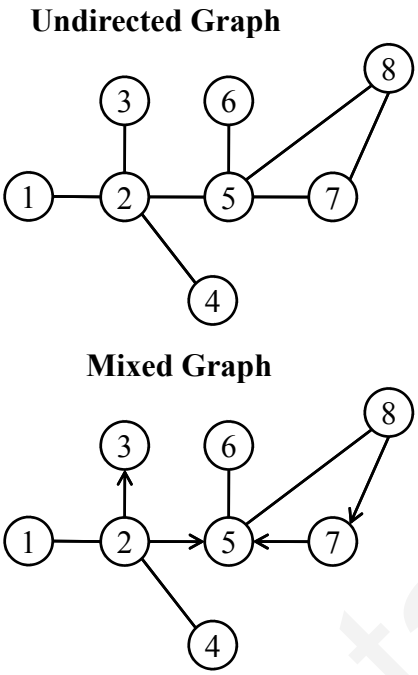


Figure 5.1: Undirected and mixed graphs.

5.3 Widely-Used Multicast Routing Heuristic Algorithms

5.3.1 PPH and MPH Heuristic Algorithms

As discussed extensively in Chapter 3, the SMT is an NP-complete problem and several heuristic algorithms have been developed for obtaining the multicast trees. Furthermore, as also discussed in Chapter 3, the heuristics that are widely used for solving the Steiner tree problem and, consequently, for multicasting applications in optical and other networks, are the *Pruned Prim Heuristic (PPH)* and the *Minimum Path Heuristic (MPH)* [15]. Even though these heuristics work satisfactorily for undirected graphs (although even more efficient heuristics can be found for this case, as it was shown in Chapter 3), they are not optimized for use in mixed-graph networks. Their limited performance in mixed-graph networks, in terms of the cost of the calculated tree, is shown in an example of a mixed graph in Figure 5.2. Here, node s is the source node and nodes d_1 and d_2 are the destination nodes of the multicast connection. Heuristics MPH and PPH give a tree of cost equal to 41, while the Steiner Minimal Tree (SMT) has cost equal to 32.

These two undirected-graph multicast routing heuristic algorithms (especially

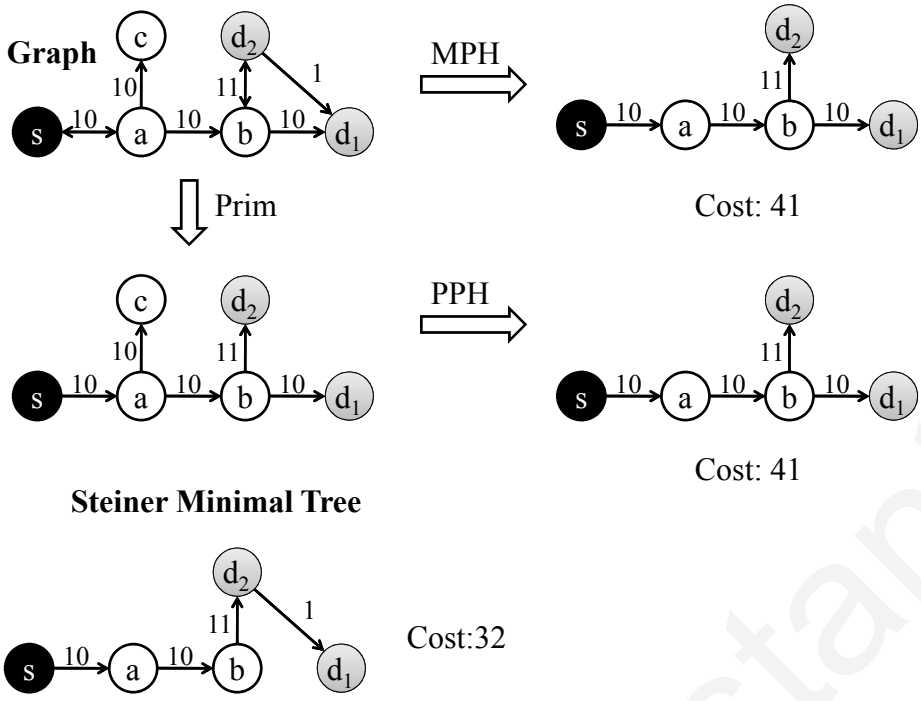


Figure 5.2: Example of the MPH and PPH heuristic algorithms for a mixed-graph network.

MPH) are extensively used in mesh optical networks [20, 23, 46-51, 56]. Several other heuristics have also been written for this problem [36, 38, 39, 41-44]. For the undirected graph case, all the latter heuristics have smaller *SR* compared to MPH and PPH, i.e., they have better *worst-case* performance. Despite this, the former are generalized in this chapter in order to work more efficiently for mixed-graph networks. The reasons are: (i) MPH and PPH are the multicasting heuristic algorithms that are widely used in optical networks. In fact, no papers were found in the literature that utilize the heuristics presented in [36, 38, 39, 41-44] for any network applications. Therefore, in this thesis it was considered important to generalize these two heuristics and compare them to the newly proposed multicast routing techniques for mixed-graph networks; (ii) They are simple and easy to develop and implement and they have lower complexity compared to the aforementioned heuristics that have smaller *SR*, which is an important parameter when provisioning dynamic multicast connection requests. Thus, their generalization to the mixed-graph network case will also yield lower complexities compared to the heuristic algorithms in [36, 38, 39, 41-44]; (iii) The heuristics in [36, 38, 39, 41-44] are more efficient than MPH and

PPH only in terms of *worst-case* performance. In practical applications, though, it is the *average-case* performance that is of interest. Therefore, there is the possibility that for network applications the benefit of using the heuristics with lower SR will not be important enough as to merit the cost of higher time-complexity; (iv) Their enhanced performance appears only under certain "trap" topologies, which are highly unlike to appear in telecommunication network topologies; and (v) Based on the way the heuristic algorithms are designed, it is not even clear whether it is even possible for any of the algorithms in [36, 38, 39, 41-44] to be modified and be applied for mixed graph networks.

The aforementioned algorithms deal with the undirected case of the Steiner problem in graphs. Algorithms that give a solution for the mixed-graph Steiner tree problem do not exist in the literature. For the directed-graph case, little work exists in the literature, and the main techniques are restricted to [64] and [65] and other related papers.

In the following section, the worst-case performance of PPH and MPH is mathematically derived in order to demonstrate the necessity of developing multicast routing heuristic algorithms that are designed specifically for mixed-graph topologies.

5.3.2 Theoretical Worst-Case Performance of PPH and MPH in Mixed Graphs

Figure 5.3 presents a graph where the PPH and MPH heuristic algorithms have their worst performance. Suppose that $C >> 1 >> \epsilon$ and consider the case where all nodes except node k are destinations. Prim's algorithm will connect all nodes named i, $1 \leq i \leq k$ through arcs (*source*, i) respectively, and then node x will be connected through arc (*source*, x). Since node k does not belong to the destination set, arc (*source*, k) is deleted (it is *pruned*). The same tree is obtained when the MPH heuristic algorithm is applied. The resulting tree has cost equal to $(C \cdot k + \epsilon)$.

The optimal tree is achieved by adding node x in the first step and all the remaining destinations will be connected through this node. The cost of the resulting (optimal) tree is $(k - 1) + (C + \epsilon)$. The ratio of the cost of the tree obtained by PPH and MPH over the cost of the optimal one, called A, is then calculated as:

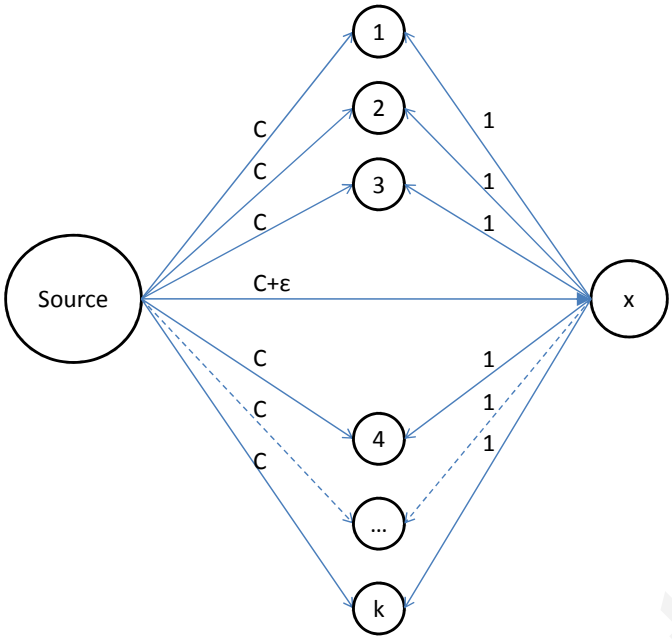


Figure 5.3: Worst-case performance of PPH and MPH.

$$A = \frac{Ck + \epsilon}{(k-1) + (C+\epsilon)} \approx \frac{Ck}{k+C} \tag{5.1}$$

Therefore, A is not bounded above. It can take an arbitrary large value, with the appropriate values of k and C. For example, if $k = C = 1000$, then $A = 500$, i.e., the tree calculated by PPH and MPH has cost 500 times the cost of the optimal one.

Since a graph was found (the one given in Figure 5.3) where the cost of the tree calculated by the PPH and MPH heuristics over the cost of the optimal tree can take an arbitrarily large value, the conclusion is that the worst-case performance of the aforementioned heuristics is not bounded above. Thus, it is necessary to develop multicast routing heuristic algorithms that are designed specifically for mixed graphs.

In the following section, an algorithm that calculates the MST in mixed graphs, called the *Chu-Liu* algorithm, is presented. It is also described how it can also be used for mixed-graph multicasting.

5.4 Pruned Chu-Liu Heuristic Algorithm

One algorithm that can be used for multicast routing in mixed graphs is the *Chu-Liu* algorithm [66]. This is an algorithm that calculates the minimum spanning tree on directed and mixed graphs. An equivalent algorithm was also written by Edmonds in [67]. Furthermore, Bock [68] has also created a similar algorithm that is stated on matrices instead of graphs. The focus in this section is on the Chu-Liu implementation which is the most commonly used in algorithmic graph theory. The Chu-Liu algorithm consists of the following steps:

Chu-Liu Algorithm

Input: Directed graph $G_0(V, E)$ ($|V| = n$, $|E| = e$).

Output: Tree $T_0(V, E^*)$ ($E^* \subset E$).

Notations:

- (x, y) is an arc that originates from node x and ends at node y.
- $c(x, y)$ is the cost of arc (x, y).

1. $i = 0$. The initial graph is G_0.
2. For each node of graph G_i except the source, select the entering arc with the smallest cost;
3. If the resulting subgraph is a tree (i.e., no cycles exist) go to Step 5. Otherwise, continue to Step 4.
4. (a) $i = i + 1$;
   (b) Each cycle is retracted into a single *pseudo-node*.
   (c) A new graph is obtained after these retractions, called G_i. The cost of each of its arcs that does not end at a pseudo-node remains the same as in graph G_{i-1}. Each arc (x, j) that originates from node x outside pseudo-node k and ends at a node j in pseudo-node k, is replaced by arc (x, k) that has cost equal to $c(x, k) = c(x, j) - c(y, j) + MAX$, where (y, j) is the arc that belongs to pseudo-node k and ends in j and MAX is the maximum cost of the arcs of the cycle that was retracted into k.
   (d) Return to Step 2.

5. If the procedure stops at the first iteration, the calculated tree is the final solution. Otherwise, the resulting tree is T_I. Tree T_{I-1} is extracted from tree T_I by expanding the pseudo-nodes in it. Each pseudo-node is replaced by the corresponding cycle and the arc that belongs to the cycle and ends at the real node at which the arc that connects the cycle with the tree ends as well, is deleted. This procedure is repeated until tree T_0 (without any pseudo-nodes) is obtained.

The aforementioned algorithm gives the minimum spanning tree for a directed graph and its time complexity is ($O(e + nlogn)$) [66], where $e = |E|$, $n = |V|$. For multicasting applications, a modification of it can be used by simply pruning the unwanted arcs and nodes (*Pruned Chu-Liu Heuristic* (*PCLH*)). It consists of the following steps:

Pruned Chu-Liu Heuristic

1. Calculate the minimum spanning tree using the Chu-Liu algorithm.
2. Keep only the paths that connect the source with the destinations of the multicast session and delete ("prune") the rest of the arcs and nodes of the minimum spanning tree.
3. The resulting tree is the multicast tree.

5.5 Proposed Mixed-Graph Routing Heuristic Algorithms

5.5.1 Mixed-Graph Pruned Prim Heuristic Algorithm

It was proven theoretically and shown through a simple example that Prim's algorithm and consequently the PPH heuristic, do not perform well for mixed-graph networks. The reason is that, after the addition of a node in the tree, it is possible that the already added nodes can be connected to the tree more efficiently through the last added node. Two algorithms that are generalizations of Prim's algorithm are presented here. They both outperform Prim for mixed graphs, and they have the same performance with it for undirected ones. They are called *Mixed Graph Spanning Tree* (*MGSTa* and *MGSTb*). The second one is more efficient in terms of the cost of the calculated tree, but with larger time-complexity. Both of them are polynomial in time. For their description, the following are used:

- s: source node.
- $p(i)$: predecessor of node i in the tree.
- $c(i, j)$: cost of arc (i, j).
- D: destination set.

(For the following heuristics, the shortest paths (and the corresponding distances) between every pair of graph nodes, are calculated using Floyd-Warshall's all-pair shortest path algorithm [16, 17]).

MGSTa Heuristic

1. Tree T_1 initially consists only of the source node s. Set $V_1 = \{s\}$, $i = 1$.
2. (a) Determine the node u in $\{V - V_1\}$, that is closest to T_i (which is connected through an arc originating from T_i and ending at u). Construct a tree T_{i+1} by adding the arc joining u and T_i.
   (b) For every node $v \in \{V_1 - \{s, p(u)\}\}$: Compare arcs (u, v) and $(p(v), v)$. If $c(u, v) < c(p(v), v)$ and the addition of (u, v) in the tree does not create a cycle, replace $(p(v), v)$ with (u, v) and put $p(v) = u$.
   (c) Replace $\{V_1\}$ with $\{V_1\} + v$. $i = i + 1$.
3. Repeat Step 2. STOP when $V_1 = V$, i.e., when all nodes are connected to the tree.

MGSTb Heuristic

1. Tree T_1 initially consists only of the source node s. Set $V_1 = \{s\}$, $i = 1$.
2. (a) Determine the node u in $\{V - V_1\}$, that is closest to T_i (which is connected through an arc originating from T_i and ending at u). Construct a tree T_{i+1} by adding the arc joining u and T_i.
   (b) For every node $v \in \{V_1 - \{s, p(u)\}\}$: Compare arcs (u, v) and $(p(v), v)$. If $c(u, v) < (cp(v), v)$ replace $(p(v), v)$ with (u, v). Put $p(v) = u$. In the case a cycle was created, modify the cost of each arc which enters a node j in the cycle and originates from some node i in the current tree (not belonging to the cycle or to the cycle branches), according to the following equation:

$c^*(i,j) = c(i,j) - c[p(j),j]$. From these arcs, select the one with the smallest modified cost and replace the arc which enters the same node ("w") in the cycle by the new selected arc. Modify the predecessor of w, according to the node from which the new selected arc originated.

(c) Replace $\{V_1\}$ with $\{V_1\} + v$. $i = i + 1$.

3. Repeat Step 2. STOP when $V_1 = V$, i.e., when all nodes are connected to the tree.

In these algorithms, when a node is added to the tree, it is checked whether the nodes that are already on the tree, can be connected to it via a "shorter" way, through the newly added node. MGSTa stops if this change creates a cycle and MGSTb allows the cycle to be created and finds the most optimal way to break it. In some cases this leads to the same tree as before the creation of the cycle, but, in general, the creation and breaking of this cycle gives a tree with less cost.

Both of the proposed spanning tree algorithms are more efficient for mixed graphs, compared to Prim's algorithm. MGSTb gives better results than MGSTa, but has higher time-complexity.

The application of the two heuristics to the example of Figure 5.4 will make them more understandable. The first three steps are identical to Prim's algorithm. In Step 4, after the connection of node 3 to the tree, the arcs $(3,4)$ and $(2,4)$ are compared. As $c(3,4) < c(2,4)$, arc $(3,4)$ now replaces arc $(2,4)$ in the tree, since this change does not create a cycle. In Step 5, arc $(4,5)$ is added. After the insertion of node 5 in the tree, node 3 is closer to the tree through arc $(5,3)$ instead of arc $(1,3)$. This replacement will create a cycle, therefore MGSTa ends at Step 5, since all nodes are connected. On the other hand, using MGSTb, the arcs are replaced (Step 6) and the resulting cycle must break. The arc that enters the cycle from a node outside the cycle and has the minimum modified cost is $(1,3)$. Consequently, arc $(1,3)$ is added and arc $(2,3)$, which is the arc in the cycle that terminates at node 3, is removed. (The predecessors of each node are modified appropriately in each step). The heuristic algorithm terminates because all nodes are added in the tree and all cycles have been removed. MGSTb succeeds in finding a MST with cost= 18, while MGSTa finds a tree with cost= 19, and Prim gives a tree with cost= 23.

It should be noted that if for example arc $(5,3)$ had now cost 4 instead of 1, the two heuristic algorithms would give the same result, since the creation and breaking

of the cycle would lead to the same tree as before the creation of the cycle, i.e., Steps 5 and 7 would result in the same tree.

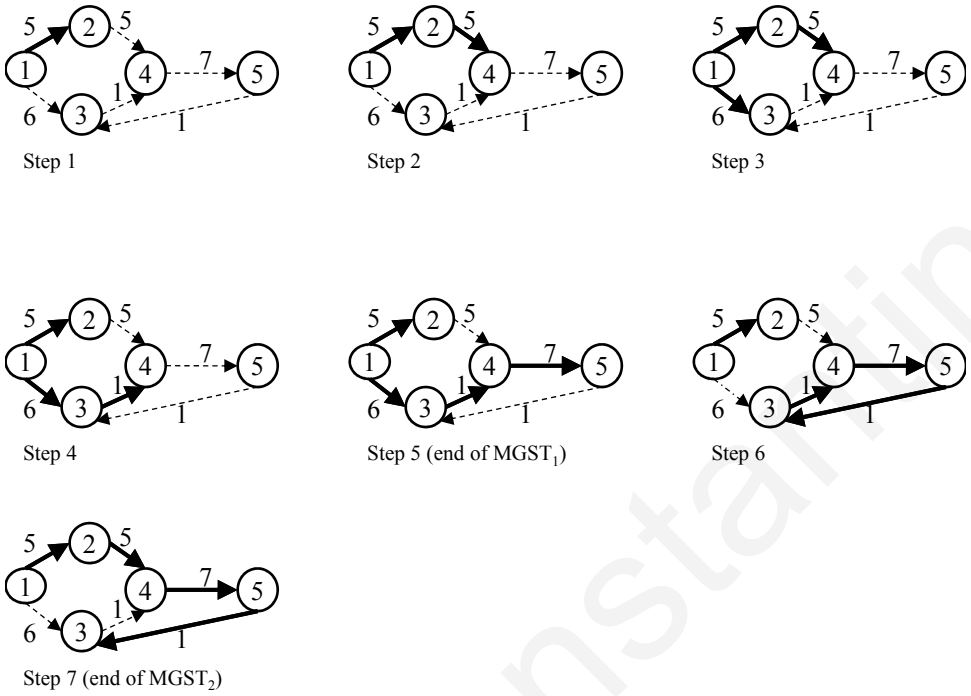


Figure 5.4: Example of the MGSTa and MGSTb heuristic algorithms.

Time Complexity

The derivation of the time-complexity for MGSTa and MGSTb is as follows:

1. Prim's algorithm has time complexity on the order of $O(elogn)$ [45] ($e = |E|$, $n = |V|$).

2. In Step 2b of MGSTa, all nodes in set $V_1 - (s, p(u))$ are checked, therefore the complexity is multiplied by $O(n)$.

3. In Step 2b of MGSTb, all nodes in set $V_1 - (s, p(u))$ are checked ($O(n)$). For the breaking of a created cycle, a set of arcs that originate from the current tree and end at the cycle are checked ($O(n)$).

The result of the three previous statements is that the time-complexity of MGSTa is $O(nelogn)$ and MGSTb has time-complexity $O(n^2elogn)$.

The two proposed heuristic algorithms can now be utilized for multicasting in a way similar to PPH: the minimum spanning tree is calculated using MGSTa or

MGSTb, and the unnecessary arcs are pruned. The resulting multicasting algorithms are called *Mixed-Graph Pruned Prim Heuristic* algorithms (*MG-PPHa* and *MG-PPHb*). It can be easily verified that both of them give the Steiner minimal tree, if they are applied to the example of Figure 5.2 as well as to the example in Figure 5.3.

5.5.2 Mixed-Graph Minimum Path Heuristic Algorithms

When MPH is applied in mixed graphs, after the addition of a destination u in the tree, there is the possibility that some of the already added destinations can be connected more efficiently through the nodes of the path that connects u to the tree. For example, in Figure 5.2, after the addition of destination d_2, d_1 can be connected through arc $d_2 - d_1$ (cost= 1), instead of arc $b - d_1$ (cost= 10) that connects it to the tree. A family of new heuristic algorithms, called *Mixed Graph Minimum Path Heuristic* algorithms (*MG-MPHa*, *MG-MPHb*, and *MG-MPHc*) were developed to overcome this.

For their description, the following are used:

- s: source node.
- D: destination set.
- Z: set consisting of the source and the destinations.

MG-MPHa Heuristic

1. Tree T_1 initially consists only of the source node s. Set $V_1 = \{s\}$, $i = 1$.
2. (a) Determine a node u in $\{D - V_1\}$ that is closest to T_i (which is connected through a directed path originating from T_i and ending at u). Construct a tree T_{i+1} by adding the minimum cost path joining u and T_i. Add the nodes of this path in set $\{V_1\}$.

   (b) Keep only the path from source node to node u as the current tree and delete all other arcs.

   (c) Add again to the current tree all the destinations that were part of the tree before u, using the MPH algorithm.

   (d) $i = i + 1$.

3. Repeat Step 2. STOP when all nodes of D are connected to the tree.

The heuristic functions as follows: After the addition of destination u (let U be the set of the nodes that are added to the tree for the connection of destination u), only the path from source s to u is kept as the current tree. All destinations that were part of the tree before the addition of u and do not belong to path $s \rightarrow u$, are added again using the MPH algorithm. Therefore, they will be connected either through the path that they were already connected, or through a path that originates from a node of set U. The best path between the two will be selected.

The MG-MPHa heuristic algorithm is applied to the graph of Figure 5.2, and its steps and result are shown in Figure 5.5. In the first step, destination d_1 is connected to the tree, since it is closer to the source than destination d_2. In Step 2, d_2 is added (MPH would stop here). In Step 3, only the path $s - d_2$ is kept as the current tree and the paths that connect the rest of the destinations (i.e., path $b - d_1$) are removed from the tree. Then, in Step 4, the destinations that were part of the tree before d_2 was added (i.e., d_1) are connected again using the MPH heuristic (d_1 is connected to the tree though d_2 using arc $d_2 - d_1$), and the heuristic terminates. Therefore, the calculated tree has cost equal to 32, while the PPH and MPH heuristics had calculated a tree with cost equal to 41.

MG-MPHa checks whether the destinations that were added in the tree before the last added destination u, can be added in a shorter way after the addition of u and its corresponding path. This procedure is not applied though for the destinations added before u that belong to path that connects the source node s and node u (see Step 2b). Therefore, two other heuristic algorithms that overcome this weakness were developed. Suppose that the destinations that were added before u belong either to set A (if they are not part of path $source - u$) or set B (if they are part of path $source - u$); the check if nodes of set B can be connected in a shorter way through u can be done before the addition of nodes of set A in the tree (heuristic algorithm MG-MPHb), or after (heuristic algorithm MG-MPHc).

MG-MPHb Heuristic

1. Tree T_1 initially consists only of the source node s. Set $V_1 = \{s\}$, $i = 1$.

2. (a) Determine a node u in $\{D - V_1\}$, that is closest to T_i (which is connected through a directed path originating from T_i and ending at u). Construct

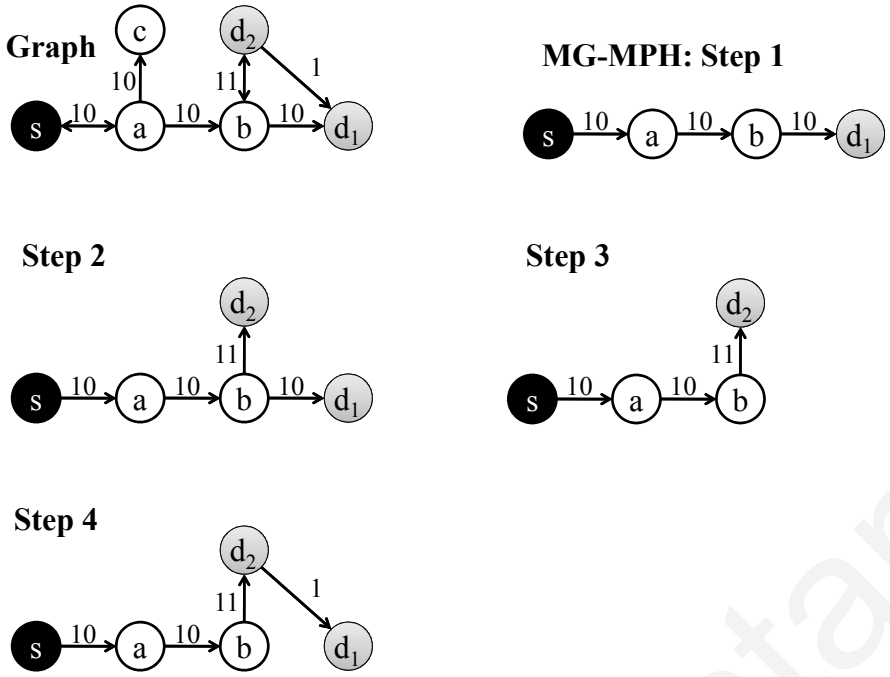


Figure 5.5: Example of the MG-MPHa heuristic algorithm.

a tree T_{i+1} by adding the minimum cost path joining u and T_i. Add the nodes of this path in set $\{V_1\}$.

(b) Keep only the path from source node to node u as the current tree and delete all other arcs and calculate its cost, $cost_a$ (resulting current tree: $tree_a$).

(c) Remove this path, connect node u with node s through the shortest path, add again in the tree the destinations of set B, and calculate the cost of the resulting current tree, $cost_b$ (resulting current tree: $tree_b$).

(d) If $cost_a < cost_b$, keep $tree_a$, otherwise keep $tree_b$.

(e) Add again to the current tree all the destinations that were part of the tree before u (i.e., nodes of set A), using the MPH algorithm.

(f) $i = i + 1$.

3. Repeat Step 2. STOP when all nodes of D are connected to the tree.

Simply, heuristic MG-MPHb checks whether the destinations that belong to path $source - u$ (i.e., were added into the tree *before* u) can be connected to the tree in a

shorter way, *after* the addition of node u. The more efficient way between the two is selected. This procedure is applied before the re-addition of nodes of set A in the tree.

A third heuristic algorithm, MG-MPHc, performs this procedure after the re-addition of nodes of set A in the tree:

MG-MPHc Heuristic

1. Tree T_1 initially consists only of the source node s. Set $V_1 = \{s\}$, $i = 1$.

2. (a) Determine a node u in $\{D - V_1\}$, that is closest to T_i (which is connected through a directed path originating from T_i and ending at u). Construct a tree T_{i+1} by adding the minimum cost path joining u and T_i. Add the nodes of this path in set $\{V_1\}$.

   (b) Keep only the path from source node to node u as the current tree and delete all other arcs and calculate its cost, $cost_a$ (resulting current tree: $tree_a$).

   (c) Add again to the current tree all the destinations that were part of the tree before u (i.e., nodes of set A), using the MPH algorithm. Calculate the cost of the resulting current tree $tree_a$, $cost_a$.

   (d) Check path $source - u$: Find the first node after the source, that has $outdegree > 1$. Name this node x. If no node with $outdegree > 1$ exists, $x = u$.

   (e) Remove path $source - x$, connect x with s through the shortest path, add again in the tree the destinations of set B, and calculate the cost of the resulting current tree $tree_b$, $cost_b$.

   (f) If $cost_a < cost_b$, keep $tree_a$, otherwise keep $tree_b$.

   (g) $i = i + 1$.

3. Repeat Step 2. STOP when all nodes of D are connected to the tree.

An example of the MG-MPHb and MG-MPHc heuristic algorithms is given in Figure 5.6. Here, s is the source and d_1, d_2, d_3 are the destinations. For the first graph of the figure, it can easily be verified that MPH gives a tree of cost equal to 49 ($s - b - d_3, s - d_1 - a - d_2$), and the cost of the tree derived by MG-MPHa is equal to 40 ($s - d_1 - a - d_2 - d_3$). If either MG-MPHb or MG-MPHc are exploited, the resulting

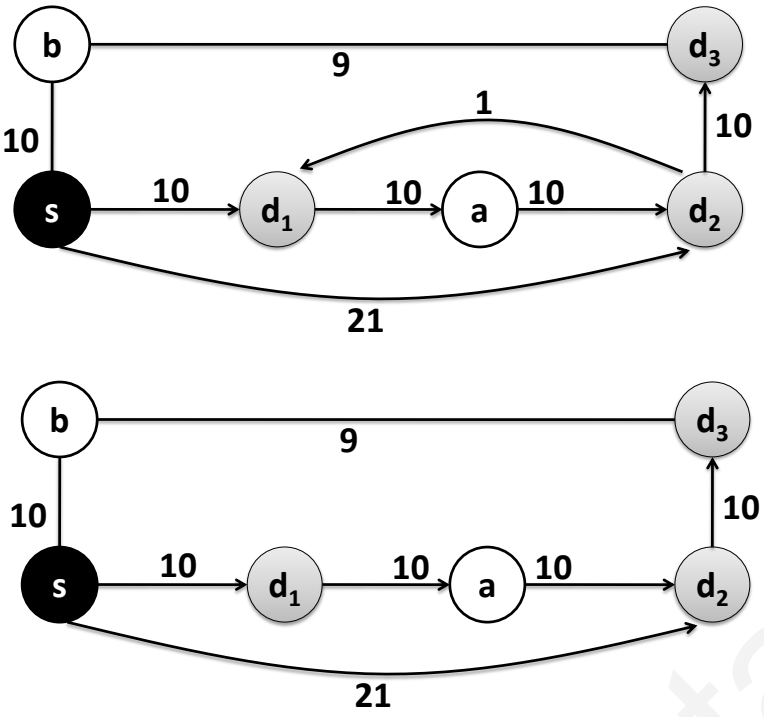


Figure 5.6: Example of the MG-MPHb and MG-MPHc heuristic algorithms.

tree has cost equal to 32 ($s - d_2 - d_1, d_2 - d_3$). This is due to the fact that, after the addition of d_2, d_1 can be connected in a shorter way through d_2. MG-MPHa does not perform this check, since d_1 belonged to the path that connected the source and d_2.

The second graph of Figure 5.6 explains why the comparison of $cost_a$ and $cost_b$ in MG-MPHb and MG-MPHc (steps d and f respectively) is necessary. Here, MG-MPHa, MG-MPHb, and MG-MPHc give the same tree ($s - d_1 - a - d_2 - d_3$) that has cost 40. If the aforementioned steps were omitted, MG-MPHb and MG-MPHc would give tree $s - d_1, s - d_2 - d_3$, with cost 41.

Complexity of the MG-MPH Heuristic Algorithms

The time-complexity of MPH is $O(kn^2)$ [15] ($k = |Z|$, $n = |V|$), since it applies Dijkstra's algorithm that has complexity $O(n^2)$ [31], for the addition of each destination in the tree. MG-MPH heuristics apply MPH after the addition of each destination, therefore they have complexity of order $O(k^2n^2)$.

Based on the discussion in Chapter 3, another heuristic algorithm which is a generalization of MPH, that on the average performs better, and can be modified to be applicable in mixed-graph networks, is the SNH heuristic described in detail in Section 3.3.

If one of the MG-MPH heuristics is used in Steps 1 and 2b of SNH (as described in Section 3.3), instead of MPH, then the resulting heuristic algorithm (called the

Mixed Graph Steiner Node Heuristic (MG-SNH)), will be applicable to mixed-graph networks.

Mixed Graph Steiner Node Heuristic (MG-SNH)

1. Calculate Steiner Tree $T^*(V^*, E^*)$ using MG-MPH and find its cost, called C.
2. For every node $v_i \in \{V - V^*\}$ do the following:
   (a) $\{Z'\} \leftarrow \{Z\} + v_i$.
   (b) Calculate Steiner Tree $T_i(V_i, E_i)$ on $G(V, E)$ for set $\{Z'\}$, using MG-MPH, and find its cost, c_i.
   (c) $\{Z'\} \leftarrow \{Z\}$.
3. Find v_j and $T_j : c_j = min_i\{c_i\}$.
4. If $c_j \geq C$, the solution is $T^*(V^*, E^*)$, STOP.
   (a) If $c_j < C$, replace $\{Z\}$ with $\{Z\} + v_j$, C with c_j and $T^*(V^*, E^*)$ with $T_j(V_j, E_j)$.
   (b) If set $\{V - V^*\} \neq 0$, return to Step 2. Else STOP.

Complexity of the MG-SNH Heuristic Algorithm

The time-complexity of MG-MPH is $O(k^2n^2)$, as shown previously. Since MG-MPH is applied once in Step 1, and Steps 2 and 4 are repeated no more than $V - D$ times each, the complexity of MG-SNH is $O(k^2n^2 + (n-k)^2k^2n^2) = O((n-k)^2k^2n^2)$.

The existing and proposed heuristic algorithms for mixed-graph networks are evaluated in the section that follows.

5.6 Performance Evaluation

In this section, the average performance of the existing and proposed heuristic algorithms for mixed-graph networks discussed previously, is evaluated through simulations. It must be noted that for network applications it is the average performance that counts, since the worst-case one may appear only for certain trap topologies that rarely exist in real telecommunication network topologies.

The simulations were performed on a random graph that was created as follows: Initially, it had 40 nodes and 100 arcs. The cost of each arc was randomly selected from 1 to 1000. The graph was simple, therefore all of the 100 edges connected a different

pair of nodes. There were no arcs with opposite orientation (the graph initially was directed). According to the *Percentage of Directionality (PoD)* (i.e., the ratio *arcs*/*edges*) the appropriate number of arcs (unidirectional edges) were converted to links (bidirectional edges). For example, in the case of a graph with PoD = 20%, 20 arcs were chosen randomly, to be converted to links, i.e., the opposite arc of each one of them was added to the graph. The simulation was repeated for $PoD = 20\%, 40\%, 60\%, 80\%$. For each one of these cases with different PoD, the procedure was repeated for all possible multicast groups (from $D = 2$ to $D = 39$). The experiment was executed 5000 times for each multicast group size, while the source and destinations of the multicast connection were distributed uniformly across the network. For every PoD, 5000 Steiner trees were calculated with each one of the existing and proposed heuristic algorithms and their average cost was calculated for each heuristic. The results are presented in Figures 5.7-5.14. The horizontal axis of each diagram is the multicast group size (i.e., the number of destinations) and the vertical one gives the average cost. Two graphs were created for each PoD, one for small multicast group sizes (up to half of the network nodes) and one for large multicast group sizes (more than half of the network nodes), for better clarity.

For the simulations, the MG-SNH technique in this analysis was realized using MG-MPH*b*, since this has (slightly) better performance compared to MG-MPHa and MG-MPHc, as simulations showed.

Analysis of the results

As it can be seen from the graphs, heuristic MG-MPHb performs slightly better than heuristics MG-MPHa and MG-MPHc, therefore this is the one exploited in MG-SNH.

For all cases of PoD, MG-SNH has the best performance, compared to the rest of the heuristics, for small multicast sessions (i.e., for number of destinations up to half of the network nodes). For large multicast sessions, PCLH has the best performance for all cases of PoD. It must be stated, though, that in practice, multicast sessions with number of destinations more than half of the network nodes, rarely exist (the number of destinations in multicast sessions usually consist of a small number of the network nodes).

As the PoD increases, the average cost reduction that the new heuristics achieve, increases as well. Let $gain_i$ for multicast group size i be defined as:

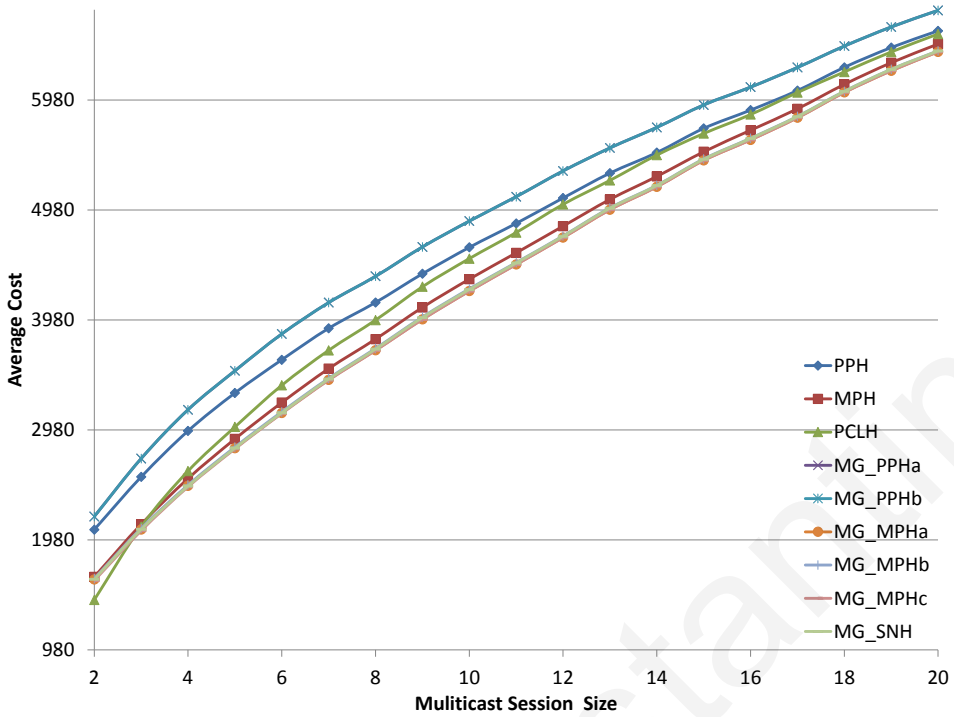


Figure 5.7: Average cost vs. multicast group size (PoD = 20%, small multicast sessions ($D \leq 20$)).

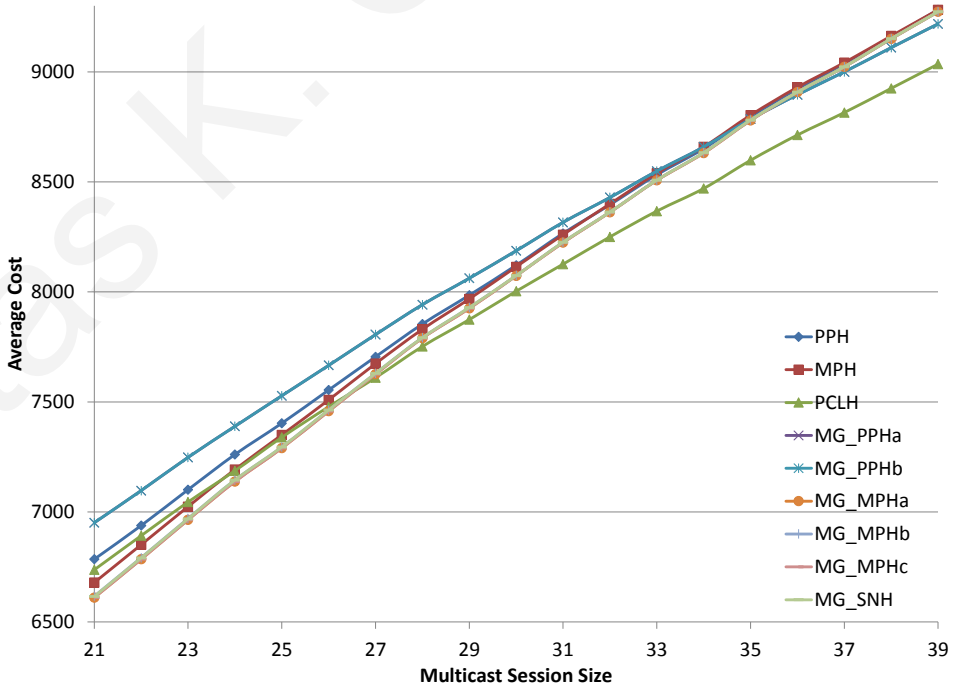


Figure 5.8: Average cost vs. multicast group size (PoD = 20%, large multicast sessions ($20 < D \leq 39$)).

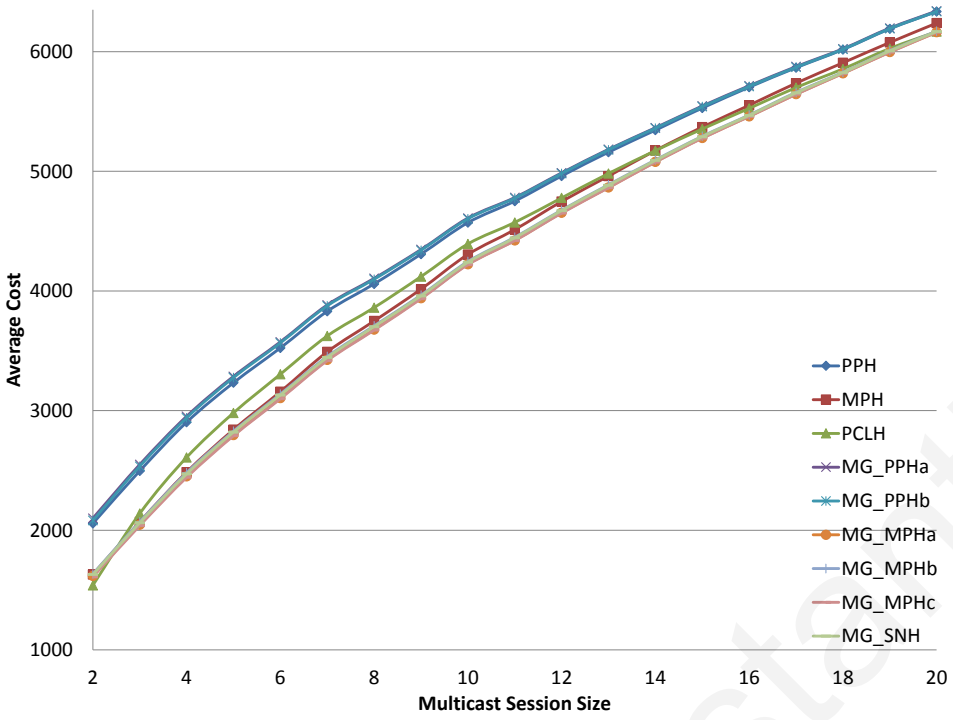


Figure 5.9: Average cost vs. multicast group size (PoD = 40%, small multicast sessions ($D \leq 20$)).

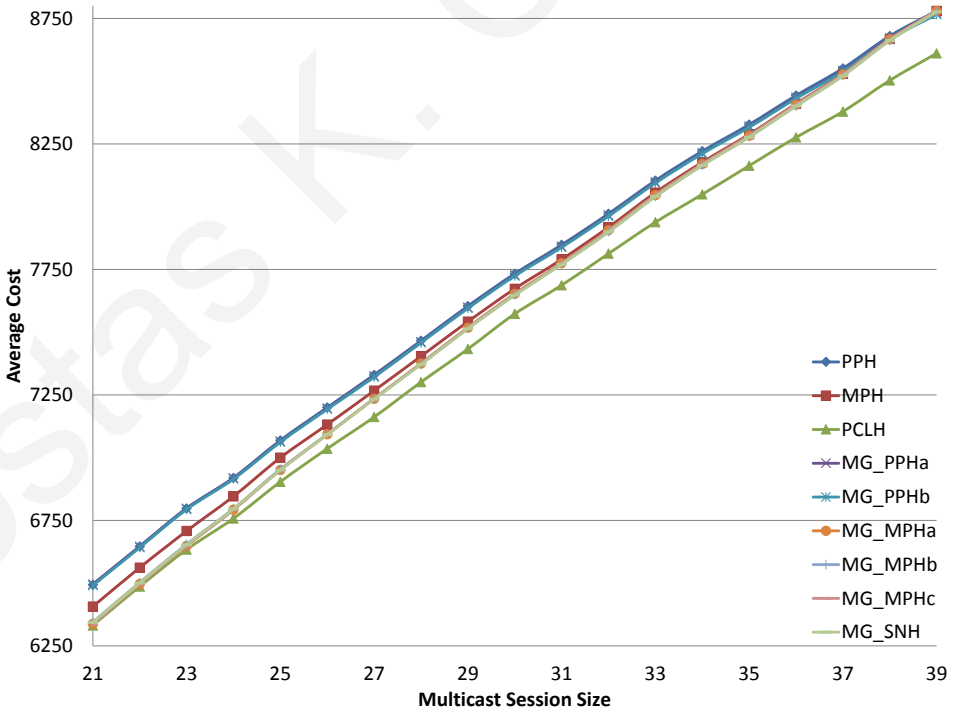


Figure 5.10: Average cost vs. multicast group size (PoD = 40%, large multicast sessions ($20 < D \leq 39$)).

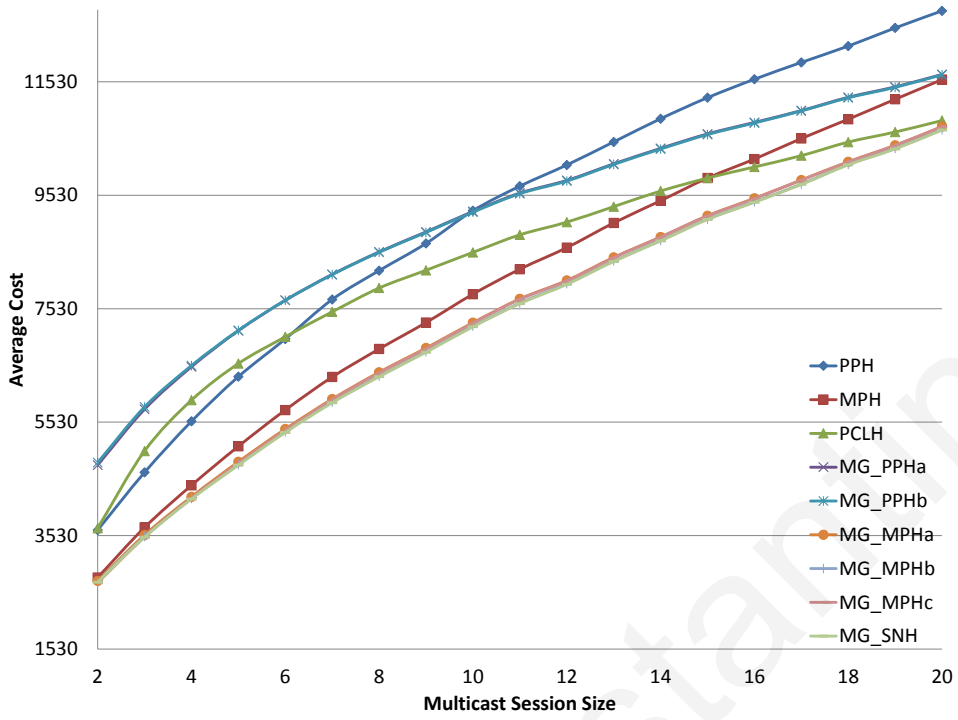


Figure 5.11: Average cost vs. multicast group size (PoD = 60%, small multicast sessions ($D \leq 20$)).

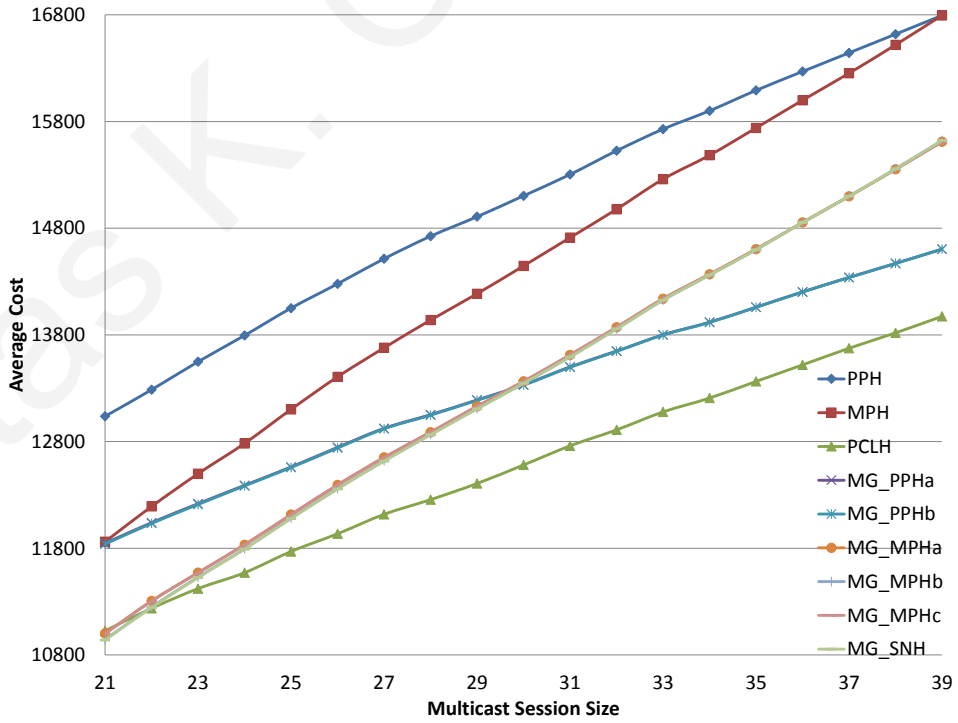


Figure 5.12: Average cost vs. multicast group size (PoD = 60%, large multicast sessions ($20 < D \leq 39$)).

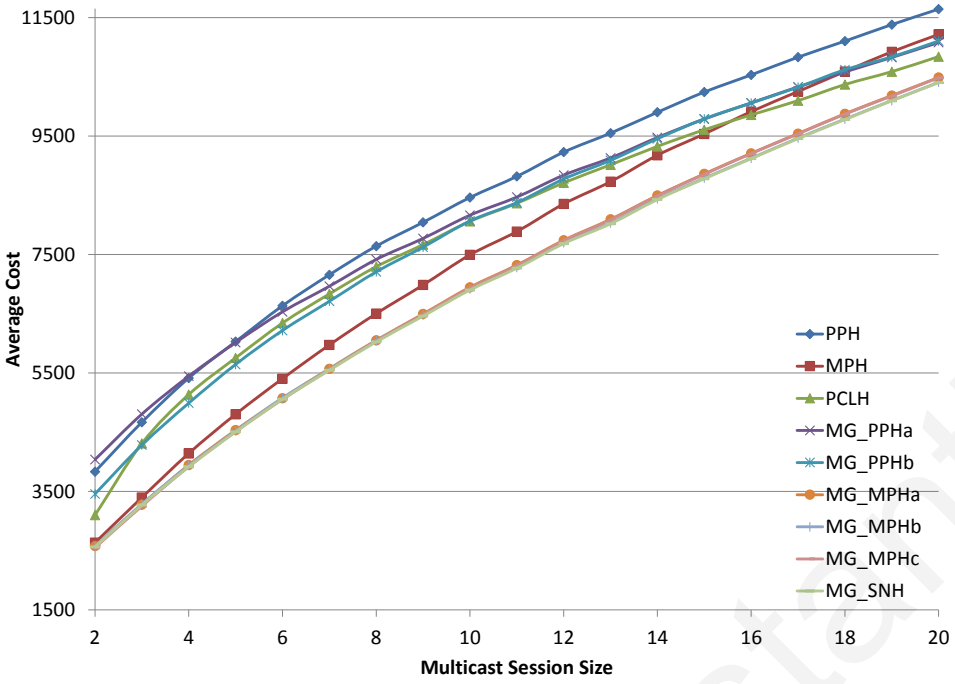


Figure 5.13: Average cost vs. multicast group size (PoD = 80%, small multicast sessions ($D \leq 20$)).

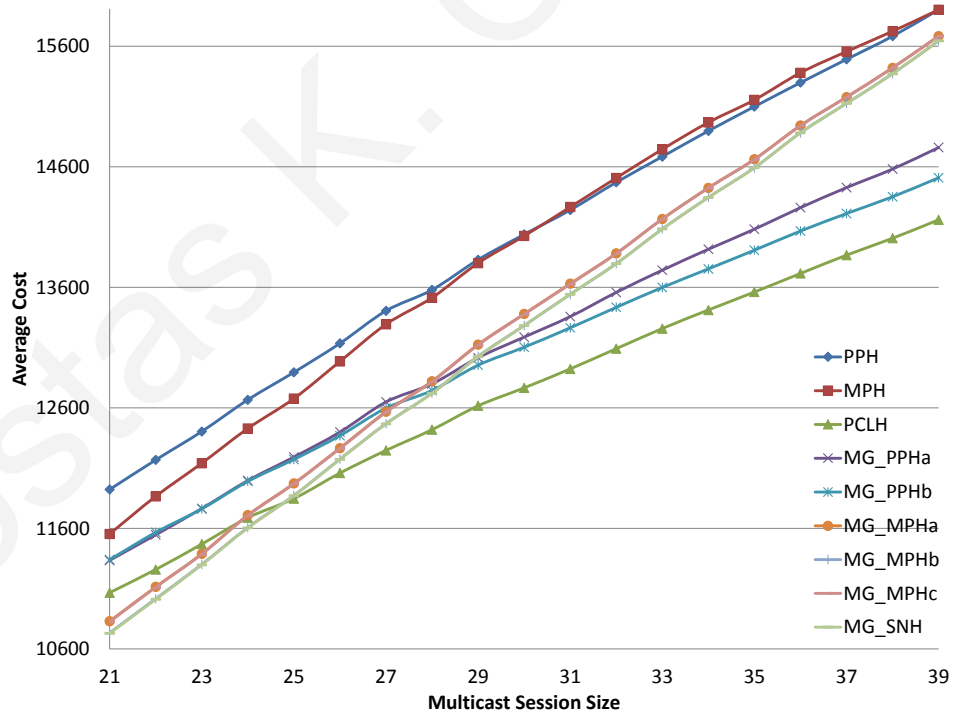


Figure 5.14: Average cost vs. multicast group size (PoD = 80%, large multicast sessions ($20 < D \leq 39$)).

$$gain_i = 100 \cdot \frac{min\{cost_{PPH}, cost_{MPH}, cost_{PCLH}\} - cost_{MG-SNH}}{min\{cost_{PPH}, cost_{MPH}, cost_{PCLH}\}}, \quad (5.2)$$

where $cost_X$ is the cost of the multicast tree obtained by heuristic X. Simply, $gain_i$ gives the cost reduction achieved by MG-SNH, compared to the best among the existing heuristics PPH, MPH, and PCLH. If $GAIN$ is defined as

$$GAIN = average\{gain_i\}, i = 2...\frac{M}{2}, \quad (5.3)$$

i.e., $gain_i$ is averaged over all possible multicast group sizes up to half of the network nodes, according to the above simulations, $GAIN \approx 6.5\%$ for the cases of $PoD = 60\%$ and 80%, and $GAIN \approx 1\%$ for the cases of $PoD = 20\%$ and 40%.

The reasons that MG-SNH and PCLH have the best performance for small and large multicast sessions respectively, are the following:

- SNH, as mentioned before, uses MPH as its basis and applies a recursive procedure in order to find the nodes that, if added into the destination set, give a tree with less cost. Therefore, it is more efficient compared to MPH, for both cases of undirected- and mixed-graph networks. Furthermore, MG-MPHb was specifically designed for mixed graphs, therefore it has enhanced performance in this category of networks, compared to MPH. Thus, since MG-SNH is the combination of the aforementioned algorithms, it is more efficient than SNH, and MG-MPHb, since it exploits the advantages of both.

- Chu Liu's algorithm gives optimal results for calculating minimum spanning trees in mixed graphs. Therefore, the multicast version of it (i.e., PCLH) has good performance for large multicast sessions, since in this case almost all network nodes are destinations and few arcs of the spanning tree are deleted in order to find the corresponding Steiner tree. However, as mentioned before this is a not such a practical case in real telecommunication network applications.

5.7 Conclusions

In this chapter, the existing as well as newly proposed mixed-graph multicast routing heuristic algorithms were presented, compared, and evaluated. It was shown

through simulations that the proposed heuristics outperform the existing ones in terms of average cost performance. For the network graphs used in the simulations, MG-SNH has the best performance for small multicast sessions; up to 8.1% average cost reduction, compared with the existing algorithms. For large multicast sessions, a case which is not very practical in real networks, PCLH is the most efficient as these multicast sessions tend to give (almost) spanning trees. The difference between MG-SNH (for small multicast sessions) / PCLH (for large multicast sessions) and the rest of the algorithms, becomes larger as the *PoD* of the graph increases. The worst-case performance of the widely-used multicast routing heuristic algorithms in optical networks was also theoretically calculated, to demonstrate the need for multicast routing heuristics that are specifically designed for mixed-graph networks.

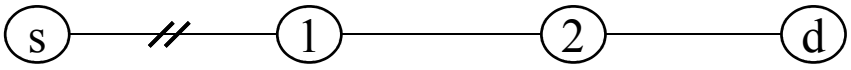
Chapter 6

Full-Splitting Undirected-Graph Networks: Multicast Protection Techniques

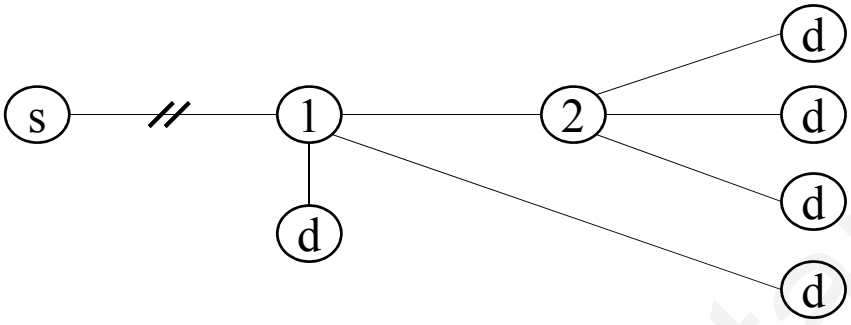
6.1 Protection of Optical Networks Against Failures

The vast amount of information that a fiber carries, as well as the amount of information loss in case of a failure on a light-tree that can affect the traffic to multiple destinations (Figure 6.1), have led to the necessity for the development of efficient multicast survivability techniques. These techniques are designed to provision multicast calls that are protected against any single link failure (e.g., an optical fiber cut), since this is the predominant form of failures in optical networks.

Two types of survivability techniques exist in optical networks, namely *protection* and *restoration*. *Protection* is the calculation of secondary ("backup") routes for every multicast request at the moment the request arrives (precomputed protection routes prior to the failure event), while *restoration* dynamically discovers a backup route after the link failure [69]. Several approaches were proposed in the literature for protecting multicast calls. These can be divided into *dedicated* and *shared* techniques, depending whether the redundant resources are dedicated or shared amongst several connections. Examples of dedicated protection are *link-disjoint* and *arc-disjoint* light-trees [23]. In these schemes one working and one protection tree are calculated for each multicast request, which have their links/arcs disjoint respectively. If there are sufficient resources for the assignment of these two trees for a multicast request,



Light-path: the failure of fiber S-1 affects only one destination



Light-tree: the failure of fiber S-1 affects five destinations

Figure 6.1: Lightpath and light-tree fiber failure.

then the request is realized. If not, it is blocked. Shared protection can be further divided into *self-sharing* and *cross-sharing*. These two cases can be found for example in [23] and [70] respectively. In the first case, the secondary tree of a multicast request can share arcs with the primary one. In the second case, secondary trees of different requests can share arcs. The advantage of these techniques is the minimization of the redundant capacity required in the network to protect the multicast sessions.

The performance metrics used in this thesis for the protection techniques are *blocking probability* (BP)) and *average cost* ($\overline{cost}$). Blocking probability is the fraction of the realized multicast requests over the total number of requests (percentage of blocking referred to here as blocking probability). The $cost_i$ of a realized multicast call i is in this case the cost of the primary and secondary routes for all its destinations (calculated in terms of link weights). Therefore, if x are the successful calls, the average cost is given by the following equation:

$$\overline{cost} = \frac{1}{x} \sum_{i=1}^{x} cost_i \tag{6.1}$$

The less BP and $\overline{cost}$ a technique gives, the more successful it is considered.

6.1.1 General Arc-Disjoint and Node-Disjoint Protection Techniques

The purpose of a large number of multicast protection schemes is the calculation of two disjoint paths from the source node to each destination node of the specific multicast request. If this is realizable, the multicast request will be serviced regardless of any single link failure in the network. One way to achieve this is by calculating two disjoint trees for every multicast request [23].

For the case of a single link failure, two Link-Disjoint Trees (*LDT*), or two Arc-Disjoint Trees (*ADT*) can be calculated to ensure the survivability of the network. Both of these protection techniques can protect the network from any single link failure. The former is omitted in this analysis, since it needs more network resources compared to the latter. A simple example where the efficiency of ADT is shown, is presented in Figure 6.2. In this example, link $d_1 - d_2$ is utilized by both the primary tree (arc $d_1 - d_2$) and the secondary tree (arc $d_2 - d_1$). In the case that this link fails (i.e., both arcs of it fail), the information can be sent to d_1 using the primary tree ($s - d_1$) and to d_2 using the secondary tree ($s - d_2$). Therefore, although the two trees are just arc- and not link-disjoint, the network can survive from link failures as well.

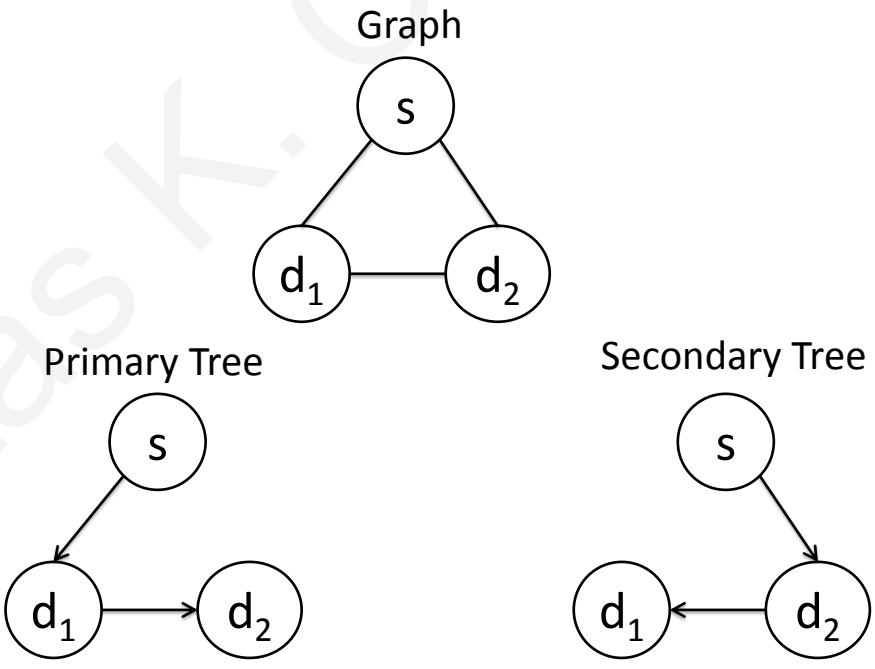


Figure 6.2: Example of two arc-disjoint trees.

For the case of a single link/node failure, two Node-Disjoint Trees (*NDT*) must be calculated to ensure the survivability of the network. The two trees must not share

any arcs, or intermediate nodes and their corresponding arcs. (An intermediate node is every node that is a part of the light-tree and it is not either the source or the destination. The corresponding arcs of a node are the arcs that originate from or end at this node.)

The general procedure for the calculation of two arc-disjoint or node-disjoint trees is as follows:

1. Calculate the primary light-tree using a multicast routing heuristic algorithm.
2. - *For arc-disjoint trees:* Remove the arcs of the primary tree from the network graph.
   - *For node-disjoint trees:* Remove the arcs of the primary tree from the network graph, its intermediate nodes and the corresponding arcs.
3. Calculate the secondary tree on the resulting graph using a multicast routing heuristic algorithm.

The selection of the multicast routing heuristic algorithm is critical; if an algorithm that manages to calculate light-trees with less arcs is utilized, after the calculation of the primary tree the resulting graph will have more arcs available, and this will increase the possibility that a secondary tree will be found on this resulting graph. Therefore, the average blocking probability will subsequently be decreased and the average cost (in terms of number of arcs for the pair of primary and secondary trees) will be decreased as well.

Also note that for the case of a node failure, it is assumed that for the source and the destination set a malfunction never occurs. To protect the network against those failures, ensuring that source and destination node failures are also recoverable, requires redundancy (and "dual-homing" techniques) for every network node.

6.1.2 Existing and Proposed ADT- and NDT-Based Protection Techniques

As described in Chapter 3, the two widely-used multicast routing heuristic algorithms in the literature are the *Pruned Prim Heuristic* (PPH) [45] and the *Minimum Path Heuristic* (MPH) heuristics [15]. If these heuristics are exploited for the calculation of the pair of the disjoint trees, the resulting arc-disjoint techniques are PPH-ADT and MPH-ADT and the node-disjoint ones are PPH-NDT and MPH-NDT.

A more efficient multicast routing heuristic algorithm is the *Steiner Node Heuristic* (SNH), also presented in Chapter 3. This heuristic outperforms PPH and MPH in terms of the cost of the calculated multicast tree. Therefore, after the calculation of the primary tree using SNH, the resulting graph will have more arcs available compared to the case PPH or MPH were used, and this will increase the possibility that a secondary tree will be found on this resulting graph. Therefore, the average blocking probability and the average cost will decrease. If SNH is exploited in the aforementioned protection method, the resulting techniques are called SNH-ADT and SNH-NDT for the case of arc and arc/node failures respectively.

6.1.3 Performance Evaluation

The performance of the existing and proposed protection methods was evaluated via simulations, using the sample network (USNet) shown in Figure 6.3, consisting of 24 nodes and 43 bidirectional links. The cost of each link is denoted in the figure. This cost may be the actual length of the link or any other cost assigned for it (e.g., latency, port cost, etc).

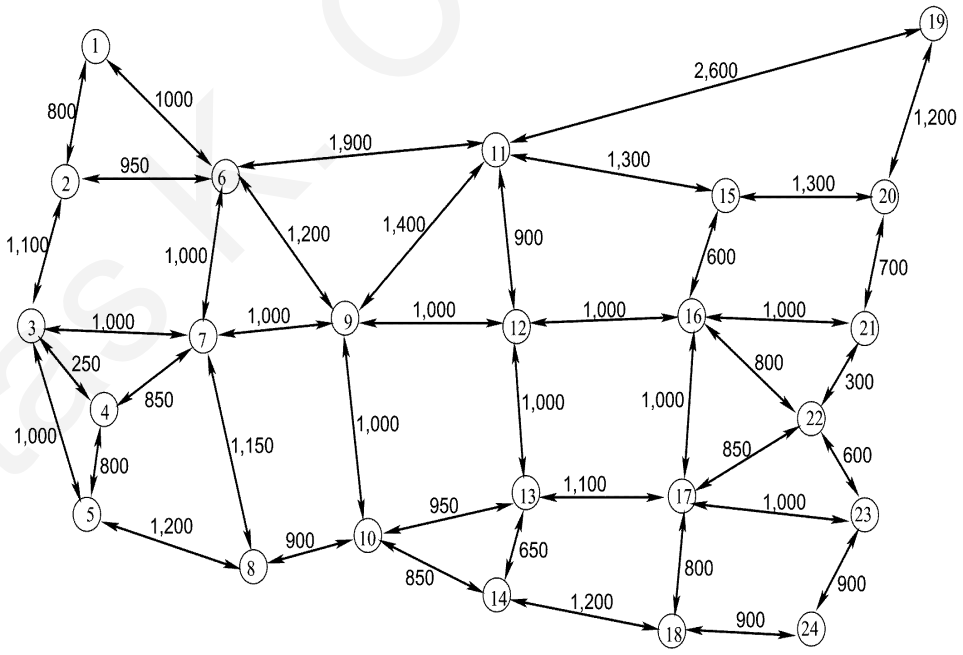


Figure 6.3: Test network used for performance evaluation.

Let $D = k - 1$ be the number of destinations of the multicast group that must be established. The experiment is executed for all possible multicast groups (from

$D = 1$ (unicast call) to $D = 23$ (broadcast call)) using the PPH-ADT, MPH-ADT, and SNH-ADT techniques to calculate the pair (working and protection) of the arc-disjoint trees. For each multicast group case, the simulation is repeated 5000 times, while the source and destinations of each connection are randomly generated and are distributed uniformly across the network. Each multicast call was assumed to depart from the network before the arrival of the following ones. A multicast request is established, if a pair of working and protection arc-disjoint trees can be found for that request, otherwise it is blocked. (It is possible to fail to find a protection tree when the removal of the arcs of the primary one leads to a disconnected network graph.)

The blocking probability and the average cost (in terms of link weights utilized for the working and protection trees) for the three arc-disjoint multicast protection techniques are presented in Figures 6.4-6.5 respectively, for different multicast group sizes (a multicast group as shown in the plots does not include the source node). For the calculation of the average cost, only the non-blocked demands are considered.

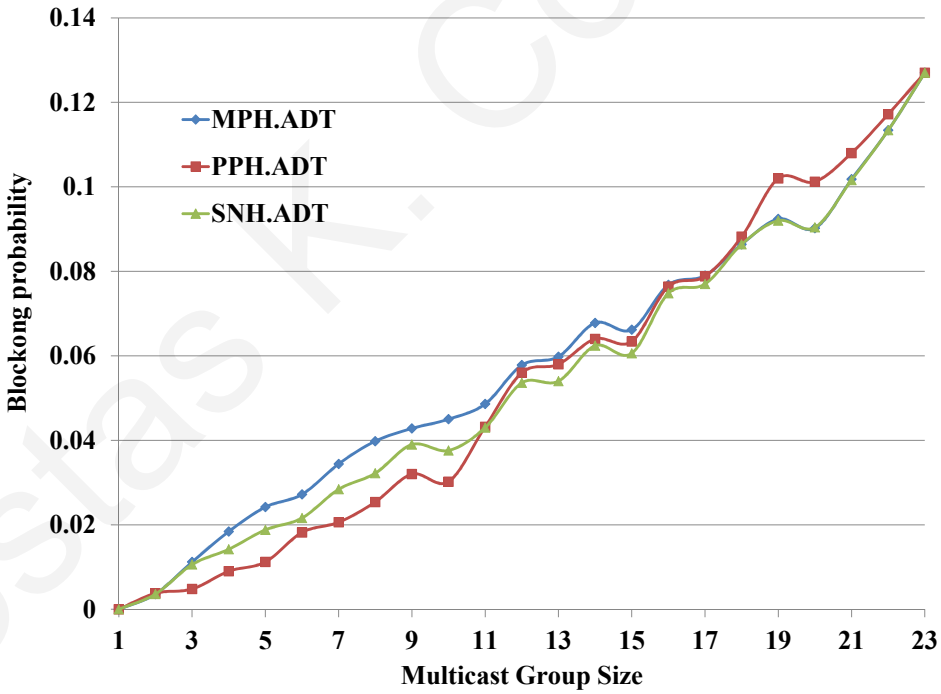


Figure 6.4: Blocking probability for the MPH-ADT, PPH-ADT, and SNH-ADT multicast protection techniques.

As it can be seen from the performance results, for the single link failure case the

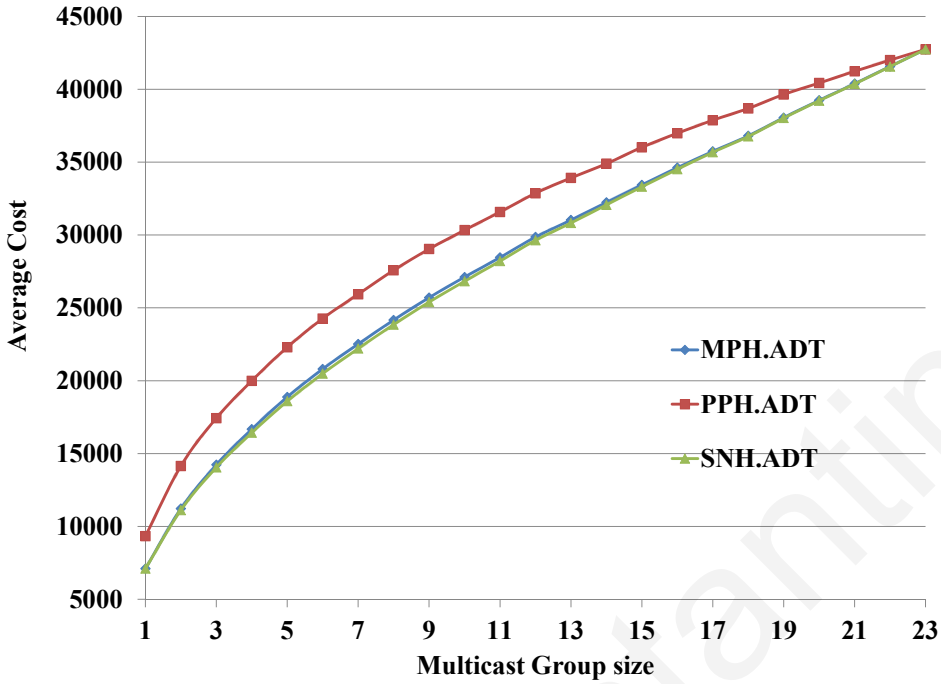


Figure 6.5: Average cost for the MPH-ADT, PPH-ADT, and SNH-ADT multicast protection techniques.

proposed SNH-ADT multicast protection method gives lower blocking probability compared to the MPH-ADT technique and lower blocking compared to the PPH-ADT approach for larger multicast group sizes. Also, it is always much better than the PPH-ADT technique and always equal or better than the MPH-ADT approach in terms of the average cost. Thus, the SNH-ADT technique will be preferred over both existing techniques when both the blocking probability and average cost are taken into consideration.

The blocking probability and the average cost (in terms of link weights utilized for the working and protection trees) for the three node-disjoint multicast protection techniques are presented in Figures 6.6-6.7 respectively. For this case (i.e., single link/node failure), the SNH-NDT technique always outperforms the PPH-NDT and MPH-NDT approaches in terms of blocking probability, while it has slightly better results compared to MPH-NDT and significantly better results compared to PPH-ADT, in terms of average cost.

The enhanced results of SNH-ADT and SNH-NDT are a direct consequence of the proposed SNH Steiner tree routing heuristic algorithm that has increased cost

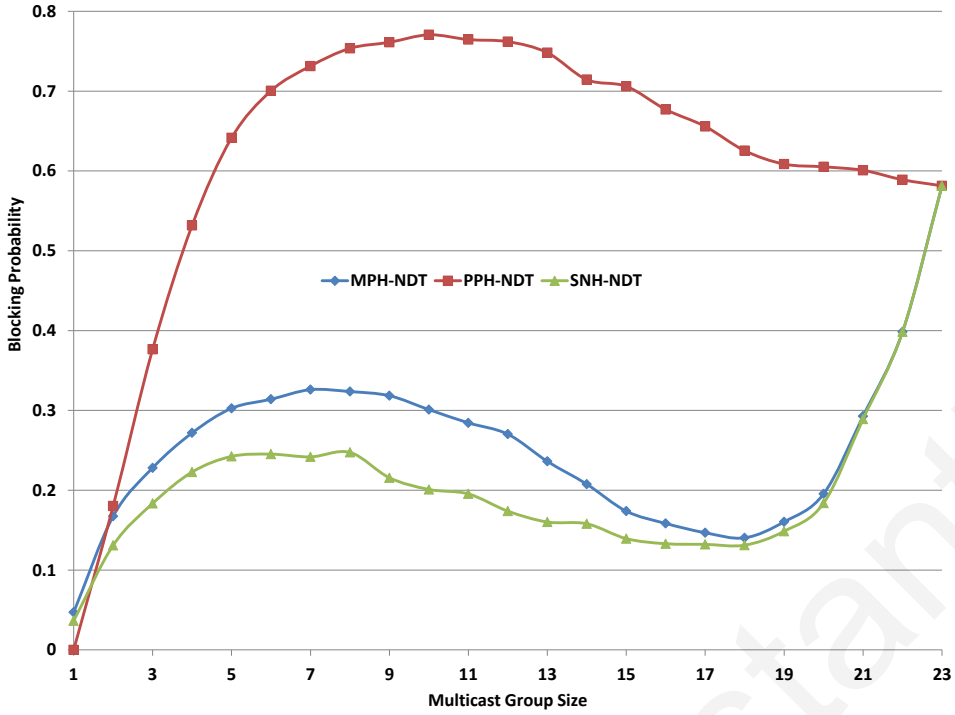


Figure 6.6: Blocking probability for the MPH-NDT, PPH-NDT, and SNH-NDT multicast protection techniques.

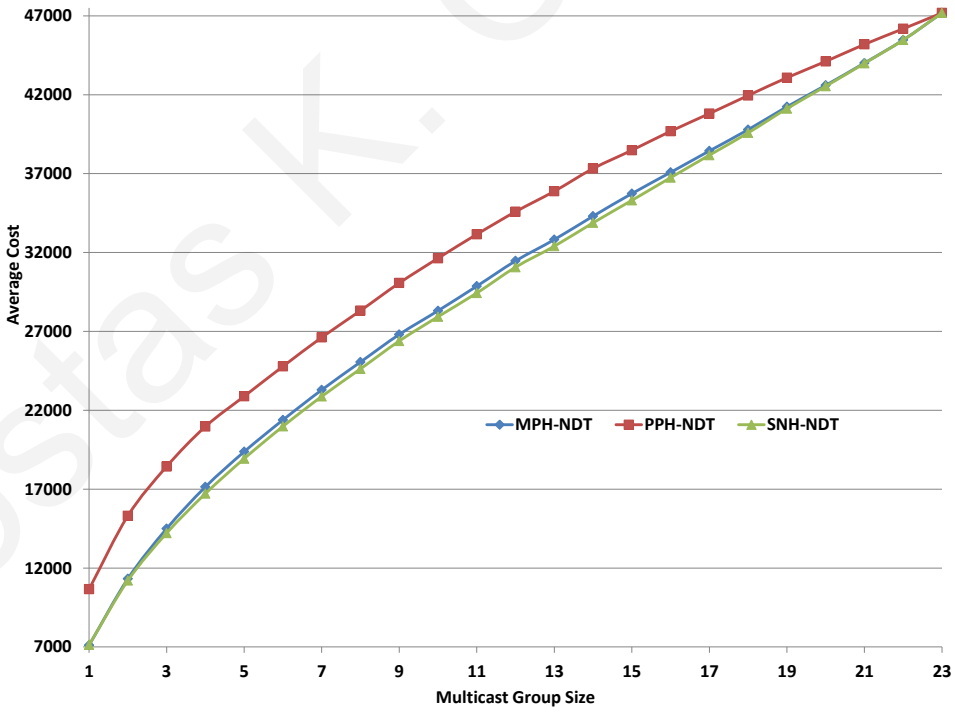


Figure 6.7: Average cost for the MPH-NDT, PPH-NDT, and SNH-NDT multicast protection techniques.

performance compared to the PPH and MPH heuristics.

The form of the graph of Figure 6.6 (i.e., it has turning points) is due to the fact that the source and the destination sets are considered to always work properly (i.e., never fail). Therefore, if the multicast group size is increased, more nodes are considered not to fail, and the blocking probability is decreased. When the multicast group size becomes even larger, it is less possible for the secondary tree to exist, even with the removal of just the arcs of the primary tree. Therefore the blocking probability increases again.

6.1.4 Conclusion

An improved protection technique (SNH-ADT, SNH-NDT), is presented in this section. Simulations have shown that it outperforms the existing ones in terms of blocking probability and average cost, for both single link and single link/node failure scenarios.

6.2 A Load Balancing Technique for Protecting Dynamic Multicast Calls in Mesh Optical Networks

6.2.1 Introduction

There is only very limited work in the literature that takes into account the current status of the network for the calculation of the primary and secondary trees for the upcoming requests [71], [72]. Most of the existing multicast protection methods ignore the distribution of the network load and the congestion it causes to the network connections. This may lead to the routing of several requests through certain arcs, something that eventually causes disconnection to some areas of the network (due to the complete utilization of resources on some links) and increases the blocking probability for the upcoming requests. An efficient way to overcome this weakness is to use a *load balancing* technique in order to intersperse the routing of the multicast requests in the network. This can be achieved by modifying the cost of the network arcs according to the change of flow of traffic through them after the establishment of each successful multicast request.

A very efficient and well known heuristic algorithm for provisioning survivable

multicast connections in mesh optical networks is the Optimal Path-Pair Shared Disjoint Paths (OPP-SDP) technique [23]. Therefore, this is the heuristic selected in order to be combined with the load balancing technique developed in this section. The resulting technique is called *Load Balancing Optimal Path-Pair Shared Disjoint Paths* (LB-OPP-SDP). Simulations show that the proposed technique has enhanced performance in terms of blocking probability (50% lower average blocking probability) under the scenario of dynamic multicast call arrivals, when compared with the OPP-SDP approach.

6.2.2 Load Balancing Optimal Path Pair Shared Disjoint Paths

The Optimal Path Pair Shared Disjoint Paths (OPP-SDP) method, has a major advantage over the arc-disjoint techniques discussed in the previous section, as it can handle the cases of trap topologies (as shown in Figure 6.8), where the arc-disjoint techniques will give poor results. (In this figure, an example with one destination is given. The case of multicasting is the generalization of this topology for more destinations.) In this example, s is the source node and d is the destination node, the first topology is the network graph, the second figure shows the first path between s and d and the third figure shows that, after the removal of the first path, a second one does not exist, i.e., the arc-disjoint method would not work for this topology.

This case can be handled using *Suurballe's* algorithm [27]. If this algorithm is applied to the aforementioned network, two disjoint paths can be found, as shown in Figure 6.9.

The characteristic that makes OPP-SDP multicast protection method outperform other existing techniques is that, by utilizing Suurballe's algorithm, it can efficiently calculate together both the (arc-disjoint) working and protection paths, managing also networks that include "trap topologies", where simpler protection methods that find the working and protection paths sequentially would be more inefficient or even fail (as the removal of the working path prior to the calculation of the protection path would result in a disconnected graph, thus precluding the calculation of a protection path) [23].

The OPP-SDP heuristic algorithm consists of the following steps:

1. For every destination node of the session, repeat Steps 2 and 3.

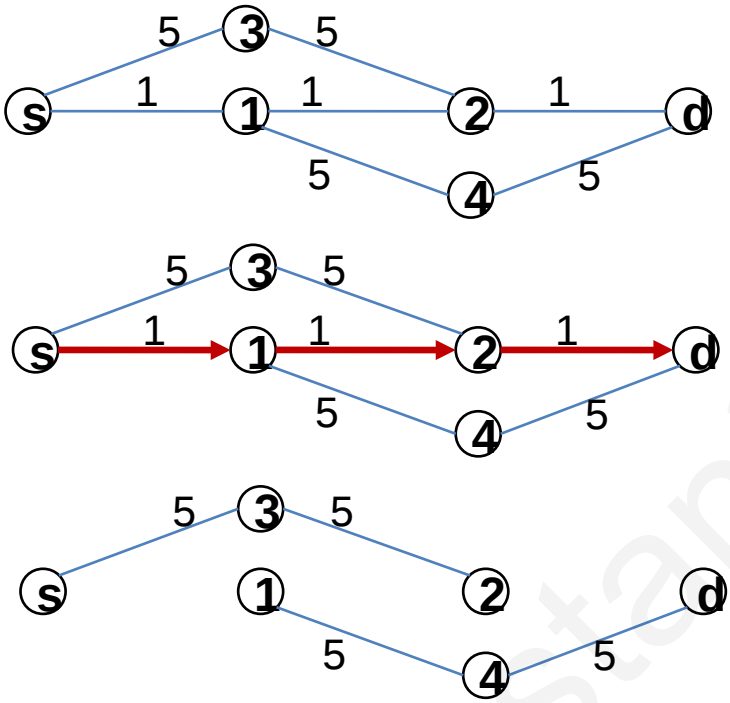


Figure 6.8: Failure of ADT multicast protection method.

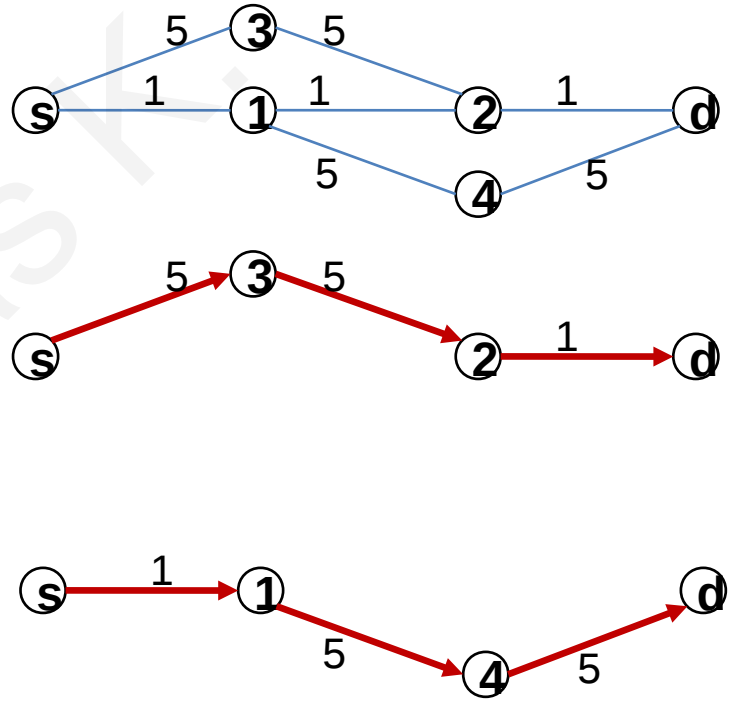


Figure 6.9: Pair of disjoint paths found using Suurballe's algorithm.

2. Find an optimal path-pair between the source and destination node using Suurballe's algorithm.

3. Update cost= 0 for already found optimal path pairs.

Although this heuristic is very efficient, for the realization of a new multicast request it does not take into account the change of the network state, due to resource holding of previous requests. The proposed Load Balancing Optimal Path-Pair Shared Disjoint Paths (LB-OPP-SDP) heuristic algorithm was designed specifically to overcome this weakness. It essentially uses *load balancing* in order to intersperse the routing of the multicast requests in the network by modifying the cost of the network arcs depending on the flow of traffic on them at any instance (i.e., depending on the network state in terms of routed connections when a new multicast connection arrives and must be provisioned in the network). It consists of the following steps for every arriving multicast request:

LB-OPP-SDP Heuristic

Given a connected graph $G(V, E)$ where each element of V is network node and each member of E is a network edge, with weights for every arc (unidirectional edge) (i, j) denoted by $cost[i][j]$.

- The $indegree(v)$ ($outdegree(v)$) of vertex v, $id(v)$, ($od(v)$) corresponds to the in-bound (outbound) edges incident to vertex v.

- Define $degree(v)$ as the $min\{id(v), od(v)\}$.

1. Calculate the $id(v)$ and $od(v)$ of each network node v, considering only the arcs that have free (available) wavelengths.

2. Calculate the $degree(v)$ of each network node v, considering only the arcs that have free (available) wavelengths.

3. For each network arc that has free wavelengths (corresponding cost is denoted by $cost[i][j]$, where i is the node that is the origin of it and j is the ending node) and $degree(i) \cdot degree(j) > 0$:

    - If $degree(i) > degree(j)$ then $cost[i][j] = cost[i][j]/degree(j)$
    - If $degree(i) < degree(j)$ then $cost[i][j] = cost[i][j]/degree(i)$

4. For every destination node of the session, repeat Steps 5 and 6.

5. Find an optimal path pair between the source and destination node utilizing Suurballe's algorithm [27].

6. Update *cost* = 0 for already found optimal path-pairs.

The proposed heuristic algorithm works as follows: Before the application of OPP-SDP (i.e., before the calculation of the optimal path-pair between the source and each destination) the network graph is modified in order to achieve a balanced distribution of the load. More precisely, the Maximum Possible Flow (MPF) through a network node is determined by the minimum between the numbers of incoming and outgoing arcs (only arcs with free wavelengths are considered). This is calculated for every node, and it is named in this algorithm as the *degree* of the node (Step 2). The MPF through an arc is determined by the MPFs of the associated nodes (the minimum of them is selected as the MPF for the arc). The modified cost of the arc is the initial cost divided by the arc's MPF. In this way, the cost of an arc is increased inversely to its MPF (Step 3). Therefore, the arriving requests are more possible to be routed through arcs with higher MPF, and the possibility of a disconnection of a network area due to load concentration, is decreased. Finally, OPP-SDP is applied to the modified network graph (Steps 4-6).

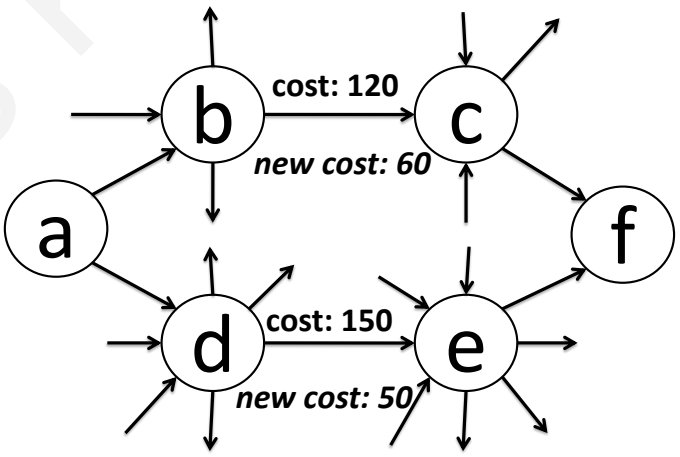


Figure 6.10: Example of the LB-OPP-SDP heuristic.

An example of the proposed heuristic is given in Figure 6.10. Suppose that there is a request with node *a* as the source node and node *f* as the destination node.

Node	*id*	*od*	*degree*
b	**2**	**3**	**2**
c	**3**	**2**	**2**
d	**3**	**4**	**3**
e	**4**	**4**	**4**

Figure 6.11: Example of the LB-OPP-SDP heuristic.

The assumption in this example is that arcs $a-b$, $c-f$, $a-d$, and $e-f$ have the same cost. In this case, OPP-SDP would select path $a-b-c-f$ since $b-c$ has lower cost compared to $d-e$. If LB-OPP-SDP is applied instead, the cost of the two arcs will be modified. For arc $b-c$, $degree(b) = min\{id(b), od(b)\} = min\{2,3\} = 2$, $degree(c) = min\{id(c), od(c)\} = min\{3,2\} = 2$ (Figure 6.11), therefore the new cost for arc $b-c$ will be $120/2 = 60$. Similarly, the new cost for arc $d-e$ will be 50. Therefore, path $a-d-e-f$ will be selected if the LB-OPP-SDP heuristic was implemented. This selection is better, since, for the current network status, arc $d-e$ has higher MPF and it is less probable to have load congestion (and disconnection) at this arc compared to arc $b-c$. The advantage of the proposed heuristic is that, for the calculation of the multicast tree, it takes into account the routing of the already established multicast requests, a critical factor that OPP-SDP ignores.

Complexity of the LB-OPP-SDP Heuristic

Suurballe's algorithm has time complexity of $O(n^2 \log n)$ for a graph consisting of n nodes [27]. Since it is applied for every destination, the time-complexity of LB-OPP-SDP is equal to $O(Dn^2 \log n)$, where D is the number of destinations.

6.2.3 Performance Evaluation

The performance of the LB-OPP-SDP multicast protection technique was evaluated and compared with OPP-SDP. The simulations were performed on one randomly created undirected graph with 35 nodes and 100 links. This graph was simple (i.e., it

had no self-loops or multiple edges having the same pair of end-vertices), therefore all links connected different pairs of nodes. For this graph, a constraint was used for the graph creation, namely that every arc that was added in the graph had to connect nodes that satisfy $d_{nom}^{ij} < 6$ for all i, j. The simulation was repeated for various possible multicast group sizes (D), from $D = 2$ to $D = 34$ (D stands for the number of destinations of the multicast group). The simulation was executed 25000 times for every multicast group size, while the source and the destinations of the multicast connections were distributed uniformly across the network. The simulation was performed for multicast calls arriving dynamically, i.e., they were considered to arrive, stay for some time, and then depart. The time of arrival was generated according to a poisson distribution with $\lambda_{poisson} = 100$ and the connection holding time was exponentially distributed with $\lambda_{exp} = 1$ (setting the network load to 100 *Erlangs*). Every fiber was considered to have 64 wavelengths and wavelength conversion was assumed at every network node. If a pair of paths that originated from the source and ended at a destination was able to be calculated for every destination in the multicast call, the call was considered to be successful. Otherwise, it was considered as unsuccessful and the entire multicast call was blocked. The blocking probability (BP) was calculated as the ratio of blocked/total multicast requests, for every multicast group size. The results of the simulation for both the OPP-SDP and LB-OPP-SDP techniques are presented in Figures 6.12 and 6.13. The cases of small (up to 17 destinations) and large (18 destinations and more) multicast group sizes were presented in different graphs for clarity purposes.

As shown from the results in Figures 6.12 and 6.13, the LB-OPP-SDP technique significantly outperforms the OPP-SDP approach for small multicast sessions (up to approximately half of the network nodes). It must be noted that this is a more realistic case, since it is highly unlikely that there will be multicast requests that will have as destinations more than the half of the network nodes. The average BP reduction for small multicast sessions (consisting of 6 to 17 destinations, as the cases with less than 6 destinations are not considered for the calculation of average BP since both heuristics have BP = 0) is approximately 50%. For large multicast sessions, both techniques have the same performance (LB-OPP-SDP performs slightly better for cases up to 24 destinations whereas OPP-SDP performs slightly better for multicast sessions of 25 destinations and more) as in those cases the limiting factor is not the design of the multicast protection technique but rather the network resources

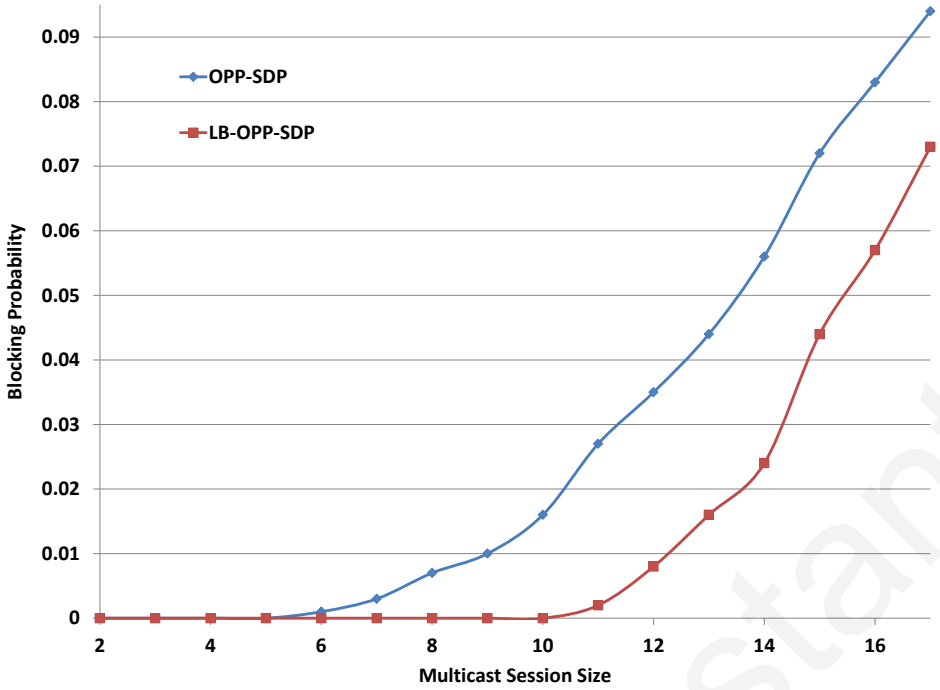


Figure 6.12: Blocking probability for small multicast group sessions.

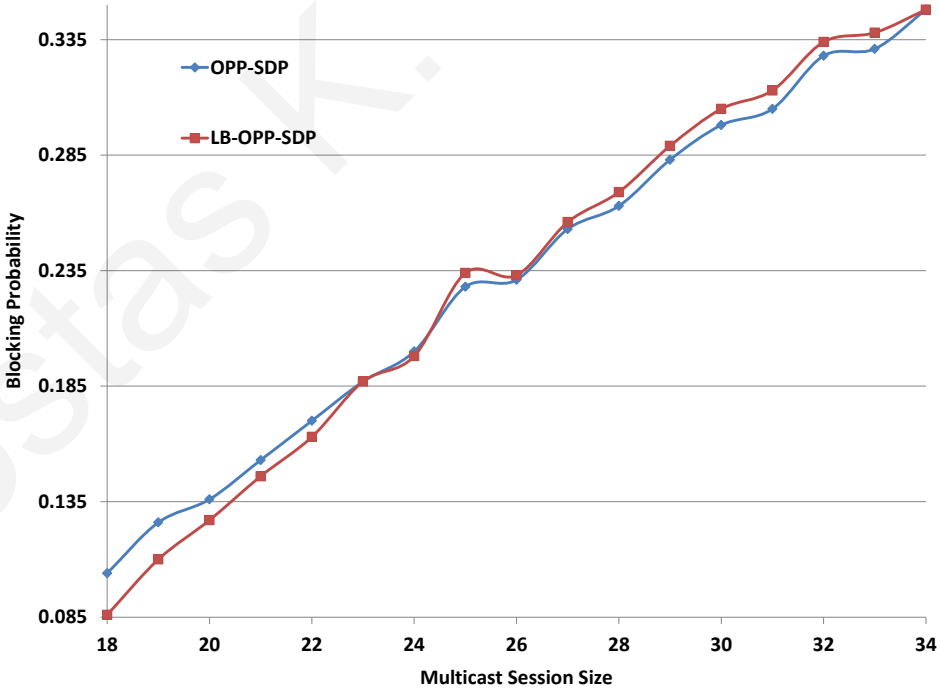


Figure 6.13: Blocking probability for large multicast group sessions.

(number of wavelengths per fiber in the network under consideration).

6.2.4 Conclusions

A new multicast protection technique, *Load Balancing Optimal Path-Pair Shared Disjoint Paths* (LB-OPP-SDP) is presented in this section. It is the combination of the very efficient and well-known OPP-SDP multicast protection heuristic, and a load balancing technique that recalculated the cost of each network arc according to the load distribution on the network due to the already established multicast requests. Simulations have shown that the proposed method outperforms OPP-SDP in terms of blocking probability (50% lower average blocking probability) for small multicast group sizes. Note that the proposed load balancing technique can also be combined with any other protection method as well. For example, it can be combined with the arc-disjoint-trees (ADT) and node-disjoint-trees (NDT) protection methods, in order to enhance the performance of these techniques.

Chapter 7

Full-Splitting Mixed-Graph Networks: Multicast Protection Techniques

7.1 Introduction

In Chapters 3 and 6, the heuristic algorithms that are mainly used in the literature for the calculation of the primary and secondary arc-disjoint light-trees (*Pruned Prim Heuristic (PPH)* and *Minimum Path Heuristic (MPH)*) and their resulting protection techniques (*PPH-ADT* and *MPH-ADT* respectively) were described [23]. However, as discussed in Chapter 5, these multicast routing heuristic algorithms were designed for undirected graphs, and although they are able to find a tree in mixed graphs as well, they do not perform well for this category of graphs (in terms of cost of the calculated tree, i.e., number of arcs). This is demonstrated, for example, in the simple test network given in Figure 7.1. Here, node s is the source and nodes d_1 and d_2 are the destinations. The MPH and PPH heuristics give a tree of cost equal to 7, while the minimal multicast tree has cost equal to 6.

Furthermore, as previously motivated in Chapter 5, even if the network is designed to be undirected, when some multicast requests arrive and hold resources of the network, the resulting network graph is mixed, therefore the calculation of multicast trees for the new requests, will be calculated on a mixed graph. Also, for the case of protected connections, even in the case that the two trees are calculated on an undirected graph, the arcs of the working tree are removed from the graph for the calculation of the protection tree, and the latter is calculated on the resulting mixed graph. Therefore, it is clear that the development of new protection algorithms that

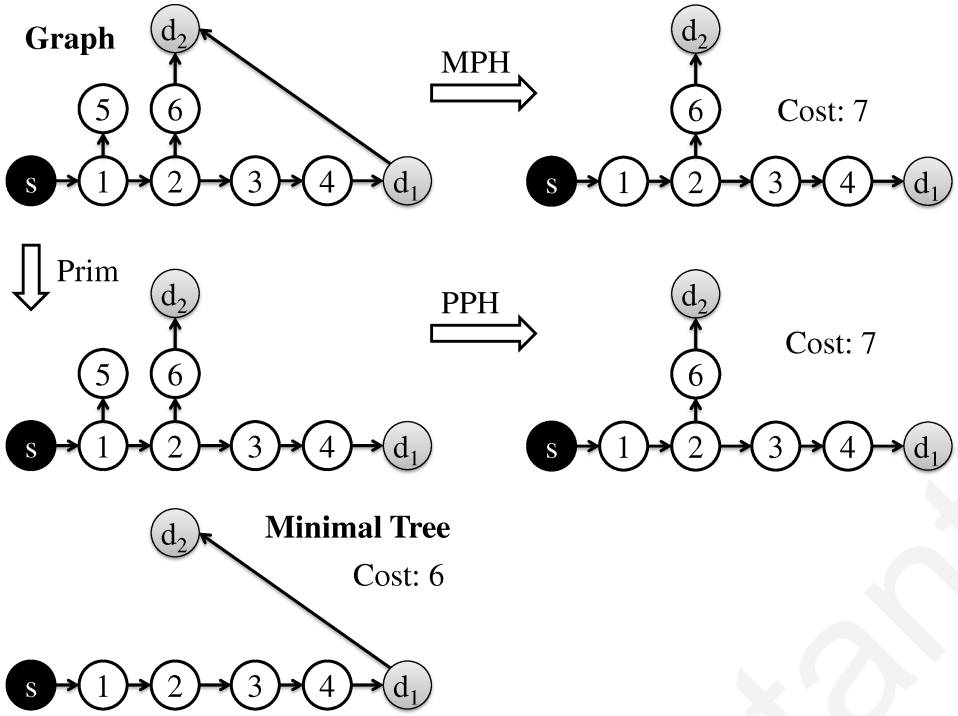


Figure 7.1: MPH and PPH heuristics.

are designed to work efficiently for mixed-graph networks, is very important.

7.2 Proposed Arc-Disjoint Tree Multicast Protection Heuristic Algorithms

7.2.1 MG-MPH-ADT Protection

Chapter 5 introduced a new multicast routing heuristic algorithm that was developed specifically for mixed graphs and outperforms the aforementioned PPH and MPH heuristics for the calculation of multicast light-trees in mixed-graph networks (called *MG-MPH*). For example, the MG-MPH heuristic is applied in graph of Figure 7.1 and its steps and result are shown in Figure 7.2. Note that contrary to the MPH and the PPH heuristics the MG-MPH heuristic succeeds in finding the minimum cost tree.

This multicast routing heuristic algorithm that is more efficient for routing in mixed graph networks can then be utilized in the arc-disjoint tree protection method for the protection against single arc failures in mixed-graph networks. The resulting protection method is called *MG-MPH-ADT* and consists of the following steps:

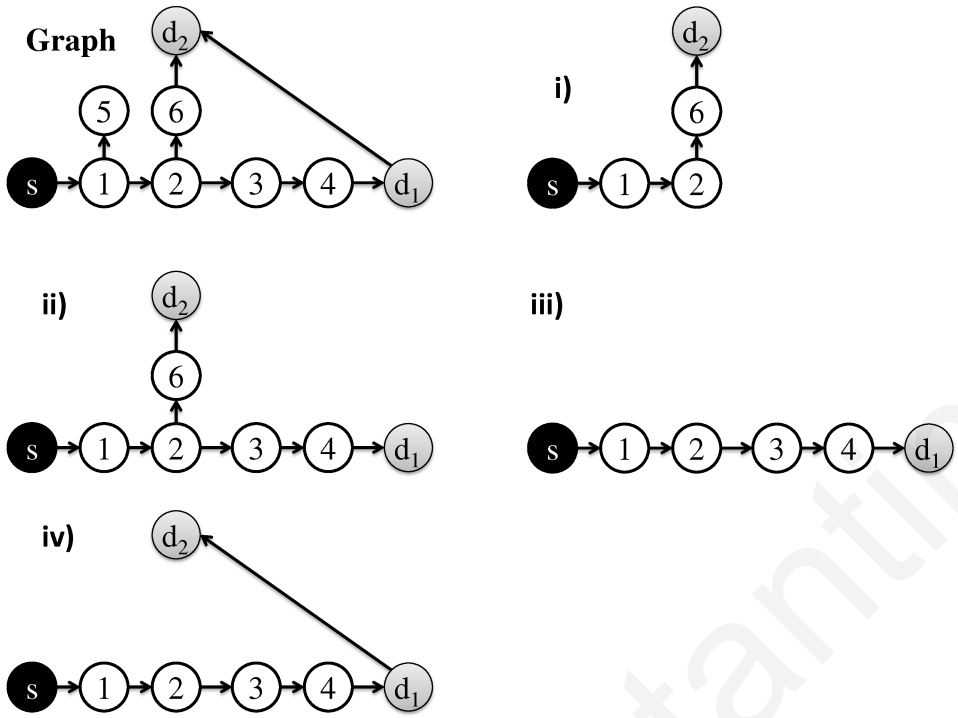


Figure 7.2: MG-MPH heuristic algorithm.

1. Calculate the primary light-tree using the MG-MPH heuristic algorithms described in Chapter 5.

2. Remove the arcs of the primary tree from the network graph.

3. Calculate the secondary tree on the resulting graph using the MG-MPH heuristic algorithm.

7.2.1.1 Performance Evaluation

The performance of the MG-MPH-ADT approach is evaluated and compared with the existing relevant protection techniques (MPH-ADT and PPH-ADT). The evaluation is performed via simulations on a mixed graph that was randomly created. In the first step, the graph had 40 nodes and 200 arcs. This graph should be simple, therefore all of the 200 arcs connected a different pair of nodes. There were no arcs with opposite orientation (the graph initially was directed). The constraint that every arc that was added in the graph had to connect nodes that satisfy $d_{nom}^{ij} \leq 4 \; \forall i, j$, was used for the random graph creation. According to the *Percentage of Directionality (PoD)* (i.e., the ratio *arcs*/*edges*) the appropriate number of arcs (unidirectional edges) were converted to links (bidirectional edges). In this simulation, the graph was selected to have PoD = 80%, therefore 160 arcs were chosen randomly, to be

converted to links, i.e., the opposite arc of each one of them was added to the graph. The procedure was repeated for various possible multicast group sizes (D), from $D = 2$ to $D = 38$, with a step equal to 2. The experiment was executed 5000 times for every multicast group size, while the source and destinations of the multicast connections were distributed uniformly across the network. 5000 pairs of arc-disjoint trees were calculated with each one of the existing and proposed multicast protection techniques (MPH-ADT, PPH-ADT, and MG-MPH-ADT). The cost criterion for the routing heuristic algorithms was the number of arcs that were used for the primary and secondary trees. The blocking probability and the average cost (i.e., the sum of the arcs of the two disjoint trees) were calculated for each protection method. The results are presented in Figures 7.3 and 7.4.

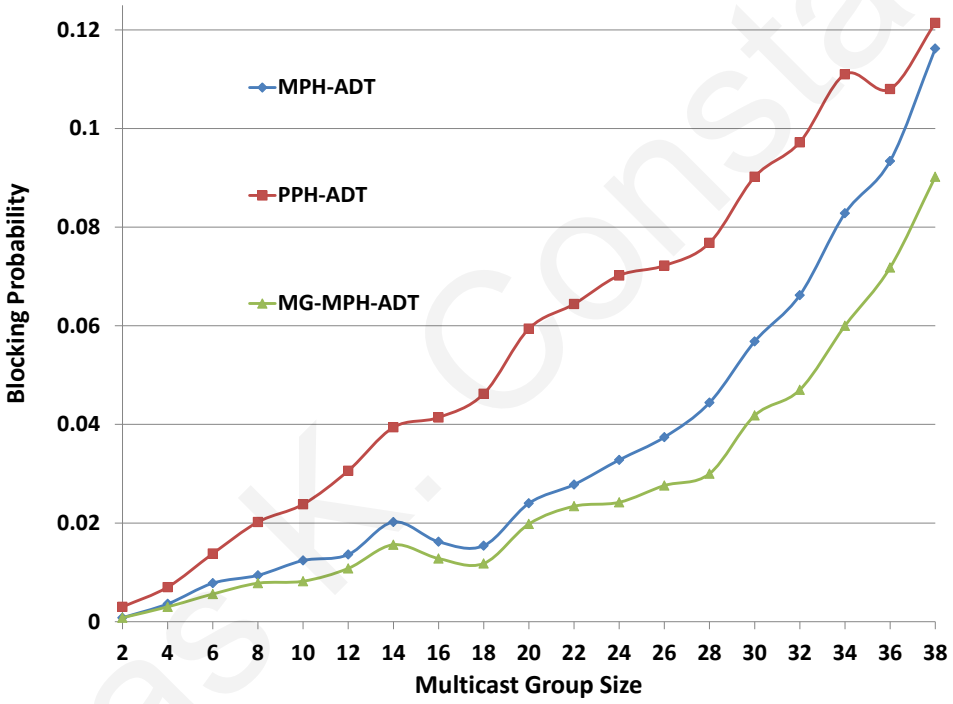


Figure 7.3: Blocking probability vs. multicast group size for the three different multicast protection techniques (MPH-ADT, PPH-ADT, and MG-MPH-ADT).

Analysis of the results

Simulations show that the MG-MPH-ADT multicast protection algorithm, based on the MG-MPH mixed-graph multicast routing heuristic algorithm, outperforms the existing techniques in terms of blocking probability (BP). The BP for this technique is significantly lower compared with the existing ones, for all multicast group sizes. More importantly, there is no tradeoff, since the average number of arcs that

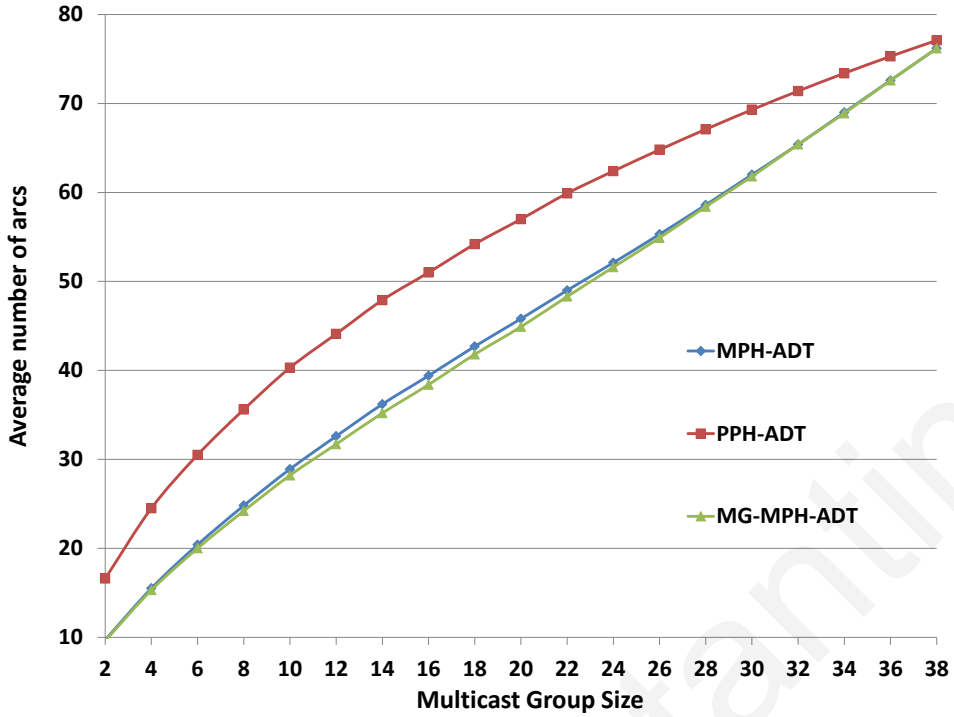


Figure 7.4: Average number of arcs vs. multicast group size for the three different multicast protection techniques (MPH-ADT, PPH-ADT, and MG-MPH-ADT).

are used is slightly less compared with the best of the existing methods (MPH-ADT). Therefore, in the case of mixed-graph networks, the application of the proposed protection method has only advantages.

7.2.2 MG-SNH-ADT Protection

The Mixed-Graph Steiner Node Heuristic (MG-SNH) described in Chapter 5 can also be utilized for the calculation of a pair of arc-disjoint trees, and the resulting technique is called *MG-SNH-ADT* [73]. This technique was designed to perform well for both undirected and mixed graphs, due to the fact that it exploits the benefits of both the SNH and MG-MPH multicast routing heuristic algorithms.

The MG-SNH-ADT multicast protection method consists of the following steps:

1. Calculate the primary light-tree using the MG-SNH heuristic as described in Chapter 5.

2. Remove the arcs of the primary tree from the network graph.

3. Calculate the secondary tree on the resulting graph using the MG-SNH algorithm.

7.2.2.1 Performance Evaluation

The performance of the MG-SNH-ADT protection method is evaluated and compared with the existing relevant heuristic algorithms (MPH-ADT) and the ones proposed previously (SNH-ADT, MG-MPH-ADT). PPH-ADT is omitted in this analysis since it performs much worse compared to the other three techniques [49, 74, 75]. The evaluation is performed via simulations on a mixed graph that was randomly created. The graph was created to be strongly connected. It has 40 nodes, 160 arcs (unidirectional connections), and 40 links (bidirectional connections). This graph should be simple, therefore all 200 edges connect different pairs of nodes. The *Percentage of Directionality (PoD)* (i.e., the ratio *arcs*/*edges*) was set to $PoD = 160/200 = 0,8 = 80\%$. The simulation was repeated for various possible multicast group sizes (D), from $D = 2$ to $D = 39$. The experiment was executed 50000 times for every multicast group size, while the source and destinations of the multicast connections were distributed uniformly across the network. Fifty thousand (50000) pairs of arc-disjoint trees were calculated with each one of the existing and proposed multicast protection methods. If an arc-disjoint pair of trees was able to be calculated, the multicast request was considered as realizable. In the opposite case it was considered as blocked. The blocking probability was calculated as the ratio of *blocked*/*total* multicast requests. The cost criterion for the routing heuristic algorithms was the sum of the arcs that were used for the primary and secondary trees. The blocking probability and the average cost (i.e., the sum of the arcs of the two disjoint trees) were calculated for each protection method. The results are presented in Figures 7.5-7.8. The cases of small (up to 20 destinations) and large (21 destinations and more) multicast group sizes were presented in different graphs for clarity purposes.

Analysis of the results

Small multicast group size (size of destination set up to half the size of the network): It is clear that the MG-SNH-ADT multicast protection method outperforms the rest in terms of blocking probability (BP) and average cost, for small multicast sessions. The first reason is that this method is able to find the nodes that give light-trees with less arcs if they are added in the destination set. The second reason is that the proposed method was specifically designed for mixed graphs, therefore it has enhanced performance for this category of networks. Since MG-SNH-ADT finds trees with less arcs, it is obvious that this minimizes the average cost (average cost

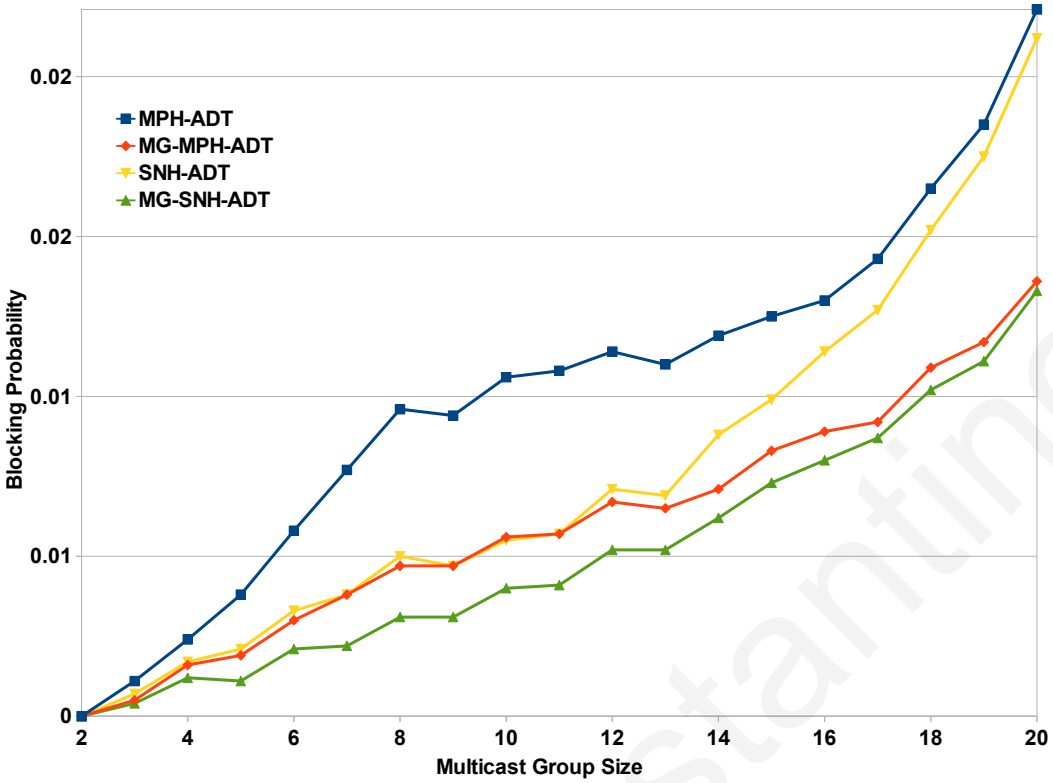


Figure 7.5: Blocking probability vs. multicast group size for the four different multicast protection techniques (MPH-ADT, SNH-ADT, MG-MPH-ADT, MG-SNH-ADT) for multicast group size up to 20 destinations.

is calculated in terms of number of arcs of the primary and secondary trees). It minimizes the blocking probability as well; if the primary tree uses a small number of arcs, the resulting graph after the removal of them consists of more arcs and this decreases the possibility that a secondary tree will not exist. Therefore, the BP is decreased as well.

Large multicast group size: For the case of large multicast sessions, MPH-ADT and SNH-ADT have the same blocking probability. The blocking probability of MG-MPH-ADT and MG-SNH-ADT is equal, much lower compared to the other two techniques. The reason is that if the destination set has many network nodes, the possibility of the existence of non-destination nodes that will give trees with less arcs if added in the destination set, is negligible. Therefore, the only characteristic that can give better results is the mixed-graph specialization of an algorithm. The four methods also have practically the same average cost for large multicast sessions.

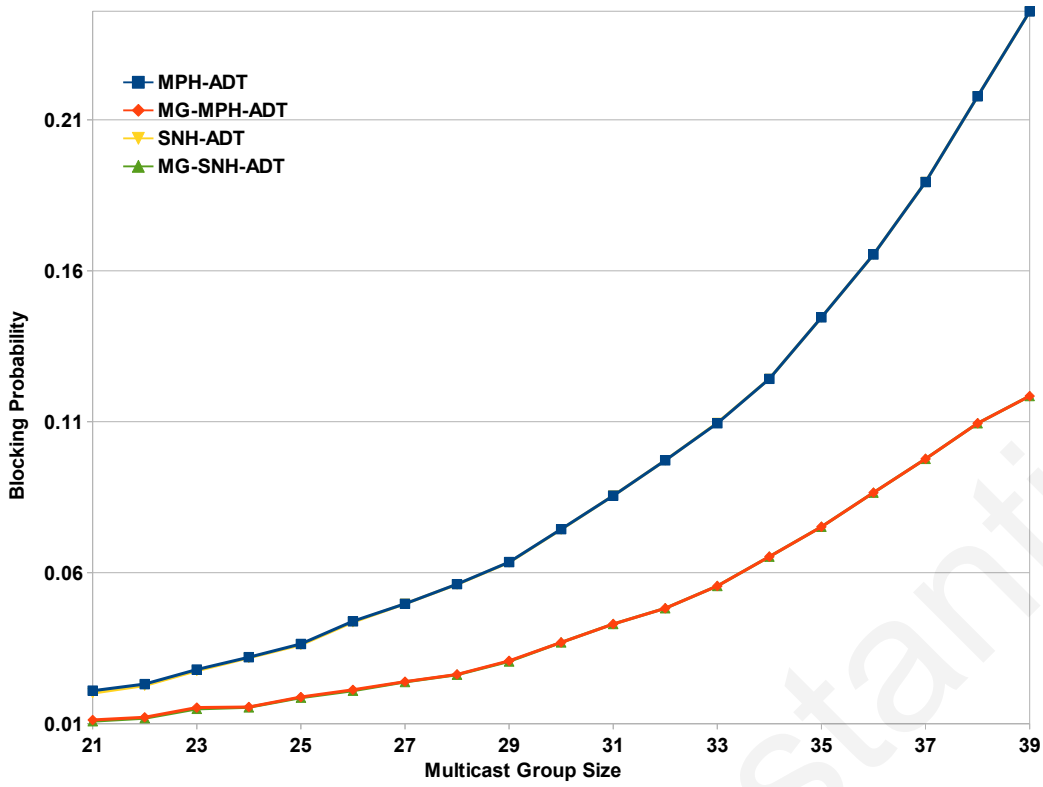


Figure 7.6: Blocking probability vs. multicast group size for the four different multicast protection techniques (MPH-ADT, SNH-ADT, MG-MPH-ADT, MG-SNH-ADT) for multicast group size ≥ 21 destinations.

7.2.3 Conclusions

An arc-disjoint multicast protection method called *MG-MPH-ADT*, that is based on the mixed-graph routing heuristic algorithm *MG-MPH* presented in Chapter 5, is presented and evaluated in the current work. Simulations show that the new method outperforms the ones mainly used in the literature (arc-disjoint protection methods *MPH-ADT* and *PPH-ADT*) for mixed-graph networks in terms of blocking probability and average cost.

Furthermore, another multicast routing heuristic algorithm that performs efficiently for mixed-graph networks, called *Mixed Graph Steiner Node Heuristic (MG-SNH)*, that was also introduced in Chapter 5 is applied for the provisioning of survivable multicast requests via the calculation of arc-disjoint tree pairs. The resulting technique is *MG-SNH-ADT*. It was specifically designed to work efficiently for mixed graphs and simulation results verify this, showing that for this category of networks, MG-SNH-ADT outperforms the existing relevant methods (*MPH-ADT*, *SNH-ADT*, and *MG-MPH-ADT*) for small multicast sessions (size of the destination

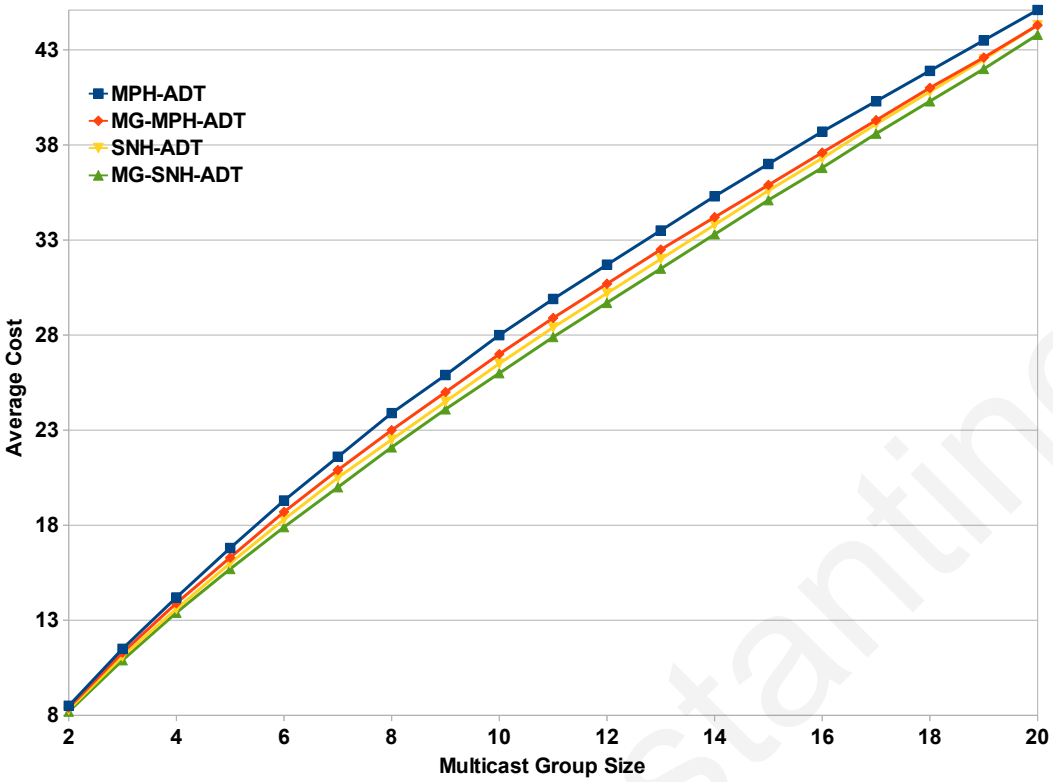


Figure 7.7: Average number of arcs vs. multicast group size for the four different multicast protection techniques (MPH-ADT, SNH-ADT, MG-MPH-ADT, MG-SNH-ADT) for multicast group size up to 20 destinations.

set up to half the number of the network nodes), in terms of blocking probability while having the same average cost.

7.3 Generalization of OPP-SDP for Multicast Protection in Mixed-Graph Networks

As explained in Chapter 6, the advantage of the OPP-SDP multicast protection method is that it can manage cases where the trap topology of Figure 6.8 exists. Its drawback is that it has poor performance for mixed-graph networks. An example to demonstrate this is given in Figure 7.9 (s is the source and a, b are the destinations). In this example, the OPP-SDP heuristic will give the solution of Step 2, where the optimal is the one given at Step 4.

The weakness of OPP-SDP in mixed-graph networks is that it cannot manage the case where, after the addition of a destination into the path-pairs-tree (i.e., the tree derived by OPP-SDP, where a pair of paths between the source and each destination

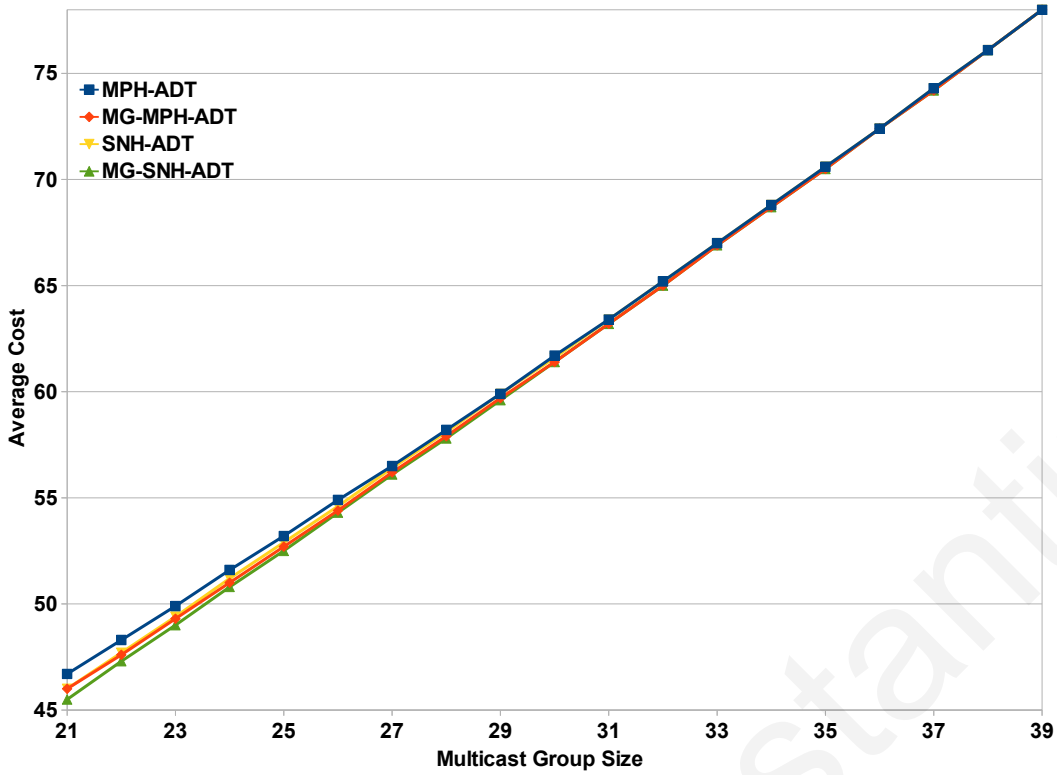


Figure 7.8: Average number of arcs vs. multicast group size for the four different multicast protection techniques (MPH-ADT, SNH-ADT, MG-MPH-ADT, MG-SNH-ADT) for multicast group size ≥ 21 destinations.

exists), the already added destinations may can be *reconnected* to the tree through a *shorter* way. Therefore, a new multicast protection algorithm is presented here, called *Mixed Graph Optimal Path Pair-Shared Disjoint Paths* (MG-OPP-SDP), consisting of the following steps:

Mixed Graph Optimal Path-Pair Shared Disjoint Paths (MG-OPP-SDP):

1. Tree T_1 consists only of the source s. Set $V_1 = \{\emptyset\}$. $i = 1$.

2. (a) Determine the node u in $\{D - V_1\}$, that has the shortest path-pair that originates from the source node s and ends at node u. Construct the tree T_{i+1} by adding this path-pair to T_i. Add u in set $\{V_1\}$.

   (b) Keep only the path-pair from source to node u as the current tree and delete all other path-pairs.

   (c) Update $cost = 0$ for this path-pair.

   (d) Update $cost = initial - graph - cost$, for the removed path-pairs.

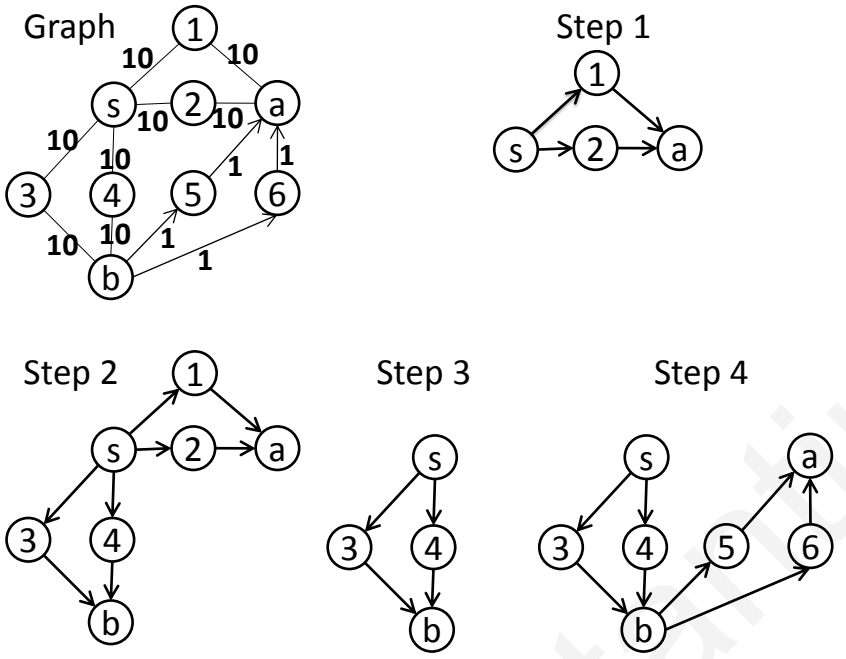


Figure 7.9: Example where OPP-SDP does not give the optimal solution.

(e) Add again to the current tree all the destinations that were part of the tree before u, by calculating their path-pairs using Suurballe's algorithm.

(f) Update $cost = 0$ for their path-pairs.

(g) $i = i + 1$.

3. Repeat Step 2. STOP when all nodes in D are connected to the tree.

Note that the path-pairs are derived using Suurballe's algorithm and $Initial - graph - cost$ of a path-pair, is the cost that the path-pair had before it was set to 0. Also, for OPP-SDP, the destinations are added in the tree sequentially, while in this case the one with the shortest path-pair is added first.

The heuristic algorithm functions as follows: After the addition of destination u (let U be the set of the nodes that are added to the tree for the connection of destination u), only the path-pair from source s to u is kept as the current tree. All destinations that were part of the tree before the addition of u and do not belong to path $s \rightarrow u$, are added again using Suurballe's algorithm. Therefore, they will be connected either through the path-pair that they were already connected to, or through a path-pair that passes through the nodes in set U and uses the arcs that their cost has been updated to 0, since they are used for destination u as well. The

best path between the two will be selected. It is obvious that OPP-SDP and MG-OPP-SDP have the same performance for undirected graphs: using MG-OPP-SDP, the already added destination path-pairs will be removed and the same ones will be added again. For mixed graphs, the proposed algorithm has improved performance, as Figure 7.9 shows: OPP-SDP stops at Step 2, while MG-OPP-SDP stops at Step 4 and gives the optimal solution.

Complexity of the MG-OPP-SDP Heuristic

- Suppose that the destination set consists of D nodes.
- Suurballe's algorithm has time complexity of $O(n^2 \log n)$ for a graph consisting of V nodes [27].
- Step 2 of MG-OPP-SDP is repeated for every destination (i.e., D times), and Suurballe's algorithm is applied at most D times for each iteration of Step 2.

The result of the aforementioned statements is that MG-OPP-SDP has time complexity of $O(D^2 n^2 \log n)$.

7.3.1 Performance Evaluation

The performance of the MG-OPP-SDP multicast protection technique was evaluated and compared to OPP-SDP. The simulations were performed on a mixed graph that was randomly created. The graph was created to be strongly connected. It had 30 nodes, 50 *arcs* (unidirectional connections) and 50 *links* (bidirectional connections), for a total of 100 *edges* (sum of arcs and links) and a *Percentage of Directionality (PoD)* (i.e., the ratio *arcs*/*edges*) equal to $PoD = 50/100 = 0,5 = 50\%$. This graph should be simple, therefore all 100 edges connected different pairs of nodes. The simulation was repeated for various possible multicast group sizes (D), from $D = 1$ to $D = 29$ (D stands for the number of destinations of the multicast group). The simulation was executed 25000 times for every multicast group size, while the source and destinations of the multicast connections were distributed uniformly across the network. The simulation was performed for dynamic arrival of multicast calls: they were considered to arrive, stay for some time, and then depart. The time of arrival was generated according to poisson distribution with $\lambda_{poisson} = 100$ and the connection holding time was exponentially distributed with $\lambda_{exp} = 1$. Therefore, the network load was set to 100 *Erlangs*. Every fiber was considered to have 64

wavelengths. If a pair of paths that originated from the source and ended at a destination was able to be calculated for every destination of the multicast call, the call was considered to be successful. Otherwise, it was considered as blocked. The blocking probability was calculated as the ratio of *blocked*/*total* multicast requests, for every multicast group size. The results are presented in Figures 7.10 and 7.11. The cases of small (up to 15 destinations) and large (16 destinations and more) multicast group sizes were presented in different graphs for clarity purposes.

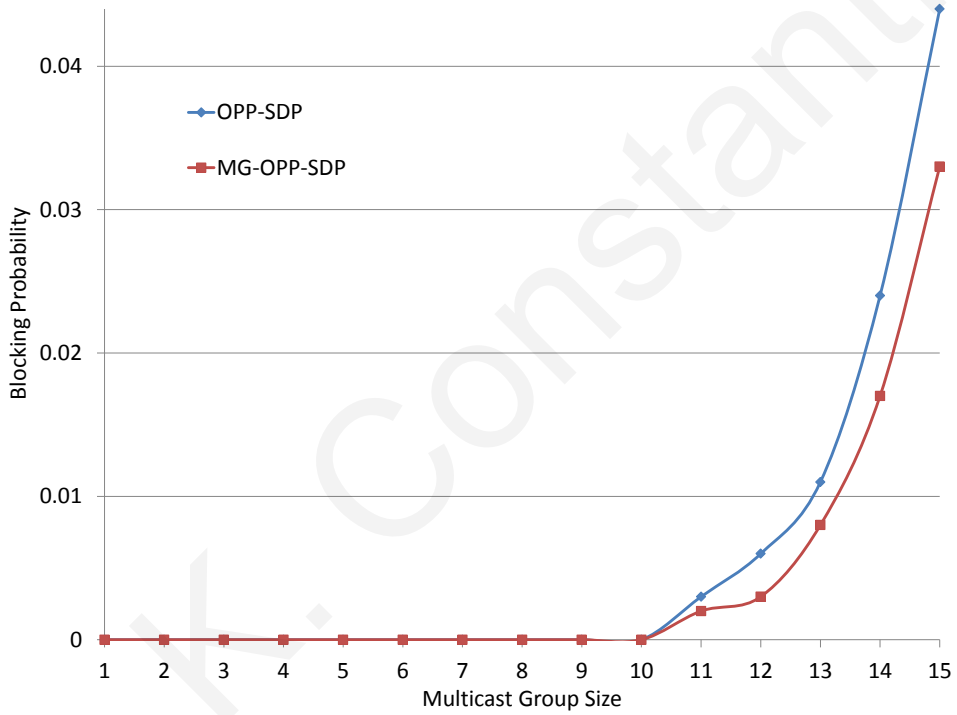


Figure 7.10: Blocking probability vs. multicast group size for OPP-SDP and MG-OPP-SDP, for multicast group size ≤ 15 destinations.

Analysis of the results

For the results, it is demonstrated that the MG-OPP-SDP technique has lower blocking probability compared to OPP-SDP. For up to 10 destinations, the blocking probability is zero for both heuristic algorithms. The difference of blocking probability between the two heuristics was averaged for multicast groups from 11 to 29 destinations. The result was that MG-OPP-SDP algorithm gives on average 16.6% lower blocking probability compared to OPP-SDP.

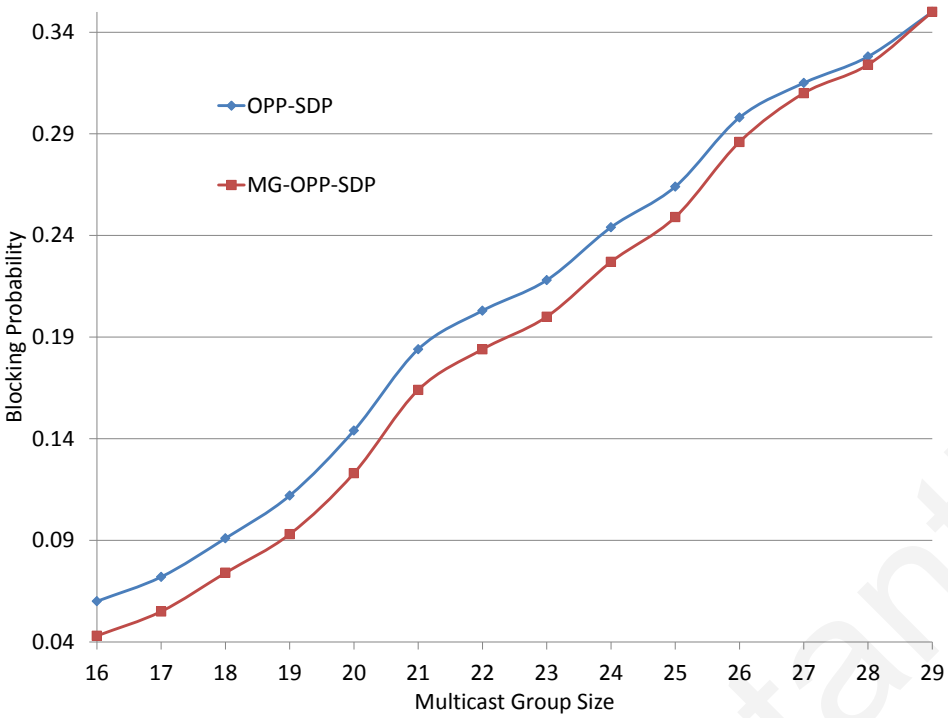


Figure 7.11: Blocking probability vs. multicast group size for OPP-SDP and MG-OPP-SDP, for multicast group size ≥ 16 destinations.

7.3.2 Conclusions

The existence of protection techniques that are efficient in mixed graphs is important, since in practice some connections in the network are unidirectional due to the resource holding of other applications. A new heuristic algorithm, called *Mixed Graph Optimal Path-Pair Shared Disjoint Paths* (MG-OPP-SDP), that is a generalization of the OPP-SDP algorithm, is presented in this section for protection of multicast connections in mixed-graph networks. While for undirected-graph networks OPP-SDP and MG-OPP-SDP have the same performance, for mixed-graph networks MG-OPP-SDP clearly outperforms OPP-SDP. Specifically, simulation results showed that the performance of MG-OPP-SDP is on the average 16.6% better compared to that of OPP-SDP.

Chapter 8

Conclusions and Future Research

8.1 Conclusions

This dissertation dealt with the problems of multicast routing and protection in optical WDM networks, a current problem in mesh optical networks where a number of multicast applications are being established in the network. Several heuristic algorithms that outperform the existing ones were presented, explained, and evaluated through examples and simulations for undirected networks and for mixed-graph networks.

The thesis achieved all its objectives as highlighted in Chapter 1. New heuristic algorithms and techniques were developed that enable the provisioning and protection of multicast requests in optical mesh networks with varying architectures (e.g., undirected networks, mixed-graph networks, networks with full or sparse-splitting capabilities, networks with DoC or DaC MI nodes, etc) while at the same time outperforming the most-utilized existing routing and protection techniques that are present up-to-date in the literature.

A new multicast routing heuristic algorithm, called *Steiner Node Heuristic* (SNH) was presented in this dissertation. This heuristic uses MPH as its basis, and applies a recursive procedure to find a tree that has less cost compared to the one obtained by MPH. It achieves this by locating the nodes that, if added into the destination set, will give a less-cost tree. It is a general algorithm, i.e., it can use any appropriate algorithm X as its basis and follow the recursive procedure to improve the solution obtained by X. It is noted that algorithms that use this method (i.e., to locate and add certain nodes in the destination set to improve the solution) did not exist in the

literature for multicast routing applications.

For sparse-splitting networks (i.e., networks where not all nodes are multicast capable) both problems of multicast routing in these types of networks and splitter allocation were investigated. For the first problem, novel heuristic algorithms were created, that outperform the best ones that exist in the literature. The proposed *Sparse-Splitting Multicast Routing Heuristic* (SSMRH) outperforms the existing ones, since it locates the multicast-capable nodes, that, if used for the calculation of the multicast tree, will reduce its cost. For the second problem, new splitter allocation heuristic algorithms were presented, that outperform the existing ones. These are the *Decreased Number of Branches* (DNB), *Decreased Number of Children Destinations*, and *Least Used Removed First* (LURF) approaches. All proposed techniques were analyzed and evaluated for networks where the multicast-incapable nodes were *Drop-and-Continue* and for networks where the multicast-incapable nodes are *Drop-or-Continue*.

Another major contribution of the current thesis was the research performed on mixed-graph networks. Previous research work on routing in optical networks was performed according to the assumption that the network can be modelled as an undirected graph (a graph consisting only of bidirectional connections), and routing was performed using undirected-graph multicast routing heuristic algorithms. However, this is not a practical scenario, and in the thesis the reasons of modelling the network as a mixed graph (a graph consisting of both bidirectional and unidirectional connections) were given. The performance of the undirected-graph multicast routing heuristic algorithms if applied in mixed graphs was mathematically analyzed and evaluated through simulations. Consequently, new multicast routing heuristic algorithms that are specially designed for this category of networks, were presented and evaluated. A modification of the *Chu Liu* algorithm for multicasting applications, was presented as well. The proposed solutions that were specifically designed for mixed-graph networks were shown to outperform solutions that were initially designed for undirected networks.

The proposed undirected- and mixed-graph network multicast routing heuristic algorithms were further exploited for the creation of new protection techniques for *single link failure* and *single link/node failure* scenarios. The created techniques were *SNH-ADT, SNH-NDT, MG-MPH-ADT*, and *MG-SNH-ADT* and these solutions were shown to outperform the existing ones in terms of blocking probability and average

cost.

Another important contribution of the thesis was the development of a load balancing technique that decreases the blocking probability of multicast connections. Currently, for the establishment of upcoming multicast requests, the network state due to the already deployed connections, was ignored by all existing heuristics. The developed method modifies the cost of each network arc according to the distribution of the load in the network, in order to achieve a balanced load distribution and to minimize the possibility that the network will be disconnected in some areas due to load congestion on certain arcs. This technique was used in conjunction with the OPP-SDP protection method, that is one of the best protection methods in the literature, and demonstrated significant improvement in the network performance in terms of blocking probability (50% lower average blocking probability for small multicast group sizes).

Additionally, a novel heuristic algorithm was developed, that provides protection for multicasting in mixed-graph networks, and is able to deal with certain "trap" topologies where other heuristic algorithms fail. It is a generalization of *Optimal Path Pair Shared Disjoint Paths* (OPP-SDP), called *Mixed Graph Optimal Path Pair Shared Disjoint Paths* (MG-OPP-SDP), which was shown to outperform OPP-SDP in terms of blocking probability for mixed-graph networks.

For the evaluation of the proposed heuristics and the comparison with the existing ones, simulations for both static and dynamic arrivals of multicast requests were performed. Furthermore, the simulations were performed for several multicast group sizes, on existing networks, randomly created graphs, as well as complex graph topologies (small-world and scale-free), with varying characteristics in terms of connectivity, directionality, etc.

The proposed heuristics have increased performance compared to the existing ones, while their time complexity remains polynomial. However, as the proposed solutions increase the time complexity, it is up to the user to decide when it is appropriate to use them. If the computational power is limited and the capacity is not, the existing heuristic algorithms are more appropriate. However, in today's networks, computational power is enough to deal with the increased complexity of the proposed heuristics. On the other hand, capacity is limited. Therefore, the new heuristic algorithms are the most suitable solution for most of the applications.

The proposed heuristics of the current thesis were developed in order to deal with

multicast routing and protection in mesh optical networks. It must be stated though, that most of them are general, i.e., they can be used in other types of networks as well, with some minor modifications.

8.2 Future Research

The work presented in this dissertation did not cover a number of topics that could be exploited as future work by other researchers working in this area. Some examples for future directions are given below. There are a number of possible avenues that one can explore as a continuation of the work presented in this thesis. This section serves only as a guide towards some interesting future directions that could be investigated but is by no means an exhaustive list.

One possible avenue for further exploration is to enhance the multicast routing heuristic algorithms developed in this dissertation:

- The problem of multicast traffic grooming in optical networks can be a subject of future research. In the current thesis it was assumed that each multicast request used the entire wavelength capacity. However, in order to deal with scenarios were the multicast request does not require the entire wavelength capacity, multicast traffic grooming techniques must be developed where a wavelength is concurrently supporting multiple requests.

- The case of multicast routing and MC-node allocation in mixed-graph sparse-splitting networks can also be examined for networks where the multicast-incapable nodes are either DaCs or DoCs. Furthermore, scenarios where the multicast-capable nodes have full or limited splitting capability can be a subject of future research (limited splitting capability refers to the case where an optical splitter has a limited splitting fanout (splitting capacity)).

- Multicast routing heuristic algorithms for optical networks with sparse (or limited) wavelength-conversion capability can be created, as well as heuristics for efficient placement of nodes that have this capability. An important issue related to this is the efficient allocation of the wavelength converters. Another open problem is the development of efficient multicast routing and protection heuristic algorithms, for the case of sparse wavelength-conversion networks.

Networks with limited wavelength conversion can be studied as well (limited conversion refers to the case where there is wavelength conversion capability in the network, but it may be restricted so that not all combinations of input/output wavelength channels may be possible).

- The new multicast routing heuristic algorithms developed in this dissertation ignored the physical layer impairments, such as fiber attenuation, component insertion loss, amplifier spontaneous emission noise, chromatic dispersion, polarization mode dispersion, and others. The proposed routing and protection algorithms can thus be generalized considering the degradation of the signal quality due to the aforementioned factors.

There are a number of future research directions that can also be undertaken for the case of multicast protection:

- One possible direction is to further improve on the OPP-SDP heuristic that is currently the most efficient one found in the literature for providing protection in undirected-graph networks. An improved heuristic can be created (e.g., "Improved OPP-SDP" (I-OPP-SDP)), that will be able to locate the nodes that, if added in the destination set, will give a less-cost tree. The new heuristic can subsequently be combined with MG-OPP-SDP, to provide a more efficient solution for mixed-graph networks. The heuristic algorithms developed can also be combined with the developed load balancing technique for provisioning dynamically arriving multicast calls with decreased blocking probability.

- The proposed protection methods can also be combined with *cross-sharing* techniques. If cross-sharing is exploited, two multicast requests can share the arcs used for protection, as long as their working trees are arc-disjoint. This will lead to slightly longer restoration times, but the redundant capacity required in the network will be decreased.

- The problem of multicast routing in sparse-splitting networks was examined in the current thesis. No algorithms that provide multicast protection for this type of networks were developed though. This can be a subject of future research. The developed heuristic algorithms can be combined with the arc-disjoint-trees protection method, in order to provide survivable multicasting

for this category of networks. Apart from these, heuristic algorithms that provide efficient multicast protection in sparse-splitting networks by being able to overcome certain network trap topologies, can be developed.

- The load balancing technique presented in the thesis took into account the possible network flow through each fiber $a - b$, according to the free wavelengths of the arcs ending at a and originating from b. The free wavelengths of $a - b$ were ignored though. A new load balancing technique can also be created and evaluated, that will take this factor into account as well.

Apart from new directions in heuristics, future work can also focus on the comparison of all proposed routing and protection heuristic algorithms with the optimal solution obtained by Integer Linear Programming to ascertain how close to the optimal solution are the proposed techniques.

Furthermore, the SNH heuristic was compared to the MPH and PPH heuristics, since these are the ones that are most widely used in optical networks. Another heuristic, written by Kou [37], was also compared to SNH. However, other heuristics, that have better worst-case performance, were not used for comparison purposes. SNH can also be compared to them as well. This comparison can be performed both theoretically (through mathematical analysis) and experimentally (via simulations).

Finally, the generalization of the proposed heuristic algorithms, so that they can be applied in a distributed rather than a centralized control and management environment, as well as the development of multicast protocols that exploit the proposed multicast routing and protection heuristic algorithms can be another direction of future work.

Obviously, the aforementioned are just some of the interesting future directions that could be investigated. The main focus of any future research direction would be to develop efficient techniques and heuristics that can allow for the provisioning and survivability of multicast applications in telecommunication networks, trying to bring the proposed solutions closer to real-network implementations.

Bibliography

[1] http://www.internetworldstats.com/stats.htm

[2] R. Ramaswami, "Multiwavelength Lightwave Networks for Computer Communication", *IEEE Communications Magazine*, 31(2):78–88, 1993.

[3] R. Ramaswami and K.N. Sivarajan, *Optical Networks: A Practical Perspective*, 2nd edition, Morgan Kaufmann Publishers, 2002.

[4] C. Siva Ram Murthy and M. Gurusamy, *WDM Optical Networks, Concepts, Design, and Algorithms*, Prentice Hall, 2002.

[5] L. Sahasrabuddhe and B. Mukherjee, "Light-Trees: Optical Multicasting for Improved Performance in Wavelength-Routed Networks", *IEEE Communications Magazine*, 37(2):67-73, 1999.

[6] B. Ramamurthy and B. Mukherjee, "Wavelength Conversion in WDM Networking", *IEEE Journal on Selected Areas in Communications*, 6(7):1061-1073, 1998.

[7] G.N. Rouskas, "Optical Layer Multicast: Rationale, Building Blocks, and Challenges", *IEEE Network*, 17(1):60-65, 2003.

[8] N.K. Singhal and B. Mukherjee, "Architectures and Algorithm for Multicasting in WDM Optical Mesh Networks Using Opaque and Transparent Optical Cross-Connects", *IEEE/OSA Optical Fiber Communication Conference (OFC)*, Anaheim, CA, March, 2001.

[9] W.S. Hu and Q.J. Zeng, "Multicasting Optical Cross Connects Employing Splitter-and-Delivery Switch", *IEEE Photonics Technology Letters*, 10(7):970-972, 1998.

[10] M. Ali and J. Deogun, "Power-efficient Design of Multicast Wavelength-Routed Networks", *IEEE Journal on Selected Areas in Communications*, 18(10):1852-1862, 2000.

[11] F. Farahmand, X. Huang, and J.P. Jue, "Efficient Online Traffic Grooming Algorithms in WDM Mesh Networks with Drop-and-continue Node Architecture", *Proc. International Conference on Broadband Networks (BroadNets)*, San Jose, CA, October 2004.

[12] R.M. Karp, "Reducibility Among Combinatorial Problems. Complexity of Computer Computations", Chapter 8 in *50 Years of Integer Programming 1958–2008*, R.E. Miller and J.W. Thatcher, eds., Plenum Press, New York, 1972.

[13] K.C. Lee and V.O.K. Li, "A Wavelength-Convertible Optical Network", *IEEE/OSA Journal of Lightwave Technology*, 11(5):962-970, 1993.

[14] N.K. Singhal and B. Mukherjee, "Protecting Multicast Sessions in WDM Optical Mesh Networks", *IEEE/OSA Journal of Lightwave Technology*, 21(4):884-892, 2003.

[15] H. Takahashi and A. Matsuyama, "An Approximate Solution for the Steiner Problem in Graphs", *Math. Japonica*, 24(6):573-577, 1980.

[16] R.W. Floyd, "Algorithm 97: Shortest Path", *Communications of the ACM*, 5(6):345, 1962.

[17] S. Warshall, "A Theorem on Boolean Matrices", *Journal of the ACM*, 9(1):11-12, 1962.

[18] R.C. Prim, "Shortest Connection Networks and some Generalizations", *Bell System Technical Journal*, 36:1389–1401, 1957.

[19] X. Zhang, J.Y. Wei, and C. Qiao, "Constrained Multicast Routing in WDM Networks with Sparse Light Splitting", *IEEE/OSA Journal of Lightwave Technology*, 18(12):1917-1927, 2000.

[20] C. Hsieh and W. Liao, "All-Optical Multicast Routing in Sparse Splitting WDM Networks", *IEEE Journal on Selected Areas in Communications*, 25(6):51-62, 2007.

[21] S. Cho and T. Lee, "Minimum Cost Multicast Routing Based on High Utilization MC Nodes Suited to Sparse-Splitting Optical Networks", *Proc. International Conference on Computational Science and its Applications (ICCSA)*, Glasgow, UK, May 2006.

[22] A. Fei et al., "A Dual-Tree Scheme for Fault-Tolerant Multicast", *Proc. IEEE International Conference on Communications (ICC)*, Helsinki, Finland, June 2001.

[23] N.K. Singhal, L.H. Sahasrabuddhe, and B. Mukherjee, "Provisioning of Survivable Multicast Sessions Against Single Link Failures in Optical WDM Mesh Networks", *IEEE/OSA Journal of Lightwave Technology*, 21(11):2587-2594, 2003.

[24] C. Lu et. al., "A Novel Shared Segment Protection Algorithm for Multicast Sessions in Mesh WDM Networks," *ETRI Journal*, 28(3):329–336, 2006.

[25] L. Liao, L. Li, and S. Wang, "Dynamic Segment Shared Protection for Multicast Traffic in Meshed Wavelength-Division-Multiplexing Optical Networks," *OSA Journal of Optical Networking*, 5(12):1084–1092, 2006.

[26] A. Khalil, T. Rahman, A. Hadjiantonis, G. Ellinas, and M. Ali, "Pre-planned Multicast Protection Approaches in WDM Mesh Networks", *Proc. European Conference on Optical Communications (ECOC*), Glasgow, Scotland, October 2005.

[27] J.W. Suurballe, "Disjoint Paths in a Network", *Networks*, 4:125-145, 1974.

[28] T. Feng, L. Ruan and W. Zhang, "Intelligent p-Cycle Protection for Dynamic Multicast Sessions in WDM Networks", *IEEE/OSA Journal of Optical Communications and Networking*, 2(7):389–399, 2010.

[29] W.D. Grover and D. Stamatelakis, "Cycle-Oriented Distributed Preconfiguration: Ring-Like Speed with Mesh-Like Capacity for Self-Planning Network Restoration", *Proc. IEEE International Conference on Communications (ICC)*, Atlanta, GA, June 1998.

[30] F.K. Hwang, D.S. Richards, and P. Winter, *The Steiner Tree Problem*, Elsevier Science Publishers B.V., 1992.

[31] E.W. Dijkstra, "A Note on Two Problems in Connexion with Graphs", *Numerische Mathematik*, 1:269-271, 1959.

[32] R. Bellman, "On a Routing Problem", *Quarterly of Applied Mathematics*, 16:87-90, 1958.

[33] L.R. Ford and D.R. Fulkerson, *Flows in Networks*, Princeton University Press, 1962.

[34] J.B. Kruskal, "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem", *Proceedings of the American Mathematical Society*, 7(1):48-50, 1956.

[35] J. Nesetril, E. Milkova, and H. Nesetrilova, "Otakar Boruvka on Minimum Spanning Tree Problem (Translation of Both the 1926 Papers, Comments, History)", *Discrete Mathematics* 233(1-3):3-36, 2001.

[36] G. Robins and A. Zelikovski, "Tighter Bounds for Graph Steiner Tree Approximation", *ACM Journal of Discrete Mathematics*, 19(1):122-134, 2005.

[37] L. Kou, G. Markowsky, and L. Berman, "A Fast Algorithm for Steiner Trees", *Acta Informatica*, 15:141-145, 1981.

[38] A. Zelikovsky, "An 11/6-approximation Algorithm for the Network Steiner Problem", *Algorithmica*, 9:351-354, 1993.

[39] M. Karpinski and A. Zelikovsky, "New Approximation Algorithms for the Steiner Tree Problems", *Journal of Combinatorial Optimization*, 1:47-65, 1997.

[40] H.J. Promel, "A New Approximation Algorithm for the Steiner Tree Problem with Performance Ratio 5/3", *Journal of Algorithms*, 36:89-101, 2000.

[41] S. Hougardy and H.J. Prijmel, "A 1.598 Approximation Algorithm for the Steiner Problem in Graphs", *Proc. ACM-SIAM Symposium on Discrete Algorithms*, Baltimore, MD, January 1999.

[42] H.J. Promel and A. Steger, "RNC-approximation Algorithms for the Steiner Problems", *Proc. 14th Annual Symposium on Theoretical Aspects of Computer Science*, Lubeck, Germany, February/March 1997.

[43] A. Zelikovsky, "Better Approximation Bounds for the Network and Euclidean Steiner Tree Problems", *Technical Report CS-96-06*, University of Virginia, 1996.

[44] P. Berman and V. Ramaiyer, "Improved Approximations for the Steiner Tree Problem", *Journal of Algorithms*, 17(3):381-408, 1994.

[45] T.H. Corman, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, 2nd ed. Cambridge, MA: MIT Press, 2001.

[46] S. Yan, J. Deogun, and M. Ali, "Routing in Sparse Splitting Optical Networks with Multicast Traffic", *Computer Networks*, 41(1):89-113, 2003.

[47] P. Soproni and T. Cinkler, "Physical Impairment Aware Multicast Routing Heuristics", *Proc. 13th International Conference on Transparent Optical Networks (ICTON)*, Stockholm, Sweden, June 2011.

[48] L. Guo, et al., "Heuristic Protection Algorithm in Multicast Multi-domain Optical Networks", *Proc. IEEE International Conference on Communication Software and Networks (ICCSN)*, Xian, China, May 2011.

[49] C.K. Constantinou and G. Ellinas, "A Novel Multicast Routing Algorithm and its Application for Protection against Single-link and Single-link/node Failure Scenarios in Optical WDM Mesh Networks", *OSA Optics Express*, 19(26):B471-B477, 2011.

[50] F. Zhou, M. Molnar, and B. Cousin, "Is Light-Tree Structure Optimal for Multicast Routing in Sparse Light Splitting WDM Networks?", *Proc. Internatonal Conference on Computer Communications and Networks (ICCCN)*, San Francisco, CA, August 2009.

[51] S. Yan, M. Ali, and J. Deogun, "Route Optimization of Multicast Sessions in Sparse Light-splitting Optical Networks", *Proc. IEEE Global Telecommunications Conference (GLOBECOM)*, San Antonio, TX, November 2001.

[52] M. Ali and J. Deogun, "Allocation of Splitting Nodes in All-Optical Wavelength-Routed Networks", *Photonic Network Communications*, 2(3):257-265, 2000.

[53] W. Tseng and S. Kuo, "All-Optical Multicasting on Wavelength-Routed WDM Networks with Partial Replication", *Proc. International Conference on Information Networking (ICOIN)*, Beppu City, Japan, January-February 2001.

[54] X. Wang, S. Wang, and L. Li, "A Novel Efficient Multicast Routing Algorithm in Sparse Splitting Optical Networks", *Photonic Network Communications*, 14(3):287-295, 2007.

[55] J. Wang, X. Yu, J. Yuan, and Z. Wu, "An Improved Multicast Routing Algorithm in Sparse Splitting Optical Networks", *Proc. Asia Communications and Photonics Conference and Exhibition (ACP)*, Shanghai, China, November 2009.

[56] M. Ali and J.S. Deogun, "Cost-Effective Implementation of Multicasting in Wavelength-Routed Networks", *IEEE/OSA Journal of Lightwave Technology*, 18(12):1628-1638, 2000.

[57] S.W. Wang, "Allocation of Light Splitters in All-Optical WDM Networks with Sparse Light Splitting Capabilities", *Springer Telecommunication Systems*, DOI 10.1007/s11235-011-9654-6, pp.1-10, 2011.

[58] M. Ali, "Optimization of Splitting Node Placement in Wavelength-Routed Optical Networks", *IEEE Journal on Selected Areas in Communications*, 20(8):1571-1579, 2002.

[59] A. Billah, B. Wang, and A. Awwal, "Topology Based Placement of Multicast Capable Nodes for Supporting Efficient Multicast Communication in WDM Optical Networks", *Photonic Network Communications*, 14(1):35-47, 2007.

[60] K. Yong, T. Cheng, G. Xiao, and L. Zhou, "Placement of Multicast Capable Nodes in Power Constrained All-Optical WDM Networks", *Proc. IEEE Global Telecommunications Conference (GLOBECOM)*, Miami, Florida, December 2010.

[61] O. Yu and Y. Cao, "Placement of Light Splitters and Wavelength Converters for Efficient Multicast in All-Optical WDM Networks", *IEICE Transactions on Information and Systems*, *E89-D*(2):709-718, 2006.

[62] M. Behzad and G. Chartrand, *Introduction to the Theory of Graphs*, Allyn and Bacon Inc., Boston, 1971.

[63] K. Thulasiraman and M.N.S. Swamy, *Graphs: Theory and Algorithms*, Wiley-Interscience, 1992.

[64] M. Charikar, et al., "Approximation Algorithms for Directed Steiner Problems", *Journal of Algorithms*, 33:73-91, 1999.

[65] L. Zosin and S. Khuller, "On Directed Steiner trees", *Proc. Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, CA, January 2002.

[66] Y.J. Chu and T.H. Liu, "On the Shortest Arborescence of a Directed Graph", *Scientia Sinica*, 14(10):1396-1400, 1965.

[67] J. Edmonds, "Optimum Branchings", *J. Research of the National Bureau of Standards*, 71B:233-240, 1967.

[68] F. Bock, *An Algorithm to Construct a Minimum Spanning Tree in a Directed Network*, Developments in Operations Research, Gordon and Breach, NY, 1971.

[69] S. Ramamurthy and B. Mukherjee, "Survivable WDM Mesh Networks, Part I - Protection", *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, New York, NY, March 1999.

[70] N.K. Singhal, C. Ou, and B. Mukherjee, "Cross-sharing vs. Self-sharing Trees for Protecting Multicast Sessions in Mesh Networks", *Elsevier Computer Networks*, 50(2):200-206, 2006.

[71] X. Wang, L. Guo, L. Pang, J. Du, and F. Jin, "Segment Protection Algorithm with Load Balancing for Multicasting WDM Mesh Networks", *Proc. International Conference on Advanced Communication Technology (ICACT)*, Shenyang, China, February 2008.

[72] L. Guo, X. Zhang, B. Han, J. Wu, W. Hou, and Y. Li, "Heuristic Protection Algorithm in Multicast Multi-Domain Optical Networks", *Proc. International Conference on Communication Software and Networks (ICCSN)*, Shenyang, China, May 2011.

[73] C.K. Constantinou and G. Ellinas, "Calculation of Arc-Disjoint Trees in Mixed-Graph Arbitrary Mesh Optical Networks", *Proc. Reliable Networks Design and Modeling (RNDM)*, St. Petersburg, Russia, October 2012.

[74] C.K. Constantinou and G. Ellinas, "Survivable Multicast Routing in Mixed-Graph Mesh Optical WDM Networks", *Proc. Optical Networking Design and Modeling (ONDM)*, Colchester, UK, April 2012.

[75] C.K. Constantinou and G. Ellinas, "A Novel Technique for Survivable Multicast Routing in Optical WDM Mesh Networks", *Proc. European Conference on Optical Communication (ECOC)*, Geneva, Switzerland, September 2011.

[76] T. Panayiotou, G. Ellinas, N. Antoniades, and A. Levine, "Designing and Engineering Metropolitan Area Transparent Optical Networks for the Provisioning of Multicast Sessions", *Proc. IEEE/OSA Optical Fiber Communications Conference (OFC)*, San Diego, CA, March 2010.

[77] B. Mukherjee, *Optical Communication Networks*, Mc-Graw-Hill, New York, 1997.

APPENDIX A: Graph Theory Definitions

Some important graph theory definitions that are used throughout the thesis are presented in this appendix. These can be found in [62], [63], as well as other sources.

- A *graph* $G(V, E)$ is a finite, nonempty set V together with a (possibly empty) set E (disjoint from V) of two-element subsets of (distinct) elements of V.
- Each element of V is referred to as a *vertex* or *node*. The members of set E are called *edges*.
- The *degree* (d_v) of a vertex v in a graph G is the number of edges incident to v.
- A *trivial* graph is a graph with one vertex and no edges.
- A graph is considered *connected* if there is a path between every pair of its vertices.
- It is called *simple* graph if it has no self-loops or multiple edges having the same pair of end vertices.
- In the case that a real number is assigned as a weight (or cost) for every edge, then the graph is *weighted* ($c(i, j)$ denotes the cost of edge (i, j)).
- Unidirectional edges are defined as *arcs* and bidirectional as *links*. Therefore, every link consists of a pair of opposite arcs (arcs with opposite orientation and equal weight).
- A graph is considered as *undirected* if all edges are links, *directed* if all edges are arcs, and *mixed* if it has both links and arcs.
- A directed or mixed graph that contains a directed path between every pair of its vertices is *strongly connected*. It is *weakly connected* if it has this feature only under the replacement of all its arcs with links.
- The indegree (outdegree) of vertex v, $id(v)$ ($od(v)$) corresponds to the inbound (outbound) edges incident to vertex v.
- A *tree* of a graph G is a connected acyclic directed subgraph of G.
- The *weight* (or *cost*) of a tree is defined as the sum of the weights (or costs) of its arcs.
- If the tree spans all the vertices of G, it is called *spanning* tree. The *Minimum Spanning Tree* (MST) is the spanning tree with the minimum weight.

- If the tree spans a subset of the vertices of G (but not all), it is called *Steiner* tree. The *Steiner Minimal Tree* (SMT) is the Steiner tree with the minimum weight.
- A *cycle* is a path starting and ending on the same vertex, while a *cycle branch* is a path that the starting vertex belongs to the cycle and the ending vertex and the intermediate ones do not belong to it.

APPENDIX B: Publications

Archived Journal Publications

1. C. Constantinou and G. Ellinas, "A Novel Multicast Routing Algorithm and its Application for Protection against Single-Link and Single-Link/Node Failure Scenarios in Optical WDM Mesh Networks", *OSA Optics Express*, 19(26):B471-B477, November 2011.

2. C.D. Charalambous and C. Constantinou, "Capacity of the Class of Channels with Incomplete CDI: Properties of Mutual Information for a Class of Channels", *IEEE Transactions on Information Theory*, 55(8):3725–3734, August 2009.

Peer-Reviewed Conference Proceedings

3. C. Constantinou and G. Ellinas, "A Novel Technique for Survivable Multicast Routing in Optical WDM Mesh Networks", *Proc. European Conference on Optical Communications (ECOC)*, Geneva, Switzerland, September 2011.

4. C. Constantinou and G. Ellinas, "Survivable Multicast Routing in Mesh Optical Networks with Mixed Graph Topologies", *Proc. IEEE 16th International Conference on Optical Network Design and Modeling (ONDM)*, Colchester, UK, April 2012.

5. C. Constantinou and G. Ellinas, "Calculation of Arc-Disjoint Trees in Mixed-Graph Arbitrary Mesh Optical Networks", *Proc. IEEE 4th International Workshop on Reliable Networks Design and Modeling (RNDM)*, St. Petersburg, Russia, October 2012.

6. C. Constantinou and G. Ellinas, "A Heuristic Algorithm for Multicast Routing in Sparse Splitting Optical WDM Networks", *Proc. IEEE 17th International Conference on Optical Network Design and Modeling (ONDM)*, Brest, France, April 2013.

7. C.D. Charalambous, S.Z. Denic, and C. Constantinou, "Information Capacity of MIMO Channels with Relative Entropy Constraint", *Proc. IEEE International Symposium on Information Theory (ISIT)*, Seattle, WA, July 2006.