

## ABSTRACT

Title of dissertation: OPTIMIZATION OF CONTEMPORARY  
TELECOMMUNICATIONS NETWORKS:  
GENERALIZED SPANNING TREES AND  
WDM OPTICAL NETWORKS

Daliborka Stanojević, Doctor of Philosophy, 2005

Dissertation directed by: Associate Professor Subramanian Raghavan  
Department of Decision and Information Technologies

We present a study of two NP-hard telecommunications network design problems - the prize-collecting generalized minimum spanning tree problem (PCGMST) and the design of optical networks with wavelength division multiplexing.

The first problem, the PCGMST problem, involves the design of regional backbone networks, where a set of local area networks (LANs) need to be connected by a minimum cost tree network using exactly one gateway site from each LAN. We present several polynomial time heuristics for the PCGMST problem and show that these algorithms, at best, provide only modest quality solutions. We also present two metaheuristics - a local search procedure and a genetic algorithm, and show that these procedures provide compelling high-quality results on a large set of test problems. Our study of the PCGMST problem is concluded by a presentation of two exact solution procedures that can be used to find optimal solutions in networks of moderate size.

The second problem studied in this dissertation is a more complex network design problem that involves optical networks with wavelength division multiplexing (WDM). These networks provide an abundance of transmission bandwidth, but require the use of expensive equipment, which, in turn, mandates careful use of the resources available for their design. The novel aspect of WDM optical networks is that they require simultaneous design of two network layers. The first layer is the virtual topology that requires routing of logical paths over the physical layer of optical fibers. The second layer involves routing and grooming of traffic requests over the logical paths established in the virtual topology. This problem has been extensively studied in the last 10 years,

but due to its notoriously hard nature, only few exact solution procedures for relaxed versions of this problem were developed so far. We propose one exact and two approximate branch-and-price algorithms for two versions of the WDM optical network design problem and present results of the computational study involving two different design objectives.

Finally, we propose two classes of valid inequalities for our branch-and-price algorithms, and discuss applicability of our algorithms to different versions of the WDM optical network design problem.

OPTIMIZATION OF CONTEMPORARY  
TELECOMMUNICATIONS NETWORKS:  
GENERALIZED SPANNING TREES AND WDM OPTICAL NETWORKS

by

Daliborka Stanojević

Dissertation submitted to the Faculty of the Graduate School of the  
University of Maryland, College Park in partial fulfillment  
of the requirements for the degree of  
Doctor of Philosophy  
2005

Advisory Committee:

Associate Professor Subramanian Raghavan, Chair/Advisor  
Professor Bruce L. Golden  
Professor Michael O. Ball  
Professor Gnanalingam Anandalingam  
Professor Ali Haghani

© Copyright by  
Daliborka Stanojević  
2005

To my little angel, Rajka.

## ACKNOWLEDGMENTS

First, I would like to thank my advisor, Dr. S. Raghavan, for all his support in my research work. His expertise has been of great help over the last four years.

My special thanks go to Dr. Bruce Golden, who has provided invaluable knowledge and advice whenever I needed it. I cannot express enough gratitude for the academic guidance that he has been continuously providing from the day I joined the PhD program.

I would also like to thank other members of my dissertation committee, Dr. G. Anandalingam, Dr. Michael Ball, and Dr. Ali Haghani, for their useful comments.

I owe my deepest thanks to Dr. Dušan Teodorović and Dr. Paul Schonfeld for their encouragement and help to start the graduate studies at the University of Maryland.

I would also like to thank my family for their unconditional love and understanding.

Finally, I would like to thank my friends Sriram Chellappan and Somnuk Ngamchai for always being there for me.

## TABLE OF CONTENTS

List of Tables	vi
List of Figures	x
1 Introduction	1
2 The Prize-Collecting Generalized Minimum Spanning Tree Problem	4
2.1 Introduction	4
2.2 Formal Problem Statement	6
2.3 Literature Review	7
2.4 Mathematical Formulations for the PCGMST problem	9
3 Heuristic Procedures for the Prize-Collecting Generalized Minimum Spanning Tree Problem	24
3.1 Simple Lower and Upper Bound Algorithms	24
3.1.1 Lower Bound Algorithm	24
3.1.2 Upper Bound Algorithms	25
3.2 Metaheuristic Procedures	34
3.2.1 Local Search	35
3.2.2 Genetic Algorithm	38
3.2.3 Computational Experiments	41
3.3 Conclusion	44
4 Exact Algorithms for the Prize-Collecting Generalized Minimum Spanning Tree Problem	46
4.1 Rooting Procedure	46
4.2 A Simple Branch-and-Cut Algorithm	50
4.3 Computational Results	52
4.4 Conclusion	60
5 Design of Optical Networks with Wavelength Division Multiplexing	61
5.1 Introduction	61
5.2 Important aspects of WDM optical network design	61
5.2.1 Physical Topology	63
5.2.2 Logical Topology	65
5.2.3 Traffic Requirements	66
5.2.4 Design Objectives	67
5.3 Problem Statement	69
5.4 Arc Based Mathematical Formulations for the WDM Optical Network Design	72
5.5 Impact of Different Network Restrictions on the Complexity of Mathematical Formulations for the WDM Optical Network Design Problem	78
5.6 Existing solution approaches for the design of WDM optical networks	83
5.6.1 Lower Bound Procedures	84
5.6.2 Upper Bound Procedures	87
5.6.3 Exact Solution Procedures	91
5.7 Conclusion	94
6 Branch-and-Price Algorithms for WDM Optical Network Design	95
6.1 Introduction	95
6.2 Review of the Branch-and-Price Algorithms	96
6.3 Path Based Mathematical Formulations for the WDM Optical Network Design Problem	100
6.3.1 MIP-PATH-LOCAL Formulation	101
6.3.2 MIP-PATH-GLOBAL Formulation	102

6.4	Review of Column Generation and Branch-and-Price Algorithms for the WDM Optical Network Design Problem . . . . .	104
6.4.1	Branch-and-Price Algorithm for the Special Case of the WDM Optical Network Design Problem with Infinite Number of Wavelengths . . . . .	108
6.4.2	Existing Column Generation and Branch-and-Price Algorithms for the WDM Optical Network Design Problem . . . . .	110
6.5	Branch-and-Price Framework for the WDM Optical Network Design Problem . . .	113
6.5.1	Branch-and-Price Algorithm for the MIP-PATH-LOCAL formulation . . . . .	113
6.5.2	Applying the Branch-and-Price Algorithms for MIP-PATH-LOCALw and MIP-PATH-LOCAL to WDM Optical Network Design with Alternative Design Objectives . . . . .	133
6.5.3	Branch-and-Price-and-Cut-Algorithm for the MIP-PATH-GLOBAL formulation . . . . .	135
6.5.4	Applying the Branch-and-Price Algorithm for MIP-PATH-GLOBAL to WDM Optical Network Design with Alternative Design Objectives . . . . .	138
6.6	Valid Inequalities for WDM Optical Network Design Formulations . . . . .	139
6.6.1	Lifted Cover Inequalities for the WDM optical network design problem . . .	140
6.6.2	Valid Inequalities for WDM Optical Networks that Guarantee Service for all Traffic Requests . . . . .	143
6.7	Computational Experiments . . . . .	144
6.8	Conclusion . . . . .	151
7	Extensions to WDM Optical Networks with Alternative Design Requirements	157
7.1	Introduction . . . . .	157
7.2	WDM Optical Networks with Traffic Bifurcation . . . . .	157
7.3	WDM Optical Networks with Infinite Number of Wavelengths . . . . .	160
7.4	WDM Optical Networks with Traffic Bifurcation and Infinite Number of Wavelengths	165
7.5	Conclusion . . . . .	169
8	Summary and Conclusions	171
	Bibliography	175



LIST OF TABLES

3.1	Comparison of Straightforward Adaptations of Kruskal’s, Prim’s, and Sollin’s Upper Bound and the Lower Bound Procedure for the PCGMST problem with node weights set to zero. . . . .	30
3.2	Comparison of Straightforward Adaptations of Kruskal’s, Prim’s, and Sollin’s Upper Bound and the Lower Bound Procedure for the PCGMST problem with the integer node weights randomly selected in the range [0, 10]. . . . .	30
3.3	Comparison of Improved Adaptations of Kruskal’s (IAK), Prim’s (IAP) and Sollin’s (IAS) with node weights set to zero. . . . .	32
3.4	Comparison of Improved Adaptations of Kruskal’s (IAK), Prim’s (IAP) and Sollin’s (IAS) with the integer node weights randomly selected in the range [0, 10]. . . . .	32
3.5	Comparison of Pilot Method for Adaptations of Kruskal’s (PAK), Prim’s (PAP) and Sollin’s (PAS) for the PCGMST problem with nodes weights set to zero. . . . .	34
3.6	Comparison of Pilot Method for Adaptations of Kruskal’s (PAK), Prim’s (PAP) and Sollin’s (PAS) for the PCGMST problem with the integer node weights randomly selected in the range [0, 10]. . . . .	35
3.7	Summary of computational results for the PCGMST problem with node weights set to zero. . . . .	43
4.1	Computational results for $RP_{rand}$ and $RP_{cycle}$ on TSPLIB instances with the center clustering procedure. (Solutions with an asterisk indicate lower bounds for instances where $RP_{rand}$ and $RP_{cycle}$ did not find the optimum within a two hour CPU time limit.) . . . . .	49
4.2	Comparison of Branch-and-Cut algorithms $BCA_{inc}$ and $BCA_{all}$ for the PCGMST problem. (Note: The TSPLIB instances used for these tests were modified by adding randomly generated integer node weights in the range [0, 10].) . . . . .	52
4.3	Computational results for $BCA_{inc}$ , LS and GA on TSPLIB instances with the center clustering procedure where $BCA_{inc}$ found the optimal solution within 2 hours of CPU time. Both LS and GA found the optimal solutions in all instances . . . . .	55
4.4	Computational results for $BCA_{inc}$ , LS and GA on TSPLIB instances with grid clustering procedure, $\mu = 3$ and 5, where $BCA_{inc}$ found the optimal solution within 2 hours of CPU time. Both LA and GA found the optimal solutions in all instances. . . . .	56
4.5	Computational results for $BCA_{inc}$ , LS and GA on TSPLIB instances with grid clustering procedure, $\mu = 7$ and 10, where $BCA_{inc}$ found the optimal solution within 2 hours of CPU times. Both LA and GA found the optimal solutions in all instances. . . . .	57
4.6	Computational results for $BCA_{inc}$ , LS and GA on TSPLIB instances where the optimal solution is unknown. (The time limit for $BCA_{inc}$ was 2 hours.) . . . . .	58
4.7	Computational Results for $BCA_{inc}$ , LS and GA on Random Instances (* - $BCA_{inc}$ did not complete search for this problem within 2 hours of CPU time. Upper bound at the time search was terminated was -194.) . . . . .	59

4.8	Summary of computational results for two different node-prize functions in instances where $BCA_{inc}$ found optimal solution within a 2 hour CPU time limit. (Our LS did not find optimal solution in 8 instances for the first node-prize function. In all other instances both LS and GA found optimal solutions.) . . . . .	60
5.1	Impact of different network restrictions on the nature of the corresponding network design problem . . . . .	79
6.1	Demand matrix for 6-node network example in Figure 6.5 . . . . .	145
6.2	Demand matrix for the NSFNET network shown in Figure 6.6 . . . . .	146
6.3	Minimizing the lost traffic in the network. Complete physical network with 2 fibers (fiber capacity: 2 lightpaths) between all pairs of nodes. . . . .	151
6.4	Minimizing the total number of transmitters and receivers in the network. Complete physical network with 2 fibers (fiber capacity: 2 lightpaths) between all pairs of nodes.	151
6.5	Minimizing the lost traffic in the network. Incomplete physical network with 2 fibers (fiber capacity: 20 lightpaths) on all arcs. . . . .	152
6.6	Minimizing the total number of transmitters and receivers in the network. Incomplete physical network with 2 fibers (fiber capacity: 20 lightpaths) on all arcs. . . . .	152
6.7	Minimizing the lost traffic in the network. The 6-node network with a single fiber on each arc. . . . .	152
6.8	Minimizing the lost traffic in the network. The NSFNET network with a single fiber on each arc. . . . .	152
6.9	Minimizing the lost traffic in the network (use of lifted cover inequalities). Selected instances for a complete physical network with 2 fibers (fiber capacity: 2 lightpaths) between all pairs of nodes. . . . .	153
6.10	Minimizing the lost traffic in the network (use of lifted cover inequalities). Incomplete physical network with 2 fibers (fiber capacity: 20 lightpaths) on all arcs. . . . .	153
6.11	Minimizing the total number of transmitters and receivers in the network (use of node degree inequalities). Selected instances for a complete physical network with 2 fibers (fiber capacity: 2 lightpaths) between all pairs of nodes. . . . .	153
6.12	Minimizing the total number of transmitters and receivers in the network (use of node degree inequalities). Incomplete physical network with 2 fibers (fiber capacity: 20 lightpaths) on all arcs. . . . .	153
6.13	Summary: Minimizing the lost traffic in the network. . . . .	154
6.14	Summary: Minimizing the total number of transmitters and receivers in the network. (We have assumed an upper bound of 380 for one instance of LOCALw where we did not have an integer solution.) . . . . .	154
6.15	Summary: Minimizing the total lost traffic in the network. Lifted Cover Inequalities. (The numbers in parenthesis indicate results for the corresponding branch-and-price algorithms when lifted cover inequalities are not used.) . . . . .	155

6.16	Summary: Minimizing the total number of transmitters and receivers in the network. Node Degree Inequalities. (The numbers in parenthesis indicate results for the corresponding branch-and-price algorithms when node degree inequalities are not used.) . . . . .	155
7.1	Minimizing the lost traffic in the network. Bifurcation of flow allowed. Selected instances for a complete physical network with 2 fibers (fiber capacity: 2 lightpaths) between all pairs of nodes. . . . .	160
7.2	Minimizing the lost traffic in the network. Bifurcation of flow allowed. Incomplete physical network with 2 fibers (fiber capacity: 20 lightpaths) on all arcs. . . . .	160
7.3	Minimizing the total number of transmitters and receivers in the network. Bifurcation of flow allowed. Selected instances for a complete physical network with 2 fibers (fiber capacity: 2 lightpaths) between all pairs of nodes. . . . .	161
7.4	Minimizing the total number of transmitters and receivers in the network. Bifurcation of flow allowed. Incomplete physical network with 2 fibers (fiber capacity: 20 lightpaths) on all arcs. . . . .	161
7.5	Summary: Minimizing the lost traffic in the network. Bifurcation of flow allowed. . . . .	161
7.6	Summary: Minimizing the total number of transmitters and receivers in the network. Bifurcation of flow allowed. . . . .	161
7.7	Minimizing the lost traffic in the network. Infinite number of wavelengths. Selected instances for a complete physical network with 2 fibers (fiber capacity: 2 lightpaths) between all pairs of nodes. . . . .	164
7.8	Minimizing the lost traffic in the network. Infinite number of wavelengths. Incomplete physical network with 2 fibers (fiber capacity: 20 lightpaths) on all arcs. . . . .	164
7.9	Minimizing the total number of transmitters and receivers in the network. Infinite number of wavelengths. Selected instances for a complete physical network with 2 fibers (fiber capacity: 2 lightpaths) between all pairs of nodes. . . . .	164
7.10	Minimizing the total number of transmitters and receivers in the network. Infinite number of wavelengths. Incomplete physical network with 2 fibers (fiber capacity: 20 lightpaths) on all arcs. . . . .	165
7.11	Summary: Minimizing the lost traffic in the network. Infinite number of wavelengths. . . . .	165
7.12	Summary: Minimizing the total number of transmitters and receivers in the network. Node Degree Inequalities. Infinite number of wavelengths. . . . .	165
7.13	Minimizing the lost traffic in the network. Bifurcation of flow allowed and infinite number of wavelengths. Selected instances for a complete physical network with 2 fibers (fiber capacity: 2 lightpaths) between all pairs of nodes. . . . .	167
7.14	Minimizing the lost traffic in the network. Bifurcation of flow allowed and infinite number of wavelengths. Incomplete physical network with 2 fibers (fiber capacity: 20 lightpaths) on all arcs. . . . .	168

7.15	Minimizing the total number of transmitters and receivers in the network. Bifurcation of flow allowed and infinite number of wavelengths. Selected instances for a complete physical network with 2 fibers (fiber capacity: 2 lightpaths) between all pairs of nodes. . . . .	168
7.16	Minimizing the total number of transmitters and receivers in the network. Bifurcation of flow allowed and infinite number of wavelengths. Incomplete physical network with 2 fibers (fiber capacity: 20 lightpaths) on all arcs. . . . .	168
7.17	Summary: Minimizing the lost traffic in the network. Bifurcation of flow allowed and infinite number of wavelengths. . . . .	169
7.18	Summary: Minimizing the total number of transmitters and receivers in the network. Bifurcation of flow allowed and infinite number of wavelengths. . . . .	169
7.19	Overall Results. Minimizing the total lost traffic in the network. . . . .	169
7.20	Overall Results. Minimizing the total number of transmitters and receivers. . . . .	169

## LIST OF FIGURES

2.1	Example of regional backbone network interconnecting three LANs through single gateway node in each LAN . . . . .	5
2.2	Generalized spanning tree corresponding to the example of the regional backbone network shown in Figure 2.1 . . . . .	6
2.3	Relationship between polyhedrons defined by linear relaxations of corresponding GMST formulations . . . . .	11
3.1	Pilot Method for the Upper Bound Heuristics for the PCGMST problem . . . . .	33
3.2	An example of a feasible GST and the corresponding problem state representation	36
3.3	Steps of our Genetic Algorithm. . . . .	39
3.4	One-point crossover operator example. . . . .	40
4.1	Steps of our implementation of the Rooting Procedure ( $RP_{cycle}$ ). . . . .	48
4.2	Steps of our Branch-and-Cut Algorithm ( $BCA_{inc}$ ). . . . .	51
5.1	ISO standard for telecommunications network structure . . . . .	62
5.2	Change in the complexity of telecommunications networks . . . . .	63
5.3	A five-node example of the WDM optical network design problem. (a) Physical Topology. (b) List of commodities and the corresponding demand. (c) Feasible logical topology and traffic routing. (d) Optimal logical topology and traffic routing.	73
5.4	Heuristic Procedures for the WDM optical network design problem . . . . .	84
5.5	Exact Procedures for the WDM optical network design problem . . . . .	85
5.6	An example of the chromosome representation used in the Multi-Objective Evolutionary Algorithm [50] . . . . .	89
6.1	Branch-and-Price Algorithm for the ODIMCF problem (Barnhart et al., 2000) . .	107
6.2	Steps of column generation algorithm for MIP-PATH-LOCALw and MIP-PATH-LOCAL formulations . . . . .	123
6.3	Example of the branching issue for the flow path visiting the same node more than once in the physical topology . . . . .	126
6.4	Example of the branching issue for the MIP-PATH-LOCALw formulation . . . . .	131
6.5	Physical topology of the 6-node optical network . . . . .	145
6.6	Physical topology of the NSFNET optical network . . . . .	146

## Chapter 1

### Introduction

Network design problems refer to the large class of problems that find applications in many different areas of industry, and natural and social sciences. In this dissertation we focus on two network design problems that arise in telecommunications.

The first problem is the prize-collecting generalized minimum spanning tree (PCGMST) problem, which arises in the design of regional backbone telecommunications networks. In this problem, a set of local area networks needs to be connected by a tree structure, and for that purpose exactly one gateway site needs to be selected out of a set of candidate sites from each region. The main component of the cost related to the design of backbone networks is usually related to the cost of the transmission facilities laid out between gateways of different local area networks. Aside from this cost, different gateway sites may offer compensation or “prize” if selected for the backbone network design. Given these two monetary factors, the design objective in this problem is to select gateway sites and connections between these gateways so that the total cost of the transmission facilities minus the total revenue from selected gateways is minimized. We present this problem in Chapters 2, 3, and 4.

In Chapter 2, we provide a literature review of the different versions of the PCGMST problem. Additionally, we provide a comprehensive review of the mathematical formulations developed for this problem so far. In this chapter, we also point out some important properties of one of the formulations for the PCGMST problem and explain how these properties can be used to develop simple exact procedures for this problem.

In Chapter 3, we present several heuristic and metaheuristic algorithms for the PCGMST problem. First, we describe a simple lower bound algorithm for the PCGMST problem. Next, we show how the well-known minimum spanning tree algorithms (Kruskal’s, Prim’s, and Sollin’s) can be adapted for the PCGMST problem. We also describe simple modifications of these algorithms

that significantly improve their performance. In the remaining part of Chapter 3 we propose two new metaheuristic procedures—local search and genetic algorithm. The chapter is concluded with a computational study of these metaheuristic procedures for a special case of the PCGMST problem where node prizes are equal and, therefore, set to zero. Computational experiments with these procedures for problems with positive node weights are presented in Chapter 4.

In Chapter 4, we present two exact solution procedures for the PCGMST problem. The first one represents an improved version of the existing solution procedure called the Rooting procedure that was developed by Pop in [46]. This procedure is a special form of delayed column and row generation, and we propose a new strategy that can be used to provide better performance of this procedure. The second exact procedure is a simple branch-and-cut algorithm that utilizes simple depth first search to identify violated inequalities at the incumbent (integer) nodes of the branch-and-bound tree. The computational study presented in this chapter provides results of these procedures for two different functions used to define the node prizes.

The second problem that we study in this dissertation is the design of optical networks with wavelength division multiplexing (WDM). WDM optical networks have received a lot of research attention in the last 10 years due to enormous bandwidth provided by optical fibers and the expensive optical and/or electronic equipment needed to process traffic requests in these networks. But, despite the importance of this problem, most researchers have developed only heuristic procedures for the WDM optical network design problem. Also, the few studies that presented exact solution procedures often apply to only relaxed versions of this problem. The reason for this situation is that the WDM optical network design problem is a two layer network design problem that is much more difficult to solve than the corresponding single layer problems.

In this dissertation, we mainly focus on WDM optical networks with wavelength conversion and without bifurcation of traffic (that is, single traffic requests use a single path in the network). In this specific optical network setting, we are given a physical network with a set of restrictions on its nodes and links. The goal is to simultaneously design a logical topology consisting of lightpaths routed over the physical network, and determine the routing of the traffic requests over the logical

topology, so that a pre-specified measure of network performance is optimized. We present our study of this problem in Chapters 5, 6 and 7.

In Chapter 5, we first review important aspects of the WDM optical network design and then explain the impact of these aspects on the complexity of the potential solution procedures. Part of this discussion is done with respect to two mathematical formulations for the WDM optical network design problem that we present in this chapter. The chapter concludes with a review of the existing heuristics and exact solution procedures related to the version of the WDM optical network design problem studied in this dissertation.

In Chapter 6, we provide a brief review of the column generation concept, and then describe the relevant existing column generation and branch-and-price procedures. We then present branch-and-price algorithms for the mathematical formulations presented in this chapter. We also propose two classes of valid inequalities for our mathematical formulations, and discuss the implementation issues when these inequalities are used in the branch-and-price framework. Finally, we provide results of our computational study of the proposed branch-and-price algorithms.

In Chapter 7, we discuss the applicability of our branch-and-price algorithms to WDM optical networks with different network settings. Specifically, we look at networks where bifurcation of flow is allowed, networks where ample optical fiber capacity exists, and networks where both bifurcation of flow is allowed and there is ample optical fiber capacity. We conclude this chapter with the computational results for the proposed branch-and-price algorithms in each of the three network settings discussed.

This dissertation concludes with Chapter 8, where we summarize our contributions and discuss directions for future research.



## Chapter 2

### The Prize-Collecting Generalized Minimum Spanning Tree Problem

#### 2.1 Introduction

In regional telecommunication networks, it is often necessary to design cost-effective backbone networks that connect sets of local area networks (LANs). The connection of LANs in the regional network can be established using different network designs, however, it is common to consider tree-like network structures in order to lower the number of telecommunication facilities and equipment needed to establish connection between all LANs. Typically, each LAN is connected to the backbone network through one or more telecommunication centers that serve as gateways between its own users and other LANs in the region. And, although each LAN may have several candidate sites that could serve as gateways, the cost of telecommunication facilities and equipment also motivates the design requirement that a single gateway is used in each LAN. This problem of designing the regional backbone network, with a tree structure spanning exactly one candidate gateway in each LAN, is known in the literature as the **generalized minimum spanning tree** (GMST) problem. Figure 2.1 illustrates an example of the generalized minimum spanning tree where the regional backbone network is constructed by selecting exactly one gateway site from each LAN and connecting the selected sites by a tree-like structure.

In certain situations, the impact of different candidate gateways in any given LAN can be taken into account through modification of the cost of transmission facilities needed to connect a specific candidate gateway to other gateways in the region. For example, if, due to compatibility (or some other reason), the direct connection of two specific gateways from different LANs is less desirable than some other connections, we could add a “penalty” cost to the actual cost of the transmission facility needed to be laid out in order to establish that connection. On the other hand, candidate gateway sites in a LAN may have an incentive to be selected for the regional

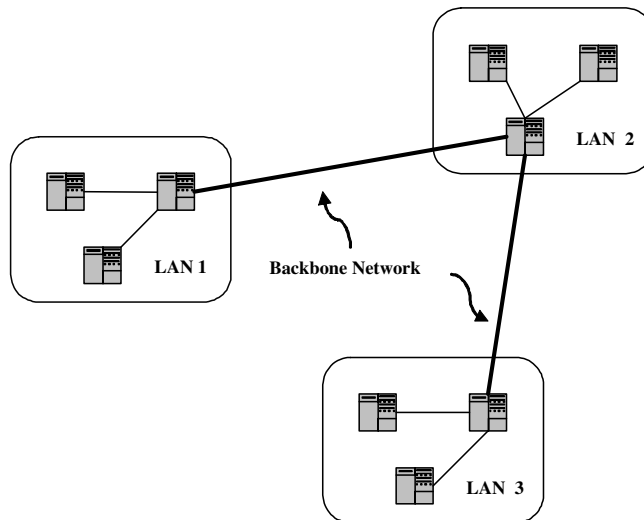


Figure 2.1: Example of regional backbone network interconnecting three LANs through single gateway node in each LAN

backbone network, and therefore they may offer compensation (or “prize”) that can be collected if that site is selected for the backbone network. In this case, we cannot just modify the cost of transmission facilities in order to take into account compensations offered by the selected gateway sites. The reason is that a single gateway site may be incident to two or more physical links. Since the number of the physical links incident to any given site is not known in advance, there is no way to know which edges should be used to take into account the prize offered by a given gateway site. For this reason, whenever gateway site compensation needs to be considered, the design objective becomes that of minimization of the total cost of the transmission facilities used minus the sum of compensations offered by the selected gateway sites. We will refer to this problem as the **prize-collecting generalized minimum spanning tree** (PCGMST) problem. The PCGMST problem is studied in Chapters 2, 3, and 4 in this dissertation. Chapter 2 provides a literature review for the PCGMST problem. In Chapter 3, several heuristic approaches, including 2 efficient metaheuristics, are presented. Finally, in Chapter 4, we propose two exact solution procedures for the PCGMST problem.

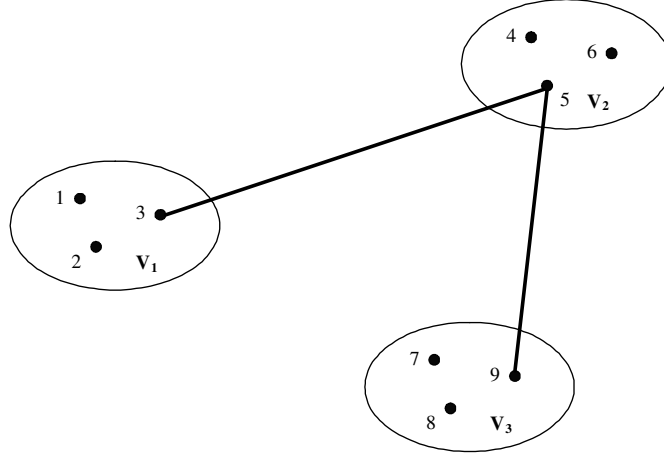


Figure 2.2: Generalized spanning tree corresponding to the example of the regional backbone network shown in Figure 2.1

## 2.2 Formal Problem Statement

The prize-collecting generalized minimum spanning tree problem can be modelled in a straightforward manner in graph theory terms. Each LAN can be thought of as a cluster of nodes, with each node representing a candidate gateway site for the regional backbone network. Possible transmission facilities between LANs can be represented by edges interconnecting nodes between different clusters. Each node and edge in this graph are assigned a certain weight, where node weights correspond to node prizes, and edge weights correspond to edge costs. Using this approach, the prize-collecting generalized minimum spanning tree can be defined as follows.

**Problem Statement.** Given an undirected graph  $G = (V, E)$ , with node set  $V$ , edge set  $E$ , cluster set of  $K$  mutually exclusive and exhaustive node sets  $V_k$  ( $k = 1, \dots, K$ ), and cost vector  $c \in \mathbb{R}_+^{|E|}$  on the edges in  $E$ , and prize vector  $p \in \mathbb{R}_+^{|V|}$  on the nodes in  $V$ , find a tree spanning exactly one node from each cluster, such that the total cost of the edges used minus the total prize from all the nodes selected is minimized.

Figure 2.2 illustrates the network from the example shown in Figure 2.1 defined in graph theoretic terms.

## 2.3 Literature Review

The GMST problem appears in several versions in the literature. The version of the GMST problem studied in this dissertation was introduced by Myung et al. (1995), and has been studied by only a few researchers.

Myung et al. [45] have shown that the GMST problem is NP-hard. They have also developed a dual ascent procedure for this problem, which was used as part of a branch-and-bound procedure. That procedure was tested on networks with up to 100 nodes and 4,500 edges. Feremans [24] and Feremans et al. [25] presented several different formulations for the GMST problem and proposed a specialized branch and cut algorithm for this problem. The proposed branch-and-cut algorithm is based on several classes of new valid inequalities and two new metaheuristic algorithms, local search and tabu search, which were used to enhance the overall performance of the algorithm. In the computational study performed in [24], this procedure provided optimal solution for the GMST problem in networks with up to 160 nodes when edge costs satisfied triangle inequality. Even better performance of the proposed procedure was reported for networks with random edge costs where this procedure provided optimal solutions for networks with up to 200 nodes.

Pop [46] proposed a new polynomial-size formulation for the GMST problem and developed a new exact procedure for the GMST problem based on the proposed formulation. The novel aspect of the proposed procedure is that it is very simple and appears to be efficient for networks of moderate size. Pop [46] and Pop et al. [49] have also proposed a constant factor approximation for the GMST problem based on the cluster size bound.

Other versions of the GMST problem that are present in the literature are closely related to the more extensively studied Group Steiner Problem (GSP). In this sense, the first version of the GMST problem was introduced by Cockayne and Melzak (1968). The GSP studied by Cockayne [14] requires the design of a tree structure spanning *at least* one node from each cluster of nodes. Additionally, there may exist a set of Steiner nodes that does not belong to any of the clusters, but can be used for the tree design. Recently, Duin and Voß [19] have pointed out that when a specific GMST problem fits the framework of the GSP, one can use the transformation of the

GSP to the well studied undirected Steiner problem in graphs (SPG) to model the GMST problem. In their computational study, Duin and Voß [19] found that two specialized SPG heuristics, Pilot-Rush and Pilot-Drop (originally defined in [18]), provide good results for this special case of the GMST problem. On a set of problems with Euclidean, random, and rectilinear distances in networks with 100 to 400 nodes, the Pilot-Drop procedure provided solutions that were on average less than 0.3% from optimality and with a maximum gap of 3.4% from optimality. These heuristics performed similarly for a slightly different version of the GMST problem that is known in the literature as *at least* GMST problem. This problem differs from the GSP problem in the sense that the set  $V$  is completely covered by the set of clusters (i.e., every node belongs to one of the clusters). Duin and Voß have also shown that the exact SPG solver similar to the one developed by Duin [20] significantly outperforms a genetic algorithm (GA) developed by Dror et al. in [17], both in terms of the solution quality and the computational time. On a set of 20 problems, the GA developed by Dror et al. provided solutions that were on average 6.53% from optimality [24], while the exact SPG procedure provided optimal solutions for all test instances within very short CPU times. Shyu et al. [56] developed an ant colony approach that provides comparable results to the GA developed by Dror et al., but requires shorter CPU times for the same problem. Recently, Haouari [32] proposed an exact branch-and-bound algorithm for the *at least* version of the GMST problem. The proposed algorithm was combined with a specialized preprocessing algorithm, and was used to find optimal solutions for problems in networks with up to 250 nodes, 1000 edges, and 25 clusters within three hours of CPU time.

The heuristics developed in this dissertation are closely related and, in part, described in our work presented in [28] and [30]. Computational studies presented in this dissertation (and [28] and [30]) indicate a compelling performance of the two proposed metaheuristic algorithms—local search and genetic algorithm. As we will show in the computational sections in the following chapters, these algorithms provide optimal solutions for most of the test instances within very short CPU times.

## 2.4 Mathematical Formulations for the PCGMST problem

In this section we review different mathematical formulations for the GMST problem. We also examine the relationship between the linear relaxations of these formulations, and then explain the properties of one of these formulations that allows development of simple and effective exact solution procedures for the PCGMST problem. The ten formulations reviewed in this Chapter include:

1. Undirected cutset formulation (*ucut*)
2. Undirected subpacking formulation (*usub*)
3. Undirected cluster subpacking formulation (*ucsub*)
4. Directed cutset formulation (*dcut*)
5. Directed subpacking formulation (*dsub*)
6. Directed cluster subpacking formulation (*dcsub*)
7. Multiflow formulation (*mflow*)
8. Flow cut formulation (*fcut*)
9. Steiner tree formulation (*stree*)
10. Subtour elimination formulation (*subel*).

Formulations *ucut*, *usub*, and *mflow* were originally proposed by Myung et al. [45]. Formulation *dcut* was proposed by Feremans [24], and Feremans et al. [25], and is a more compact version of the equivalent formulation proposed by Myung et al. [45]. Feremans [24], and Feremans et al. [25] have also proposed formulation *dsub*, *fcut*, *ucsub*, and *dcsub*. The *stree* formulation was proposed by Raghavan [51], and the last formulation, the *subel* formulation was proposed by Pop [46]. Feremans [24] studied the relationship between the linear relaxations of the first eight formulations, while the strength of the linear relaxations of the *stree* formulation was proven by

Raghavan [51]. In Figure 2.3 we illustrate relationship between the linear relaxations of nine formulations that will be presented in this chapter (we use the same pictorial representation of polyhedrons of linear relaxations of different formulations presented in [24], but we also include the *stree* formulation). Each set in Figure 2.3 represents the polyhedron defined by the linear relaxation of the corresponding formulation. So, if in Figure 2.3 a set corresponding to one formulation is a subset of another formulation, it means that the linear relaxation of the first formulation is stronger than the linear relaxation of the second formulation. For example, set corresponding to the linear relaxation of the *usub* formulation is a subset of the linear relaxation of the *ucut*, which indicates that the linear relaxation of the *usub* formulation is stronger than the linear relaxation of the *ucut* formulation. Linear relaxation of the *subel* formulation is not included, as its relationship to other formulations has not been studied. Pop [46] did however show that this formulation does not describe the convex hull of the integer feasible region for the GMST problem. As Figure 2.3 indicates, five of the nine formulations are equivalent in terms of their linear relaxation.

We note here that the equivalence of linear relaxations of different formulation does not necessarily mean that these formulations are identical from a computational perspective. For example, our computational tests of the *mflow* and the *stree* formulation indicate that the *stree* formulation is able to solve larger instance of the problem (in the same amount of time) compared to the *mflow* formulation. We have also found that the *stree* formulation requires significantly shorter CPU times for problems in smaller networks.

Different formulation used to model the same problem may also differ in terms of the type of variables and constraints used. In some cases, mathematical formulations of a given problem may have special properties that can be used to develop an efficient exact solution procedures. The *subel* formulation is an example of one such formulation. In this chapter we will discuss special properties of the *subel* formulation, and, then, in Chapter 4, we will present two efficient exact solution procedures that are based on this formulation.

In the remaining part of this chapter, we will present the ten formulations for the GMST problem. We will first present three formulation defined on an undirected graph  $G = (V, E)$ , and

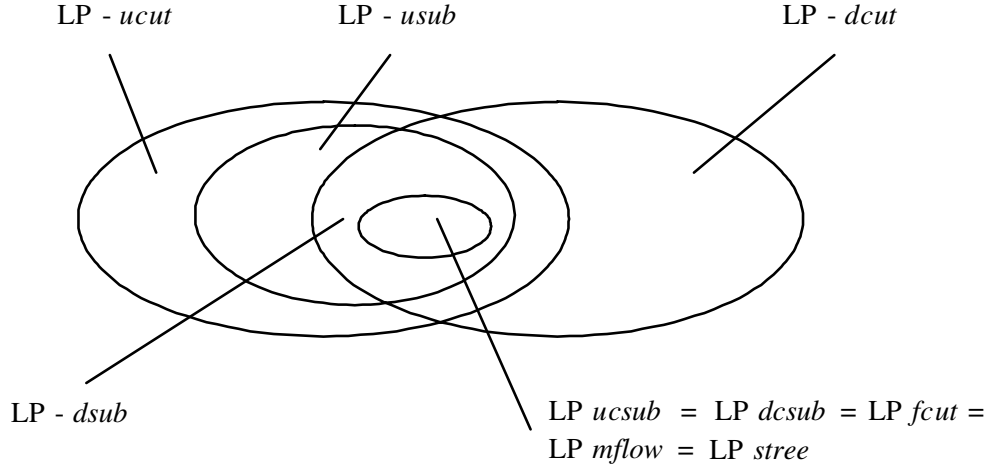


Figure 2.3: Relationship between polyhedrons defined by linear relaxations of corresponding GMST formulations

then we will present formulations that are defined on the directed graph  $G = (V, A)$  obtained by replacing each undirected edge  $(i, j)$  by two directed arcs  $(i, j)$  and  $(j, i)$ .

**Notation**

Input:

$c_{ij}$  - cost of edge  $(i, j)$

$p_i$  - prize offered by node  $i$

Variables:

$u_{ij}$  - indicator variable equal to 1 if **edge**  $(i, j)$  is used in the final solution

$z_i$  - indicator variable equal to 1 if node  $i$  is used in the final solution

Note that although the presented formulations are defined for the GMST problem, these formulations are also valid for the PCGMST problem. In the latter case we only need to add the prize term  $\sum_{\forall i} p_i z_i$  in the objective function.

**Undirected cutset formulation (*ucut*)**

$$\text{Min} \quad \sum_{\forall (i,j) \in E} c_{ij} u_{ij} \tag{2.1}$$



Subject to:

$$\sum_{i \in V_k} z_i = 1 \quad \forall k = 1, \dots, K \quad (2.2)$$

$$\sum_{\forall (i,j) \in E} u_{ij} = K - 1 \quad (2.3)$$

$$\sum_{\forall (l,m) \in E: l \in S \subset V, m \notin S} u_{lm} \geq z_i + z_j - 1 \quad \forall i \in S \subset V, j \notin S \quad (2.4)$$

$$z_i \in B^1 \quad \forall i \in V \quad (2.5)$$

$$u_{ij} \in B^1 \quad \forall (i,j) \in E \quad (2.6)$$

Constraint (2.2) ensures that exactly one node is selected from each cluster. Constraint (2.3) ensures that exactly  $K - 1$  edges are used in the GMST design. Finally, constraint (2.4) is a **cutset** constraint that ensures that the node selected for a given cluster is connected to the remaining part of the generalized spanning tree.

#### Undirected subpacking formulation (*usub*)

$$\text{Min} \quad \sum_{\forall (i,j) \in E} c_{ij} u_{ij} \quad (2.7)$$

Subject to:

$$\sum_{i \in V_k} z_i = 1 \quad \forall k = 1, \dots, K \quad (2.8)$$

$$\sum_{\forall (i,j) \in E} u_{ij} = K - 1 \quad (2.9)$$

$$\sum_{\forall (i,j) \in E: i \in S, j \in S} u_{ij} \leq \sum_{\forall i \in S \setminus \{r\}} z_i \quad r \in S \subset V, 2 \leq |S| \leq |V| - 1 \quad (2.10)$$

$$z_i \in B^1 \quad \forall i \in V \quad (2.11)$$

$$u_{ij} \in B^1 \quad \forall (i,j) \in E \quad (2.12)$$

This formulation is similar to the *ucut* formulation with the only difference that the **subpacking constraint** (2.10) takes the role of the cutset constraint (2.4). The subpacking constraint (2.10) guarantees that, for each subset of nodes  $S$  ( $S \subset V, 2 \leq |S| \leq |V| - 1$ ), and each node  $r$  ( $r \in S \subset V$ ), the number of edges selected over subset  $S$  cannot be greater than the number of nodes selected for the GMST design in the same set. (In case that node  $r$  is selected for the GMST design, the maximum number of edges selected within this set is equal to the number of nodes selected minus

1.)

**Undirected cluster subpacking formulation (*ucsub*)**

$$\text{Min} \quad \sum_{\forall(i,j) \in E} c_{ij} u_{ij} \quad (2.13)$$

Subject to:

$$\sum_{i \in V_k} z_i = 1 \quad \forall k = 1, \dots, K \quad (2.14)$$

$$\sum_{\forall(i,j) \in E} u_{ij} = K - 1 \quad (2.15)$$

$$\sum_{\forall(i,j) \in E: i \in S, j \in S} u_{ij} \leq \sum_{\forall i \in S} z_i - 1 \quad S \subset V, 2 \leq |S| \leq |V| - 1, |\{k : V_k \subseteq S\}| \neq 0 \quad (2.16)$$

$$z_i \in B^1 \quad \forall i \in V \quad (2.17)$$

$$u_{ij} \in B^1 \quad \forall(i,j) \in E \quad (2.18)$$

The *ucsub* formulation is similar to the *usub* and *ucut* formulations. It differs in a sense that the connectivity between all clusters is achieved by the **cluster subpacking** constraints (2.16). These constraints guarantee that the number of edges selected from any subset of nodes  $S$  ( $S \subset V, 2 \leq |S| \leq |V| - 1$ ), cannot be greater than the number of nodes selected from that set minus 1.

The remaining formulations presented in this chapter are defined on a directed graph  $G = (V, A)$ , so they use an additional set of binary variables  $x_{ij}$  defined for each arc  $(i, j) \in A$ . Any given variable  $x_{ij}$  is equal to 1 if the arc  $(i, j)$  is selected for the GMST design, and is equal to 0 otherwise.

**Directed cutset formulation (*dcut*)**

$$\text{Min} \quad \sum_{\forall(i,j) \in E} c_{ij} u_{ij} \quad (2.19)$$

Subject to:

$$\sum_{i \in V_k} z_i = 1 \quad \forall k = 1, \dots, K \quad (2.20)$$

$$\sum_{\forall(i,j) \in E} u_{ij} = K - 1 \quad (2.21)$$

$$\sum_{\forall(l,m) \in A: l \notin S, m \in S} x_{lm} \geq z_i \quad \forall S \subseteq V \setminus V_r, i \in S \quad (2.22)$$

$$x_{ij} \leq z_i \quad \forall i \in V_r, j \notin V_r \quad (2.23)$$

$$x_{ij} + x_{ji} = u_{ij} \quad \forall (i, j) \in E \quad (2.24)$$

$$z_i \in B^1 \quad \forall i \in V \quad (2.25)$$

$$x_{ij} \in B^1 \quad \forall (i, j) \in A \quad (2.26)$$

This formulation is a directed version of the *ucut* formulation, and it differs from *ucut* in sense that custset constraint (2.4) is replaced by constraints (2.22) - (2.24), which together with constraints (2.20) and (2.21) define an arborescence with a root (origin) node in the arbitrarily chosen root cluster  $V_r$  ( $r = 1, \dots, K$ ).

### Directed subpacking formulation (*dsub*)

$$\text{Min} \quad \sum_{\forall (i,j) \in E} c_{ij} u_{ij} \quad (2.27)$$

Subject to:

$$\sum_{i \in V_k} z_i = 1 \quad \forall k = 1, \dots, K \quad (2.28)$$

$$\sum_{\forall (i,j) \in E} u_{ij} = K - 1 \quad (2.29)$$

$$\sum_{\forall (i,j) \in A: i \in S, j \in S} x_{ij} \leq \sum_{\forall i \in S \setminus \{r\}} z_i \quad r \in S \subset V, 2 \leq |S| \leq |V| - 1 \quad (2.30)$$

$$x_{ij} + x_{ji} = u_{ij} \quad \forall (i, j) \in E \quad (2.31)$$

$$\sum_{\forall j \in V: (j,i) \in A} x_{ji} = z_i \quad i \in V \setminus V_r \quad (2.32)$$

$$z_i \in B^1 \quad \forall i \in V \quad (2.33)$$

$$x_{ij} \in B^1 \quad \forall (i, j) \in A \quad (2.34)$$

This formulation is a directed version of the *usub* formulation where set of constraints (2.30) - (2.32) is used instead of undirected subpacking constraint (2.10).

### Directed cluster subpacking formulation (*dclub*)

$$\text{Min} \quad \sum_{\forall (i,j) \in E} c_{ij} u_{ij} \quad (2.35)$$

Subject to:

$$\sum_{i \in V_k} z_i = 1 \quad \forall k = 1, \dots, K \quad (2.36)$$

$$\sum_{\forall(i,j) \in E} u_{ij} = K - 1 \quad (2.37)$$

$$x_{ij} + x_{ji} = u_{ij} \quad \forall(i,j) \in E \quad (2.38)$$

$$\sum_{\forall j \in V: (j,i) \in A} x_{ji} = z_i \quad i \in V \setminus V_r \quad (2.39)$$

$$\sum_{\forall(i,j) \in E: i \in S, j \in S} u_{ij} \leq \sum_{\forall i \in S} z_i - 1 \quad S \subset V, 2 \leq |S| \leq |V| - 1, |\{k : V_k \subseteq S\}| \neq 0 \quad (2.40)$$

$$z_i \in B^1 \quad \forall i \in V \quad (2.41)$$

$$x_{ij} \in B^1 \quad \forall(i,j) \in A \quad (2.42)$$

The *dcsb* formulation is a directed version of the *ucsub* formulation that strengthens the linear relaxation of the *ucsub* through additional constraints (2.38) and (2.39).

### Multiflow formulation (*mflow*)

$$\text{Min} \quad \sum_{\forall(i,j) \in E} c_{ij} u_{ij} \quad (2.43)$$

Subject to:

$$\sum_{i \in V_k} z_i = 1 \quad \forall k = 1, \dots, K \quad (2.44)$$

$$x_{ij} + x_{ji} = u_{ij} \quad \forall(i,j) \in E \quad (2.45)$$

$$\sum_{(j,i) \in A: i \in V_r, j \notin V_r} x_{ji} = 0 \quad (2.46)$$

$$\sum_{(j,i) \in A: i \in V_k, j \notin V_k} x_{ji} \leq 1 \quad \forall k = 1, \dots, K, k \neq r \quad (2.47)$$

$$\sum_{\forall j: (i,j) \in A} f_{ij}^k - \sum_{\forall j: (j,i) \in A} f_{ji}^k = \begin{cases} z_i & \text{if } i \in V_r; \\ -z_i & \text{if } i \in V_k; \quad \forall i \in V, k = 1, \dots, K, k \neq r \\ 0 & \text{otherwise;} \end{cases} \quad (2.48)$$

$$0 \leq f_{ij}^k \leq x_{ij} \quad \forall(i,j) \in A, k = 1, \dots, K, k \neq r \quad (2.49)$$

$$z_i \in B^1 \quad \forall i \in V \quad (2.50)$$

$$x_{ij} \in B^1 \quad \forall(i,j) \in A \quad (2.51)$$

In the *mflow* formulation the GMST problem is treated as a fixed charge network design problem with multicommodity flow requirements. One cluster is arbitrarily selected as the root cluster  $r$ ,

and  $K - 1$  commodities each originating at the root cluster and terminating in cluster  $k$  ( $k = 1, \dots, K, k \neq r$ ) are created. Constraints (2.48) are the flow balance constraints that send 1 unit of flow from the node selected in the root cluster to the node selected in cluster  $k$ .

**Flow cut formulation (*fcut*)**

$$\text{Min} \quad \sum_{\forall(i,j) \in E} c_{ij} u_{ij} \quad (2.52)$$

Subject to:

$$\sum_{i \in V_k} z_i = 1 \quad \forall k = 1, \dots, K \quad (2.53)$$

$$x_{ij} + x_{ji} = u_{ij} \quad \forall(i, j) \in E \quad (2.54)$$

$$\sum_{\forall(j,i) \in A: i \in V_r, j \notin V_r} x_{ji} = 0 \quad (2.55)$$

$$\sum_{\forall(j,i) \in A: i \in V_k, j \notin V_k} x_{ji} \leq 1 \quad \forall k = 1, \dots, K, k \neq r \quad (2.56)$$

$$\sum_{\forall(j,i) \in A: i \in S, j \notin S} x_{ji} \geq \sum_{\forall i \in V_k \cap S} z_i - \sum_{\forall i \in V_r \cap S} z_i \quad S \subset V, 1 \leq |S| \leq |V| - 1, k = 1, \dots, K, k \neq r \quad (2.57)$$

$$z_i \in B^1 \quad \forall i \in V \quad (2.58)$$

$$x_{ij} \in B^1 \quad \forall(i, j) \in A \quad (2.59)$$

The *fcut* formulation is closely related to the *mflow* formulation. The main difference between the two is that the flow balance constraints (2.48) and constraints (2.49) are replaced by a directed cutset constraint (2.57), which together with constraints (2.53) - (2.56) guarantees tree design and connectivity between all clusters.

**Steiner tree formulation (*stree*)**

$$\text{Min} \quad \sum_{\forall(i,j) \in A} c_{ij} x_{ij} \quad (2.60)$$

Subject to:

$$\sum_{(i,j) \in A: i \notin V_k, j \in V_k} x_{ij} = 1 \quad \forall k = 1, \dots, K, k \neq r \quad (2.61)$$

$$x_{ij} + x_{ji} \leq 1 \quad \forall(i, j) \in E \quad (2.62)$$

$$0 \leq f_{ij}^h \leq x_{ij} \quad \forall(i, j) \in A, h \in H \quad (2.63)$$

$$\sum_{\forall j:(j,i) \in A} f_{ji}^h - \sum_{\forall j:(i,j) \in A} f_{ij}^h = \begin{cases} -1 & \text{if } i = O(h); \\ 1 & \text{if } i = D(h); \\ 0 & \text{otherwise;} \end{cases} \quad \text{for all } i \in V \text{ and } h \in H \quad (2.64)$$

$$x_{ij} \in B^1 \quad \forall (i,j) \in A \quad (2.65)$$

This formulation is a multicommodity flow formulation for the GMST problem, once the GMST problem is transformed into a Steiner tree problem with degree constraints. The transformation of the GMST problem to a Steiner tree problem is performed using the following steps. Let  $s_r$  be an artificial root node, and  $T = \{t_1, t_2, \dots, t_k\}$  be a set of artificial sink nodes that are required to be in the Steiner tree. Artificial node  $s_r$  is connected to each node  $i \in V$  with zero cost arc  $(s_r, i)$ , and all nodes  $i \in V_k$  of a given cluster  $k = 1, \dots, K$  are connected with zero cost arc  $(i, t_k)$  to the corresponding artificial node  $t_k$ . In this new graph, a set of commodities  $H$  is also defined so that there is exactly one commodity  $h \in H$  originating at node  $s_r$  and terminating at node  $t_k$  for each  $k = 1, \dots, K$ . Formulation *stree* is then the standard multicommodity flow model for the Steiner tree problem with additional degree constraints (2.61) that ensure an in-degree of 1 for each of the clusters ( $k \in K$ ). This formulation ensures the design of a minimum-cost directed Steiner tree, containing root node  $s_r$ , all the nodes in  $T$ , and exactly one node from each cluster.

In [51], Raghavan has shown that formulation *stree* is equivalent to the multicommodity flow formulation *mflow*, where the node variables had to be used to indicate whether a node is selected to be in the GMST. In the *stree* formulation, on the other hand, the graph is expanded with additional arc variables without the use of node variables. As mentioned earlier, our computational experiments indicate that, although the two formulations are equivalent in terms of the strength of their linear relaxations, the *stree* formulation can be solved significantly faster using CPLEX.

#### Subtour elimination formulation (*subel*)

$$\text{Min} \quad \sum_{\forall (i,j) \in E} c_{ij} u_{ij} \quad (2.66)$$

Subject to:

$$\sum_{i \in V_k} z_i = 1 \quad \forall k = 1, \dots, K \quad (2.67)$$

$$\sum_{i \in V_l, j \in V_r} u_{ij} = y_{lr} \quad \forall l, r = 1, \dots, K, l \neq r \quad (2.68)$$

$$\sum_{j \in V_r} u_{ij} \leq z_i \quad \forall r = 1, \dots, K, \forall i \in V \setminus V_r \quad (2.69)$$

$$\sum_{(i,j) \in E} u_{ij} = K - 1 \quad (2.70)$$

$$y_{ij} = w_{kij} + w_{kji} \quad \forall k, i, j = 1, \dots, K, i \neq j \quad (2.71)$$

$$\sum_j w_{kij} = 1 \quad \forall k, i = 1, \dots, K, i \neq k \quad (2.72)$$

$$w_{kkj} = 0 \quad \forall k, j = 1, \dots, K \quad (2.73)$$

$$w_{kij} \geq 0 \quad \forall k, i, j = 1, \dots, K \quad (2.74)$$

$$u_{ij} \geq 0 \quad \forall (i, j) \in E \quad (2.75)$$

$$y_{lr} \in B^1 \quad \forall l, r = 1, \dots, K \quad (2.76)$$

$$z_i \geq 0 \quad \forall i \in V \quad (2.77)$$

The *subel* formulation uses three types of variables to define the connections between clusters in the graph. The first group of variables, which we have already used in the previous formulations, are indicator edge variables,  $u_{ij}$ . These variables define the use of specific edges between nodes in the graph. The other two groups of variables,  $y_{lr}$  and  $w_{klr}$ , on the other hand, define connections between **clusters** in the graph. The  $y_{lr}$  variables indicate whether the solutions includes **any** of the edges directly connecting clusters  $l$  and  $r$ . The  $w_{klr}$  variables are used to define  $K$  **directed** trees, one for each of the clusters in the graph. Each variable  $w_{klr}$  indicates whether cluster  $r$  is a predecessor of cluster  $l$  in the directed tree rooted at cluster  $k$ . Specifically:

$u_{ij}$  - indicator variable equal to 1 if **edge**  $(i, j)$  is used in the final solution

$y_{lr}$  - indicator variable equal to 1 if clusters  $V_l$  and  $V_r$  are directly connected in the final solution

$w_{klr}$  - indicator variable equal to 1 if cluster  $r$  is a predecessor of cluster  $l$  in the directed tree rooted at cluster  $k$ .

Since the  $u_{ij}$  variables provide more specific information on connections within the graph, these variables can be referred to as **local edge variables** [46]. Similarly, the variables  $y_{lr}$  can be referred to as **global edge variables** due to the less specific information provided by these variables [46].

The three groups of variables are used to model the GMST problem in the following way. Constraint (2.67), as before, ensures that exactly one node is selected from each cluster. Constraint (2.68) ensures that an edge connecting two nodes can be used only if the clusters corresponding to these nodes are connected by a global edge. Constraint (2.64) guarantees that node  $i$  belonging to cluster  $V_l$  can be connected to at most one node in any given cluster  $V_r$  ( $l \neq r$ ). Constraint (2.70) specifies the number of edges that are supposed to be selected for the GMST. Constraint (2.71) ensures that clusters  $i$  and  $j$  can be adjacent in the directed tree rooted at cluster  $k$  only if clusters  $i$  and  $j$  are directly connected. Constraints (2.72) and (2.73) ensure that, with the exception of a root cluster  $k$  ( $k \in K$ ), every cluster in the directed tree rooted in the cluster  $k$  has exactly one predecessor cluster.

The *subel* formulation has several nice properties that can be used for the development of simple exact solution procedures. To get a better insight into the nature of this formulation, it is best to first look at the well-known subtour elimination formulation for the minimum spanning tree (MST) problem from which this formulation was derived. This (*submst*) formulation can be stated as follows.

$$\text{Min} \quad \sum_{(i,j) \in E} c_{ij} u_{ij} \quad (2.78)$$

$$\text{Subject to:} \quad \sum_{\forall (i,j) \in E} u_{ij} = n - 1 \quad (2.79)$$

$$\sum_{\forall (i,j) \in S} u_{ij} \leq |S| - 1 \quad S \subset V, |S| > 1 \quad (2.80)$$

$$u_{ij} \in B^1 \quad \forall (i,j) \in E \quad (2.81)$$

The linear relaxation of this formulation describes the convex hull of the integer feasible region [40], but has an exponential number of subtour elimination constraints (2.80). However, this formulation can be solved polynomially using a cutting plane algorithm that would allow us to start with a small number of subtour elimination constraints (SECs) and add other SECs only when needed. It is well known that the separation problem for the identification of violated SECs is a polynomially solvable min-cut problem that could be solved either using one of the well-known combinatorial algorithms for this problem or a linear program. It turns out that the latter fact is very important,



since it allows a simple and efficient modification of the original formulation.

Martin [41] shows that when the separation problem can be formulated as a polynomial size linear program, we can use the dual of the separation problem to obtain a more efficient version of the original mathematical formulation. The main advantage of this approach is that it can provide formulations valid for the original problem, but with a polynomial number of variables **and** constraints. In the case of the *submst* formulation, use of this approach leads to new formulation (*ref-submst*) that replaces constraint (2.80) by the following three constraints.

$$w_{kij} + w_{kji} = u_{ij} \quad \forall (i, j) \in E, \forall k \in V \quad (2.82)$$

$$\sum_j w_{kij} \leq 1 \quad \forall k \in V, \forall i \in V \setminus k \quad (2.83)$$

$$\sum_j w_{kkj} = 0 \quad \forall k \in V \quad (2.84)$$

The formal proof of equivalence of the formulations *submst* and *ref-submst* is given in [41], but it is easy to see that the *ref-submst* formulation is valid. In terms of the MST problem, each set of  $w_{kij}$  variables for a given  $k$  represents a set of variables defining a directed tree rooted at node  $k$ , and where individual variables  $w_{kij}$  are equal to 1 if node  $j$  is predecessor of node  $i$  in the directed tree rooted at node  $k$ . Constraints (2.83) and (2.84), together with (2.79) and (2.82), guarantee that each node in the directed tree rooted at node  $k$  has *exactly* one predecessor node, while the node  $k$  has no predecessors at all. Constraints (2.82) through (2.84) guarantee that there are no cycles, since if there was a cycle it would mean that for some root node  $k$ , and the corresponding set of variables  $w_{kij}$ , there is a node with either two predecessors, or a root node has a predecessor, which is not valid by the definition of these constraints.

We now discuss the properties of a special type of relaxation of the *ref-submst* formulation. In this relaxation, integrality constraints (2.81) are relaxed. Additionally, only a subset of nodes  $W$  ( $W \subset V$ ) is selected to be used as root nodes of directed trees. So, instead of having variables  $w_{kij}$  defined for each  $k \in V$ , only the  $w_{kij}$  variables for  $k \in W$  are included in the formulation. We will refer to this type of relaxation of the *ref-submst* formulation as a *p-root* relaxation, where  $p$  indicates the number of nodes used as roots in a given relaxation.

**Lemma 2.1** *The linear relaxation of the ref-submst formulation with only a subset of  $w_{kij}$  variables, such that  $k \in W \subset V$  ( $1 \leq |W| \leq |V| - 1$ ), provides optimal solution for a given MST problem if the solution is integer feasible to the ref-submst.*

**Proof:** It is easy to see that Lemma 2.1 is true, as any relaxation of the *ref-submst* not including a set of constraints (2.82) - (2.84) is actually equivalent to the *submst* formulation without a set of subtour elimination constraints. And, for the latter relaxation, an integer solution without cycles represents the optimal solution.  $\square$

Lemma 2.1 implies that an integer solution of the *p-root* relaxation of the *ref-submst* formulation always represents either a tree, or a structure with at least one cycle (note that if the solution is integer but does not define a tree structure, then, by constraints (2.80), the solution must contain at least one cycle). So, one way to solve the original problem is to start with a *p-root* relaxation, and then add violated constraints (2.82) - (2.84). Another approach to solve the original problem is to start with a *p-root* relaxation, and then use violated subtour elimination constraints (2.80) to eliminate any cycles that may be present in the solution of the *p-root* relaxation. Note that the use of either of these approaches requires solving of a separation problem. While the violated SECs can be identified using polynomially solvable min-cut algorithm, identification of the violated constraints (2.82)-(2.84) is not so obvious. However, we show that when the solution of the *p-root* relaxation is integral, then with the use of the following proposition, the separation problem for constraints (2.82)-(2.84) can be replaced by a problem that can be solved in a straightforward manner.

**Proposition 2.1** *If there is a cycle in the integer solution of the p-root relaxation of the ref-submst formulation, then nodes  $k^* \in W$  (where  $W$  is the set of root nodes) can never be a part of the cycle(s) present.*

**Proof:** Observe that if a node  $k^* \in W$  was a part of a cycle, it would imply that either one of the nodes in the cycle (other than node  $k^*$ ) has 2 predecessors in the directed tree rooted at node  $k^*$ , or that node  $k^*$  has a predecessor in this tree. However, neither is possible by definition of constraints (2.82) - (2.84) for the set of variables  $w_{k^*ij}$ .  $\square$

Proposition 2.1 suggests that violated constraints (2.82) - (2.84) can always be identified as those corresponding to the directed tree rooted at nodes forming a cycle in the current solution of the  $p$ -root relaxation. We will now show how these ideas can be applied to the GMST problem.

Formulation *subel* represents a straightforward extension of the *ref-submst* formulation, achieved through the introduction of local and global variables. Pop [46] has shown that we can define a  $p$ -root relaxation of the *subel* formulation using a similar relaxation strategy defined for the *ref-submst* formulation (except that now we do not relax integrality constraint (2.76) for the  $y_{lr}$ . In other words, the  $p$ -root relaxation of the *subel* formulation includes only a subset of  $w_{kij}$  variables that define directed trees for the pre-specified set of the root clusters  $W$  ( $1 \leq |W| \leq K - 1$ ). (Note that the directed trees in the  $p$ -root relaxation for the GMST problem are defined over the clusters of the graph  $G$ , while the directed trees in the  $p$ -root relaxation for the MST problem are defined over the nodes of the graph  $G$ .) We will show next that this relaxation has similar properties as the  $p$ -root relaxation for the MST problem.

**Lemma 2.2** *The  $p$ -root relaxation of the *subel* formulation with only a subset of  $w_{kij}$  variables, such that  $k \in W \subset K$  ( $1 \leq |W| \leq |K - 1|$ ), provides an optimal solution to the GMST problem if the solution is integer feasible to the *subel* formulation.*

**Proof:** First, observe that we can write the *subel* formulation using the subtour elimination constraints instead of the constraints (2.71) - (2.74). (We will refer to this new formulation as the *subel'* formulation for the GMST problem.) This directly implies that any relaxation of the *subel* formulation that does not include a set of constraints (2.71) - (2.74) is equivalent to the *subel'* formulation without a set of subtour elimination constraints. And, for the latter relaxation, an integer solution without cycles represents the optimal solution. This means that an optimal solution of the  $p$ -root relaxation of the *subel* formulation that is integer feasible to the *subel* formulation, is also an optimal solution for the *subel* formulation.  $\square$

As in the MST problem, we can similarly argue that if an integer solution of the  $p$ -root relaxation is infeasible to the *subel* formulation, then there must exist a global cycle defined by the  $y_{lr}$  variables. Further, by definition of constraints (2.71) - (2.74), clusters that are used as roots

of directed trees in the *p-root* relaxation of the *subel* formulation cannot belong to global cycles.

In the next two chapters we will present several heuristic and exact procedures for the PCGMST problem. Chapter 3 investigates several polynomial time heuristics and two metaheuristic procedures—local search and genetic algorithm. In Chapter 4, we will use the *subel* formulation to develop two exact algorithms for the PCGMST problem.

## Chapter 3

### Heuristic Procedures for the Prize-Collecting Generalized Minimum Spanning Tree Problem

Given the fact that the generalized minimum spanning tree problem is a generalization of the polynomially solvable minimum spanning tree problem, it is natural to raise a question regarding the applicability of existing MST algorithms to the PCGMST problem. This question is studied in the first part of this chapter. In the second part, two metaheuristic procedures, local search and genetic algorithm, are developed for the PCGMST problem. Note that although the algorithms developed in this and the following chapter are referred to as algorithms for the PCGMST problem, they can also be directly applied to the GMST problem by setting the node weights to zero.

#### 3.1 Simple Lower and Upper Bound Algorithms

The idea of using modified versions of the well-known MST algorithms for the PCGMST problem is not new. Dror et al. [17], for example, looked at this problem in the framework of the *at-least* version of the GMST problem. Feremans [24] has also used adaptations of the MST algorithms to generate upper bounds in an exact branch-and-cut algorithm for the GMST problem, but did not look at individual upper bounds that can be obtained using these adaptations independently. The adaptation of MST algorithms to the PCGMST problem sometimes require specific choices that may not be immediately obvious, and affect the quality of the solution. These issues are discussed and computationally tested for the three well-known MST algorithms next.

##### 3.1.1 Lower Bound Algorithm

A very simple lower bound for a given PCGMST problem can be obtained through the straightforward application of Kruskal's algorithm for the MST. This procedure solves the MST problem on a modified graph where each cluster is contracted into a single node (the prize of this

node is set to the highest prize offered by nodes from the same cluster) and multiple edges between pairs of clusters are replaced by the one of minimum cost. To calculate this lower bound, it is not necessary to contract the graph. Instead, it is fairly easy to modify Kruskal's algorithm to accomplish the same. The steps of this procedure are as follows.

**Lower Bound Algorithm for the PCGMST:**

Step 1: Start with a completely disconnected graph  $T$  with  $K$  clusters defined.

Step 2: Order the edges of the initial graph  $G$  in ascending order of their cost.

Step 3: Starting from the beginning of the list, add edges to  $T$  provided that this addition does not create a cycle among the clusters. If  $K - 1$  edges have been added, modify the objective value of the GST found by subtracting the sum (over all clusters) of the maximum prizes offered by each cluster.

Step 4: Repeat Step 3 until  $K - 1$  edges have been added.

The running time of the lower bound algorithm is identical to Kruskal's and is  $\mathcal{O}(|E| + |V| \log |V|)$ .

### 3.1.2 Upper Bound Algorithms

As already mentioned, one could adapt well known MST algorithms to develop upper bound procedures for the PCGMST problem. We discuss these adaptations separately for Kruskal's, Prim's, and Sollin's algorithms, as these adaptations differ significantly.

The adaptation of Kruskal's algorithm builds the PCGMST in a similar fashion to Kruskal's algorithm for the MST, with two exceptions. First, we need to make sure that exactly one node in each cluster is selected. And, second, we need to take into account prizes offered by the selected nodes. The precise steps of this procedure are as follows:

**Kruskal's Heuristic for the PCGMST:**

Step 1: Start with a completely disconnected graph  $T$  with  $K$  clusters defined.

Step 2: Modify the cost of every edge  $(i, j)$  in the initial graph  $G$  by subtracting the prize of the nodes  $i$  and  $j$  incident to this edge (i.e.  $\widetilde{c}_{ij} = c_{ij} - p_i - p_j$ ).

Step 3: Find the minimum cost edge such that its addition to  $T$  does not create a cycle among the clusters, and that at most one node from each cluster is selected for  $T$ . Add this edge to  $T$ .

Step 4: If one, or both end nodes of the last edge added to  $T$  was not already in the solution, update the cost of all edges incident to the end node(s) that is not already in the solution. The new cost for these edges is modified by adding the prize of the node(s) not already in the solution. (In other words if edge  $(i, j)$  is added and node  $j$  was in the solution, and node  $i$  was not in the solution, update  $\widetilde{c}_{ik} = \widetilde{c}_{ik} + p_i$  for all nodes incident to node  $i$ .)

Step 5: Repeat Steps 3 and 4 until  $K - 1$  edges have been added.

The bottleneck step in Kruskal's adaptation is Step 3, which takes  $\mathcal{O}(|E|)$  time. Since we add  $K - 1$  edges, the running time of Kruskal's adaptation is  $\mathcal{O}(|E|K)$ .

The adaptation of Prim's algorithm requires selection of a starting node from which the tree is grown. Once the starting node is selected, we proceed in a straightforward manner identical to Prim's algorithm for the MST problem while taking care that exactly one node is selected from each cluster. That is, in each step we add the minimum cost edge from the nodes in the tree to the nodes not in the tree, taking care that exactly one node is selected from each cluster. Additionally, we need to make sure that the node prizes are taken into account. Observe that the solution provided by Prim's adaptation may depend on the node selected as the starting node (we refer to this node as root node). In our implementation of this algorithm we select the root node randomly. The precise steps of this adaptation are as follows:

**Prim's Heuristic for the PCGMST:**

Step 1: Start with a completely disconnected graph  $T$  with  $K$  clusters defined.

Step 2: Modify the cost of every edge  $(i, j)$  in the initial graph  $G$  by subtracting the prize of the nodes  $i$  and  $j$  incident to this edge.

Step 3: Randomly select the root node and add it to  $T$ .

Step 4: Find the minimum cost edge with one end point in  $T$ , and such that the other end point is the first node from its cluster to be added to  $T$ .

Step 5: If one, or both end nodes of the last edge added to  $T$  are not already in the solution, update the cost of all edges incident to the end node(s) that is not already in the solution. The new cost for these edges is modified by adding the prize of the node(s) not already in the solution. (Note that in Prim's adaptation both end nodes are not in the solution only for the first edge added. For all subsequent edges, exactly one end node is always in the solution, while the other end node is not in the solution.)

Step 6: Repeat Steps 4 and 5 until  $K - 1$  edges have been added.

The running time of Prim's adaptation is identical to Prim's algorithm for the MST plus the time needed to make updates of edge costs. Since we will update edge cost only once, this takes  $\mathcal{O}(|E|)$  time. So, the running time of Prim's adaptation remains  $\mathcal{O}(|E| + |V| \log |V|)$ . (Note that we actually don't need to explicitly update costs. This can be achieved by changing the labelling step as  $d(j) = \text{Min}\{d(j), c_{ij} - p_j\}$ .)

The adaptation of Sollin's algorithm for the GMST problem requires a bit more care than the adaptations of Prim's and Kruskal's algorithms. This is due to the fact that in one iteration of Sollin's algorithm for the MST problem many edges may be added at the same time, while taking care that no cycles are formed while adding these edges. In the PCGMST problem we additionally need to make sure that exactly one node is selected from each cluster and that node prizes are taken into account. So, the initial selection of edges performed at each step of Sollin's algorithm involves selection of a single least cost edge (taking into account the prizes) emanating from each subtree being built, such that the other end of this edge belongs to a new cluster. The edges are then added one by one in the non-descending order of their cost, taking care that no cycles are formed, and only one node from each cluster is selected, and the node prizes are taken into account. The precise steps of this procedure are as follows:



**Sollin's Heuristic for the PCGMST:**

Step 1: Start with the  $K$  completely disconnected graphs  $T_k$ . Define set of clusters  $S = \{1, \dots, K\}$  (i.e., the set  $S$  initially contains all clusters).

Step 2: Modify cost of every edge  $(i, j)$  in the initial graph  $G$  by subtracting the prize of the nodes  $i$  and  $j$  incident to this edge.

Step 3: For each graph  $T_k$  ( $k \in S$ ), select a minimum cost edge emanating from that graph (while taking care that selected edges do not create cycles and exactly one nodes is selected from each cluster). Order the selected edges in ascending order of their cost.

Step 4: Starting from the beginning of the list, add edges to corresponding graphs  $T_k$ , provided that this addition does not create a cycle among the clusters, and the new node added to  $T_k$  is the first node selected from its own cluster. Merge graphs  $T_i$  and  $T_j$  connected by new edge  $(i, j)$  into new graph  $T_m = T_i \cup T_j$ . Set  $m = i$ ,  $S = S \setminus \{j\}$ .

Step 5: If one, or both end nodes of the last edge added are not already in the solution, update the cost of all edges incident to the end node(s) that is not already in the solution. The new cost for these edges is modified by adding the prize of the node(s) not already in the solution.

Step 6: Repeat Steps 4 and 5 until all the edges selected in Step 3 have been examined.

Step 7: Repeat Steps 3 through 6 until  $K - 1$  edges have been added.

Each iteration of this adaptation takes  $\mathcal{O}(|E| + K^2 \log K)$  time as we go through the list of  $|E|$  edges to select the minimum-cost feasible edge out of each tree (or cluster), sort them, and then consider at most  $K - 1$  edges to add in an iteration. In each iteration at least one edge is added, since it is always feasible to add the first selected edge. Thus there are at most  $K - 1$  iterations. Consequently, the overall running time is  $\mathcal{O}(|E|K + K^2 \log K)$ . The time for making edge updates is  $\mathcal{O}(|E|)$ . Consequently, the overall running time of Sollin's adaptation is  $\mathcal{O}(|E|K + K^2 \log K)$ .

Note that the proposed heuristics are guaranteed to return a feasible solution only when the underlying graph is complete (in terms of edges between all pairs of nodes). Ideally, we would like

to add an edge in our adaptations only if the addition of the edge does not cause the heuristic to fail (i.e., make the problem infeasible). In order to do this, we need to answer the following question. Given a graph, clusters, and edges, does it contain a feasible GST? (When we add an edge, we have selected a node in a cluster. Thus, we can delete all other nodes in the cluster and edges emanating from them, and ask the question: does the modified graph contain a feasible GST?) Unfortunately, in general, from the transformation given by Myung et al. [45], even the recognition of whether a graph contains a generalized spanning tree is  $\mathcal{NP}$ -complete. Consequently, we handle infeasibility as follows.

In the case of Kruskal’s adaptation, if the algorithm results in an infeasible solution, we propose that the algorithm remove the most expensive edge added to the tree from the problem and run the heuristic again. We repeat this procedure until either a feasible GST is obtained or one cluster has no edges out of it. In the case of Prim’s and Sollin’s adaptations, if the algorithms result in an infeasible solution, we delete the edge that was most recently added to the tree, and run the algorithm again.

When feasibility is an issue, a crude running-time bound for Kruskal’s adaptation is  $\mathcal{O}(|E|^2K)$  (since we run Kruskal’s at most  $|E|$  times). However, this bound is somewhat misleading, because when the graph is dense (i.e.,  $|E|$  is large), infeasibility is very unlikely. Furthermore, observe that in Kruskal’s algorithm the most expensive edge in the tree constructed is the last edge that was added to it. Thus, instead of building a tree from scratch, we may simply delete the last edge added and continue to build the tree from there. Similarly, arguing as for Kruskal’s adaptation, a crude running-time bound for Prim’s adaptation, when feasibility is an issue, is  $\mathcal{O}(|E|^2 + |E||V| \log |V|)$ . Again this is somewhat misleading for the very same reason as in the case of Kruskal’s adaptation. Also, as in the case for Kruskal’s adaptation, we can delete the most recently added edge and continue to rebuild a tree from there. Using the same argument, in this situation, a crude running-time bound for Sollin’s adaptation is  $\mathcal{O}(|E|^2K + |E|K^2 \log K)$ .

Our computational results on TSPLIB instances for which the optimal solution is known (these instances are described in section 3.2.) are shown in Table 3.1 and Table 3.2. Table 3.1 shows

Clustering Type	Number Instances	Lower Bound Procedure	Upper Bound Procedures		
			Kruskal's	Prim's	Sollin's
			Avg error	Avg error	Avg error
center	37	37.13%	8.27%	13.10%	7.99%
grid, $\mu = 3$	28	22.22%	5.05%	7.94%	5.22%
grid, $\mu = 5$	28	32.95%	10.65%	12.41%	10.70%
grid, $\mu = 7$	28	42.34%	15.87%	17.18%	15.64%
grid, $\mu = 10$	29	44.39%	15.90%	18.56%	15.67%
Overall	150	35.94%	11.00%	13.83%	10.89%

Table 3.1: Comparison of Straightforward Adaptations of Kruskal's, Prim's, and Sollin's Upper Bound and the Lower Bound Procedure for the PCGMST problem with node weights set to zero.

Clustering Type	Number Instances	Lower Bound Procedure	Upper Bound Procedures		
			Kruskal's	Prim's	Sollin's
			Avg error	Avg error	Avg error
center	33	74.18%	12.06%	25.42%	17.64%
grid, $\mu = 3$	22	47.84%	7.95%	13.71%	10.89%
grid, $\mu = 5$	22	68.68%	12.59%	22.16%	14.21%
grid, $\mu = 7$	26	101.10%	21.33%	32.12%	23.28%
grid, $\mu = 10$	27	74.39%	18.98%	23.76%	19.93%
Overall	130	74.22%	14.75%	23.88%	17.52%

Table 3.2: Comparison of Straightforward Adaptations of Kruskal's, Prim's, and Sollin's Upper Bound and the Lower Bound Procedure for the PCGMST problem with the integer node weights randomly selected in the range  $[0, 10]$ .

results for the PCGMST problem with node weights set to zero (so, these problems correspond to the GMST problem), and Table 3.2 shows results for the PCGMST problem with the integer node weights randomly selected from the interval  $[0, 10]$ . The results in both tables indicate a significant gap between the upper bound of the proposed heuristics and the actual optimal solution. The results for the lower bound procedure described in Section 3.1.1 indicate an even greater gap between the lower bound and the optimal solution, meaning that this procedure is probably not a good choice for obtaining lower bounds for the PCGMST problem. The results in Table 3.2 also show an interesting pattern with respect to the quality of results provided by the upper bound procedures. We can see that the adaptation of the Kruskal's algorithm performs much better than adaptations of Prim's and Sollin's algorithms. This differs from the results in Table 3.1 where all three procedures have comparable performance. A possible reason is that, when some of the nodes have non-zero weights, Kruskal's adaptation is somewhat less constrained in the selection of edges added to the tree, than the other two algorithms. That is, both Prim's and Sollin's adaptations

require that edges added to the tree are incident to subtree(s) that are being built. Kruskal’s adaptation, on the other hand, only makes sure that no cycles are created and that exactly one node is selected from each cluster. Other than that, in the Kruskal’s adaptation, edges are selected in a greedy fashion allowing selection of nodes that offer high prizes.

The results in Table 3.2 additionally indicate that the quality of upper bounds does not necessarily deteriorate with an increase in the average number of nodes per cluster (parameter  $\mu$  can be thought of as the average number of nodes per cluster). This result also differs from our findings for the case with zero node weights, where (on average) the quality of upper bounds strictly deteriorates with an increase in the average number of nodes per cluster.

The poor performance of the proposed upper bound heuristics is not surprising, given the fact that edge selection is used to direct the search. This kind of search can be inefficient for the PCGMST problem since when we select an edge, we automatically select (“fix”) nodes that will be used for the two clusters incident to the edge selected. This motivates the question of whether a different type of search that would take into account node selection, would perform better. For example, we could repeat each of the proposed algorithms  $|V|$  times, starting each time with one of the nodes fixed for the design. The results for our upper bound heuristics improved by running the proposed algorithms  $|V|$  times (we refer to these adaptations as *improved adaptations*), are shown in Tables 3.3 and 3.4. The average CPU time for IAK, IAP, and IAS in these instances was 0.06, 0.01, and 8.24 seconds respectively in the case of zero node weights, and 5.85, 0.01, and 6.55 seconds in the case of non-zero weights. We can see that this simple modification significantly improves these upper bounds. Notice that the running times of the improved adaptations of the MST heuristics are still polynomial as the original adaptations are simply repeated  $|V|$  times.

We describe now an extensive search strategy that significantly improves our upper bound heuristics by progressively fixing *all* the nodes in the generalized spanning tree (GST). First, let  $T$  be a set of nodes that are fixed for the initial GST design, such that  $1 \leq |T| \leq K$ . Next, let  $S_T^t$  represent a solution obtained by fixing *all* the nodes from the set  $T$  ( $t = |T|$ ) for the GST design and then applying one of the three upper bound heuristics. (The steps for finding a solution  $S_T^t$

Clustering Type	Number Instances	IAK	IAP	IAS
		Average error	Average error	Average error
center	37	4.80%	6.81%	4.82%
grid, $\mu = 3$	28	3.42%	5.06%	3.60%
grid, $\mu = 5$	28	6.48%	6.45%	6.59%
grid, $\mu = 7$	28	8.98%	7.43%	8.67%
grid, $\mu = 10$	29	8.43%	6.76%	8.07%
Overall	150	6.34%	6.52%	6.27%

Table 3.3: Comparison of Improved Adaptations of Kruskal’s (IAK), Prim’s (IAP) and Sollin’s (IAS) with node weights set to zero.

Clustering Type	Number Instances	IAK	IAP	IAS
		Average error	Average error	Average error
center	33	5.50%	12.80%	10.30%
grid, $\mu = 3$	22	5.47%	8.55%	7.87%
grid, $\mu = 5$	22	7.03%	11.81%	7.73%
grid, $\mu = 7$	26	12.08%	15.89%	13.04%
grid, $\mu = 10$	27	9.46%	9.43%	9.19%
Overall	130	7.89%	11.83%	9.77%

Table 3.4: Comparison of Improved Adaptations of Kruskal’s (IAK), Prim’s (IAP) and Sollin’s (IAS) with the integer node weights randomly selected in the range  $[0, 10]$ .

for a given set  $T$  is specific to each of our upper bound heuristics, and will be explained later in this section.) Also, let  $T^*$  indicate a set of nodes selected for the *final* GST design, and  $V^*$  indicate a set of all nodes that belong to the same clusters as the nodes in  $T^*$ . The search is started with empty sets  $T$  and  $T^*$ , and one of our upper bound heuristics is run  $|V|$  times, each time starting with a different node selected for the design. The node  $i^*$  ( $i \in V$ ), that provides the minimum cost solution is then fixed for the final GST (i.e., it is added to the set  $T^*$ ). In order to add another node to the set  $T^*$ , we first set  $T = T^*$  and then run the same upper bound heuristic  $|V \setminus V^*|$  times, each time with a different node  $i$  ( $i \in V \setminus V^*$ ) added to the set  $T$  (i.e., each time we set  $T = T^* \cup \{i\}$ , fix the nodes in the set  $T$  for the GST design, and apply the same upper bound heuristic). We continue this procedure until  $K$  nodes are added to the set  $T^*$ . The search paradigm that we just described represents a search strategy called the *pilot method*, which was formally defined by Duin and Voß [18].

An example of the steps of our implementation of the pilot method for the proposed upper bound heuristics is illustrated in Figure 3.1. In this example, we are given an 8-node network

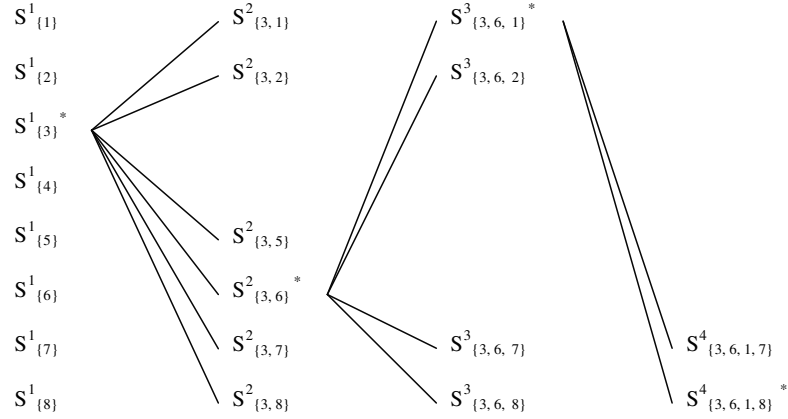


Figure 3.1: Pilot Method for the Upper Bound Heuristics for the PCGMST problem

with 4 clusters, each containing 2 nodes. In the first iteration of the pilot method we run one of our upper bound heuristics  $|V|$  times. The node that provides the best solution (node 3 in this example) is then added to set  $T^*$ , i.e., it is fixed for the final GST design. In the next iteration of the pilot method the same upper bound heuristic is run  $|V \setminus V^*|$  times. As in the previous iteration, we select the node that provides the best solution (in this case node 6) and add it to the set  $T^*$ . The pilot method is continued until  $K$  nodes are fixed for the GST design.

The procedure used to obtain solutions  $S_T^t$  at any given iteration of the pilot method is as follows. In the case of the adaptation of Kruskal's and Sollin's algorithm, we simply fix the nodes from the set  $T$  for the GST design and apply either Kruskal's adaptation or Sollin's adaptation respectively. In other words, we eliminate all nodes belonging to the same clusters as nodes that are fixed for the design and run our upper bound heuristics without any modifications. In the case of Prim's adaptation, we need to implement a slightly different strategy. Here we need to make sure that the new node added to the set  $T^*$  at the end of a given iteration of the pilot method is connected to the tree that is being built. To ensure this, we have modified the first step of the Prim's adaptation as follows. Instead of finding the node that is closest to the current tree (built by the nodes in the set  $T^*$  that are already fixed for the final design), we first connect the candidate node  $i \in V \setminus V^*$  to this tree using the least-cost edge available and then proceed with

Clustering Type	Number Instances	PAK	PAP	PAS
		Average error	Average error	Average error
center	37	0.80%	1.53%	0.83%
grid, $\mu = 3$	28	0.42%	0.83%	0.48%
grid, $\mu = 5$	28	0.71%	1.30%	0.65%
grid, $\mu = 7$	28	1.56%	1.42%	1.74%
grid, $\mu = 10$	29	1.49%	1.42%	1.71%
Overall	150	0.99%	1.31%	1.07%

Table 3.5: Comparison of Pilot Method for Adaptations of Kruskal’s (PAK), Prim’s (PAP) and Sollin’s (PAS) for the PCGMST problem with nodes weights set to zero.

the straightforward Prim’s adaptation.

The results of the pilot method applied to each of our upper bound heuristics are shown in Tables 3.5 and 3.6. We can see that solutions obtained using the pilot method provide upper bounds that are less than 2% from optimality in the case of zero node weights, and less than 3% from optimality in the case of non-zero node weights. This is a significant improvement when compared to the quality of the upper bounds obtained by the initial straightforward adaptation of the MST algorithms. Also, an interesting observation is that the pilot method for our upper bound heuristics runs in polynomial-time (a very crude running time of the pilot method for each of our upper bound heuristics is  $|V|K$  times their running times). The actual average CPU times of the PAK, PAP, and PAS procedures were 0.84, 0.34, and 66.29 second respectively in the case of zero node weights, and 101.22, 0.23, and 49.30 seconds respectively in the case of non-zero weights. (We note here that the significantly longer CPU times of the PAK and PAS procedures compared to the PAP procedure in the case of non-zero node weights are probably due to the codes used for these procedures. We suspect that the running times for these procedures can be improved with more efficient codes.)

### 3.2 Metaheuristic Procedures

In order to explore the possibility of obtaining better results using more general heuristic procedures, we have developed two metaheuristic procedures—local search and genetic algorithm.

The problem state representation for both algorithms is defined as an array of size  $K$ ,

Clustering Type	Number Instances	PAK	PAP	PAS
		Average error	Average error	Average error
center	33	1.75%	1.18%	3.97%
grid, $\mu = 3$	22	1.21%	0.62%	2.30%
grid, $\mu = 5$	22	1.43%	1.85%	1.59%
grid, $\mu = 7$	26	0.91%	1.63%	1.27%
grid, $\mu = 10$	27	1.25%	1.99%	1.70%
Overall	130	1.33%	1.46%	2.27%

Table 3.6: Comparison of Pilot Method for Adaptations of Kruskal’s (PAK), Prim’s (PAP) and Sollin’s (PAS) for the PCGMST problem with the integer node weights randomly selected in the range  $[0, 10]$ .

where each element of an array represents a node selected from a given cluster. An example of a generalized spanning tree and the corresponding problem state representation are shown in Figure 3.2.

### 3.2.1 Local Search

Local search has been successfully used to find near-optimal solutions for a wide variety of combinatorial optimization problems (see Aarts and Lenstra [2] for an up-to-date discussion of local search in combinatorial optimization). In this section, we propose a simple local search heuristic for the PCGMST problem.

Our local search procedure works as follows. The search starts by randomly selecting a feasible GST. It then uses a randomly defined order to visit clusters in a wraparound fashion. At each visit to a given cluster, our local search algorithm performs a 1-opt improvement of a given GST. That is, for each cluster visited, this algorithm finds a node that provides a minimum cost GST, while keeping the remaining part of the GST fixed. Once this node is identified, the current GST is updated and the search moves to the next cluster. Specific steps of our local search algorithm are as follows.

#### **Local Search Heuristic (LS):**

Step 0. Specify the number of feasible solutions to be generated,  $X$ . Repeat Steps 1 through 3 on each of the  $X$  feasible solutions.

Step 1. Randomly select a single node from each cluster. If the subgraph defined by the edges



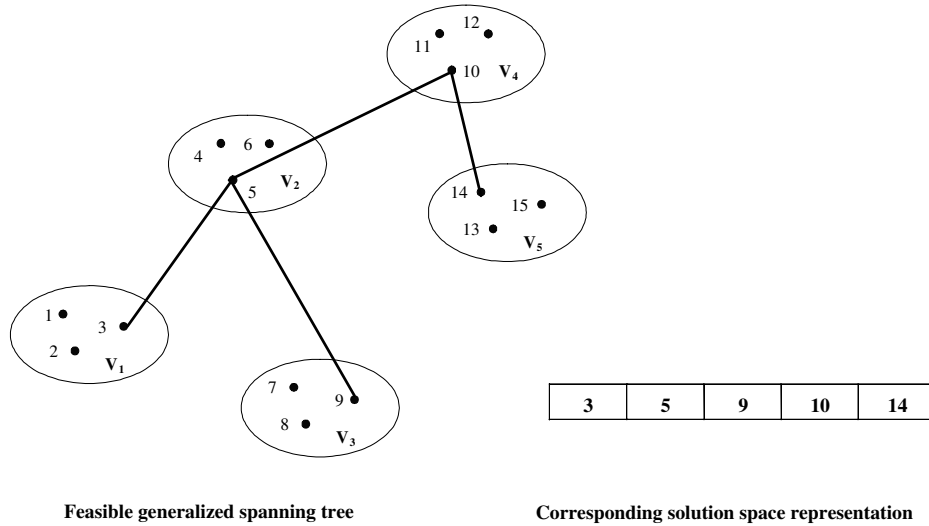


Figure 3.2: An example of a feasible GST and the corresponding problem state representation

between the selected node is connected, apply any one of the minimum spanning tree algorithms to the subgraph defined on the selected nodes in order to build an MST. Otherwise, repeat Step 1.

Step 2. Randomly define an order in which clusters will be searched.

Step 3. Follow the order defined in Step 2 in visiting clusters. Repeat the following steps until  $K$  sequential cluster visits result in no improvement.

- While visiting a cluster, consider each node in the cluster as a replacement for the current node in the cluster contained in the GST. Compute the cost of the solution for each replacement.
- Among the solutions computed in the previous step, identify the one giving the greatest improvement in the objective function (i.e., greatest reduction in solution cost offset by prizes of selected nodes). If there is an improvement, implement it by replacing the node representing the cluster, and update the current tree.

We briefly comment on the complexity of some of the steps of LS. Observe that the subgraph defined on the selected nodes contains  $K$  nodes and can be identified in  $\mathcal{O}(|V| + |E|)$  time. With

this observation, it is easy to show that Step 1 may be accomplished in  $\mathcal{O}(|E| + K^2 + K \log K)$  time if Kruskal's and Prim's are used to construct the spanning tree, and in  $\mathcal{O}(|E| + K^2 \log K)$  time if Sollin's is used to construct the spanning tree. A key part of Step 3 is to compute the cost of a minimum spanning tree when one node from the existing MST is deleted and a new node is added to the graph. Observe that when a node is deleted from an existing spanning tree the resulting forest may contain no edges (it is possible that all the edges in the MST are connected to the node that was deleted). Thus, to find the new MST we need to run an MST algorithm again, and the complexity is  $\mathcal{O}(K^2 + K \log K)$  (assuming Prim's or Kruskal's algorithm is used to generate the tree). If there are  $|V_i|$  nodes in the cluster being visited, then Step 3 takes  $\mathcal{O}(|V_i|(K^2 + K \log K))$  time. We do not comment on the overall worst-case complexity of LS, as, in general, for any neighborhood structure the worst-case complexity analysis for local-search heuristics is bad, while their average-case complexity is good (see Tovey [59]).

Feremans [24] also describes a local search heuristic for the GMST problem that was used in [24] to improve upon feasible solutions found in the proposed branch-and-cut procedure. Adaptation of this local search algorithm to the PCGMST problem works as follows.

**Feremans Local Search Heuristic (FLS):**

- Visit the clusters once in increasing order of cluster number ( $k = 1, \dots, K$ ) and perform the following steps during a visit to a cluster.

Step 0. Consider the set of nodes in the current solution. For this set, delete the node representing the cluster in the current solution, and add all the other nodes in the cluster to it. Create the graph consisting of this set of nodes and all edges between them.

Step 1. Apply the straightforward adaptation of Kruskal's to the graph defined in Step 0.

Step 2. If a lower-cost solution is found in Step 1, replace the current solution by this lower-cost solution.

Notice that LS explicitly tries every node in a cluster as a substitute for the current node

representing the cluster in the solution. On the other hand, FLS considers all the other nodes in the cluster and applies the straightforward adaptation of Kruskal's. However, when Kruskal's is applied, the lowest-cost edge from among the nodes in the cluster will be selected first. Thus, FLS will select the node with the smallest-cost edge (taking into account the prize of the node) out of the cluster to represent the cluster being visited. Thus, it effectively tries only *one other node in the cluster as a substitute* for the current node representing the cluster. Consequently, we expect LS to provide better solutions than FLS. On the other hand, FLS visits the clusters in order once, and performs Kruskal's adaptation at most  $K$  times in Step 1. Consequently, we expect the running time of FLS to be significantly faster than LS. These expectations are confirmed in our computational experiments.

### 3.2.2 Genetic Algorithm

Genetic algorithms are metaheuristics that use ideas from evolution to solve hard combinatorial problems. Although these algorithms have been successfully used to solve many hard problems (see [28] for a recent review of GA applications for network design problems), their specific application is often criticized for the large number of parameters that need to be fine-tuned in order to obtain good performance. The genetic algorithm presented in this section was developed with this issue in mind. In other words, we have tried to design a genetic algorithm that uses as few parameters as possible, but provides high quality of its solutions. We describe this algorithm next.

Our GA uses a randomly generated initial population  $P(0)$  that contains feasible GSTs. Before any genetic operators are applied, local search (in the form of the algorithm described in the previous section) is performed on each of the chromosomes in the initial population. Once the initial generation is generated, new chromosomes are created by applying two genetic operators: mutation and local search enhanced crossover. After a pre-specified number of offspring is created, a fraction of the entire population  $P(t)$  is eliminated, and the remaining fraction is carried to the next generation  $P(t + 1)$ . This process is repeated until a pre-specified number of generations has

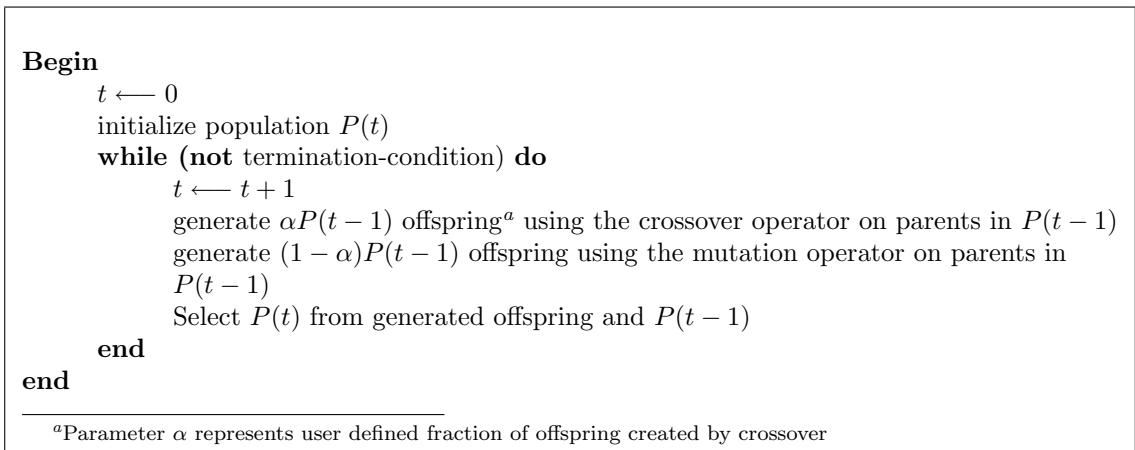


Figure 3.3: Steps of our Genetic Algorithm.

been reached. An outline of these steps is given in the Figure 3.3, while more specific details of our GA are presented next.

**Representation.** As already mentioned, a chromosome is represented as an array of size  $K$ , where the  $i^{th}$  entry (gene value) indicates the node selected to represent cluster  $i$  in the generalized spanning tree. Although there are other possible chromosome representations (such as a binary string of size  $|V|$  indicating nodes in the solution, or a binary array of size  $|E|$  indicating the edges in the solution), we chose to represent chromosomes by array of size  $K$ , because this representation is compact and allows easy application of genetic operators.

**Initial Population.** The initial population is created through random construction of the pre-specified number of feasible solutions. This is achieved by randomly selecting of nodes for each cluster and constructing an MST on the selected nodes. In case this results in an infeasible solution, another set of nodes is randomly selected. Before adding a new chromosome (new solution) to the initial population, we apply our local search algorithm and record the fitness of the resulting chromosome as the cost of the tree (i.e., fitness of the resulting chromosome is defined as the cost of edges in the tree reduced by the sum of prizes corresponding to the nodes in a given chromosome).

**Local Search Enhanced Crossover.** We apply a standard one-point crossover operation as shown in Figure 3.4. As in the initial population, only feasible children solutions are accepted. Basically, two parent chromosomes are randomly selected and a subset of their genes is exchanged

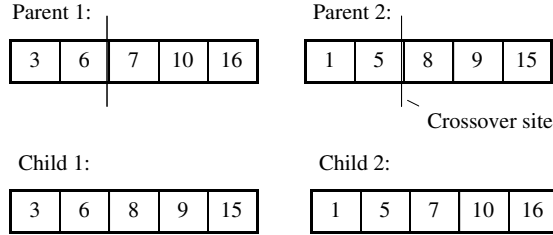


Figure 3.4: One-point crossover operator example.

with respect to the randomly selected crossover site. Each child chromosome created using this operator is used as input to the local search procedure, and the resulting chromosome is added to the population.

**Mutation.** Our mutation operator modifies a single, randomly selected chromosome by randomly selecting one of its clusters and replacing its current nodes by another, randomly selected, node from the same cluster. If the new chromosome is infeasible, it is discarded and the mutation operator is applied again. Although we could have applied local search to the new chromosomes created using the mutation operator, we did not do this, as we wanted to use this operator to maintain the diversity of the population.

**Selection.** Our selection procedure is as follows. At the end of each generation, from the offspring and parents we select a fraction  $\theta$  of the population to survive and constitute the next generation based on the following criteria. We employ elitism to select the top 10% of  $P(t)$ . The remaining 90% of the population to be carried to the next generation is selected based on the ranking-probability distribution function (see Michalewicz [42]). (For example, if 50 individuals will survive to generation  $t$ , 5 are selected by elitism and 45 are selected using the ranking-probability distribution function.) The probability with which each chromosome of rank  $r$  is selected for the next generation is defined as:

$$\text{Prob}\{r\} = q(1 - q)^{r-1} \times \frac{1}{1 - (1 - q)^{(|P(t-1)| + (\alpha + \beta)|P(t-1)|)}},$$

Parameter  $q$  in the above expression is defined by the user and it controls the relative importance of the chromosome rank in selection process (higher rank indicates better objective value). This

parameter takes values between 0 and 1, where higher values indicate more selective pressure.

### 3.2.3 Computational Experiments

We now report on our computational experiments with the LS heuristic and the genetic algorithm (GA) procedure. We coded both LS and GA in Microsoft Visual C++ on a workstation with 2.66 GHz Xeon processor and 2GB RAM. We found that, on small instances, both LS and GA always found the optimal solution. Consequently, we tested our heuristics on two classes of *large* instances—those with edge costs that satisfy the triangle inequality, and those with random edge costs.

The instances where the edge costs satisfy the triangle inequality are from TSPLIB, and are identical to those in Feremans [24] and Feremans et al. [26]. These are actually a subset of the instances described in Fischetti [27]. They contain all problems from TSPLIB 2.1 (see Reinelt [55]) with 48 to 226 nodes (i.e.,  $48 \leq |V| \leq 226$ ) and two 47 node problems, *spain47* and *europa47*, generated by Fischetti [27]. (The naming convention of the TSPLIB problems identifies the number of nodes in the problem at the end of the problem name). Two clustering procedures were used by Fischetti [27] to generate the clusters. For geographical instances, *spain47* (cities in Spain), *europa47* (European cities), *gr96* (African cities), *gr137* (American cities), and *gr202* (European cities), clusters are chosen in a natural way where clusters correspond to a country (or region for *spain47*). For the remaining problems, Fischetti [27] develop two clustering procedures to simulate geographical regions. The first procedure works roughly as follows. It fixes the number of clusters  $K$  to be  $\lceil |V|/5 \rceil$  and then determines  $K$  centers by considering  $K$  nodes that are as far as possible from one another. The clusters are then obtained by assigning nodes to their closest cluster. The second procedure, called *grid clusterization*, works roughly as follows. It constructs an  $H \times H$  square grid, and nodes contained within a box of this square grid are assigned to a cluster. The size of the square grid  $H$  is selected to be the smallest  $H$ , so that the number of nonempty boxes in this  $H \times H$  square grid is  $\geq |V|/\mu$  (where  $\mu$  is a user-defined parameter that may be thought of, roughly, as the average number of nodes in a cluster).

The second set of instances are random instances that we generated using the following two-step procedure, which is similar to the GMST problem-generation technique described in Dror et al. [17]. In the first step, nodes are partitioned into clusters, and in the second step, edges are randomly added to the graph. Node partitioning is performed by generating a random number in the range  $[0, 1]$  for each cluster. The ratio of this random number to the sum of random numbers over all clusters defines the percentage of nodes that are assigned to a given cluster. Edges are then added to the graph using the following approach. In order to insure feasibility of the generated problem, we first create an arbitrary generalized spanning tree, and add the corresponding edges to the graph. The remaining edges are randomly assigned to different pairs of nodes taking care that each node is incident to at least one edge. Each edge is then assigned an integer cost that is randomly generated in the range 1 to 50.

In both sets of instances we have assumed that all nodes offer the same prize, and therefore we have set node prizes to zero. Additional computational experiments for different prize functions are presented in Chapter 4, where we discuss a simple branch-and-cut algorithm to generate exact solutions.

**Parameter Settings for LS and GA:** After initial testing on a separate set of random instances, we selected the following parameters for LS and GA. The GA was used with population size of 100 chromosomes, 15 generations, and 50% of offspring created using crossover and mutation operators each. The fraction  $\theta$  of the population discarded at the end of each generation was set to 0.5, and user-defined parameter  $q$  in the rank-based probability distribution function was set to 0.15. The LS algorithm was performed using 500 independent randomly generated starting solutions.

Table 3.7 summarizes the results of our computational experiments for both sets of instances. In the case of TSPLIB instances, we have evaluated performance of our metaheuristics against 150 instances for which optimal solutions are known, and were determined by Feremans [24]. In the remaining 19 instances, for which the optimal solution is not known, we have compared our solutions against the best known upper and lower bounds (also determined in [24]). In the case of random instances, we were able to determine optimal solutions for 41 out of 42 instances using

Problem Set	Inst	Opt Known	LS		GA	
			Opt	Time (sec)	Opt	Time (sec)
TSPLIB, geog	41	37	36	14.01	37	5.70
TSPLIB, $\mu = 3$	32	28	26	40.60	28	19.46
TSPLIB, $\mu = 5$	32	28	28	24.42	27	7.68
TSPLIB, $\mu = 7$	32	28	27	3.16	28	3.33
TSPLIB, $\mu = 10$	32	29	29	2.17	29	2.78
TSPLIB	169	150	146	16.38	149	7.63
Random	42	41	37	5.54	41	6.64
Summary	211	191	183	14.19	190	7.42

Table 3.7: Summary of computational results for the PCGMST problem with node weights set to zero.

our branch-and-cut algorithm presented in Chapter 4. As the results in Table 3.7 indicate, both metaheuristics have a compelling performance. Out of 191 instances for which the optimal solution is known, our local search algorithm did not find the optimal solution in only 8 instances, while our GA did not find the optimal solution in only 1 instance. It is interesting to note that GA required about half of the CPU time used by LS. In the 19 TSPLIB instances for which the optimal solution is not known, both of our algorithms improved upon the best known upper bound in all instances. The average improvement achieved by LS was 2.42%, and 2.51% for our GA. The average gap of our metaheuristics (over these 19 TSPLIB instances) from the best known lower bound was 12.44% for LS, and 12.35% for GA. However, given the impressive performance of our algorithms over 150 instances for which optimal solution is known, it is reasonable to assume that the lower bounds provided in [24] are weak, and the average gap of our algorithms from the optimal solutions for these instances is significantly lower.

We finally report on some computations we performed to compare LS against FLS. As we indicated in Section 3.2.1, FLS considers only one other node as a substitute for the current node in the cluster, while LS considers all other nodes in the cluster as possible substitutes. We wanted to see to what extent this difference in neighborhood structure affected the quality of the solutions (and the ability of the local search heuristics to find the optimal solution). We coded FLS and ran it on the 211 instances with the identical set of 500 starting solutions for each problem instance as LS. We found that out of the 211 instances, FLS found the optimal solution in only 24 instances, as compared to LS, which found the optimal solution in 179 instances. In 187 instances, the solution



obtained by LS is strictly better than the solution obtained by FLS, while FLS never obtained a better solution than LS. Over the 187 instances for which the optimal solution is known, FLS generated solutions that are, on average, 3.28% from optimality (compared to 0.07% for LS). However, in these instances, FLS required significantly shorter CPU times compared to our LS.

We note that, compared to LS, FLS performs an abbreviated search, visiting each cluster just once. Since we want to compare the difference in neighborhoods, we developed an improved version of FLS, called *complete FLS*, where local search is performed until no further improvement is possible. Complete FLS is identical to LS, except that the neighborhood structure is as in FLS. We coded complete FLS and also ran it on the 211 instances (with the identical set of 500 starting solutions for each problem instance as LS). On examining the results for complete FLS, we found it is better than FLS, but its results fall short of LS. Out of the 211 instances, complete FLS found the optimal solution in 83 instances. Further, in 121 instances, LS is strictly better than complete FLS, while complete FLS is strictly better than LS in two instances. Over the 187 instances for which the optimal solution is known, complete FLS generated solutions that were, on average, 0.55% from optimality. The running time of complete FLS was longer compared to the original FLS, but on average, complete FLS required shorter CPU time compared to our LS.

These results confirm our earlier assertion that LS is expected to provide better results than FLS, while the running time of FLS is expected to be significantly faster than LS. They also show that the neighborhood considered by LS is very effective in converging to the global optimum, while the neighborhood considered by FLS generally gets stuck at a locally optimal solution.

### 3.3 Conclusion

In this chapter, we have presented a simple lower bound algorithm and three upper bound algorithms for the PCGMST problem that represent adaptations of the three well known MST heuristics. We discussed a polynomial search paradigm called the Pilot Method that significantly improves these adaptations. We have also presented two metaheuristics procedures - local search and genetic algorithm. In developing our local search procedure we showed that the neighborhood

structure of our LS procedure is particularly effective, compared to the neighborhood structure of the LS procedure proposed by Feremans [24].

Our computational experiments have shown that, as expected, the straightforward adaptations of the MST heuristics do not provide good bounds for the PCGMST problem. However, when the Pilot Method is applied to these heuristics they provide substantially better results. The results of our computational experiments for the two metaheuristic algorithms presented in this chapter were particularly compelling. They indicate that both LS and GA can be used to rapidly obtain high quality solutions for the PCGMST problem in relatively large networks.

## Chapter 4

### Exact Algorithms for the Prize-Collecting Generalized Minimum Spanning Tree Problem

The two exact procedures described in this chapter are based on the *subel* formulation presented in Chapter 2. In developing these procedures, we make use of the property that the subtour elimination constraints (SECs) in the *subel* formulation are represented by  $K$  directed trees. The first procedure represents an improved version of the Rooting Procedure originally proposed by Pop [46], which solves the PCGMST problem by progressively adding variables and constraints for one directed tree at the time. The second procedure is a new branch-and-cut algorithm that works with SECs defined over connections between clusters. Details of these two procedures and the corresponding computational experiments are presented next.

#### 4.1 Rooting Procedure

The rooting procedure is an exact algorithm for the GMST problem developed by Pop [46]. This procedure is motivated by the fact that the *subel* formulation that includes all the  $w_{kij}$  variables and the corresponding constraints may be too large (and, therefore, too difficult) to solve for large networks. The *p-root* relaxation of the *subel* formulation, on the other hand, includes a smaller number of the variables and constraints and should be easier to solve. The idea behind the rooting procedure, is therefore, to try to find the optimal solution by using only a small subset of clusters as the root clusters. (Recall that in the *p-root* relaxation of the *subel* formulation directed tree variables and corresponding constraints are included for the initially selected set of  $p$  root clusters only. Correspondingly, when we talk about addition of a root cluster, we imply addition of all variables and constraints defining the directed tree rooted at the cluster being added.)

The rooting procedure starts with the *p-root* relaxation of the *subel* formulation with a single randomly selected root cluster  $k_r$ . In the next step, the *p-root* relaxation that includes only

variables and constraints defining a directed tree from root node  $k_r$  is solved. If the solution of this relaxation turns out to be a global tree, then, according to Lemma 2.2, the problem is solved; otherwise, variables and constraints for another root cluster are added (or used to replace existing root clusters) and the relaxation is resolved. This procedure is repeated until the optimal solution is found.

We have implemented a version of the rooting procedure that randomly selects clusters and adds one root cluster at a time to the  $p$ -root relaxation. We refer to this version of the rooting procedure as  $RP_{rand}$ . Although computational results of Pop [46] and Pop et al. [48] indicate that an optimal solution can be found by adding only a few root clusters (the strategy used to add or substitute root clusters is not explained in [48] and [46]), our computational experiments on a set of TSPLIB instances indicate that in many instances, many root clusters need to be included in the  $p$ -root relaxation in order to obtain the optimal solution. We have also found that the  $p$ -root relaxation becomes significantly harder to solve as the number of root clusters grows. Consequently, we propose a more deliberate strategy for addition of the root clusters to the  $p$ -root relaxation that has the potential to provide better results.

We use our results from Chapter 2, where we argued that clusters, which are already added to the  $p$ -root relaxation (i.e., are already root clusters) can never be a part of the global cycle. So, instead of randomly selecting root clusters, we can examine the solution to a given  $p$ -root relaxation for the existence of global cycles and add only those clusters that are part of the identified cycle(s). We refer to this procedure as  $RP_{cycle}$ . In our implementation of  $RP_{cycle}$  we identify one global cycle, and one cluster that is part of the identified cycle and add it to the  $p$ -root relaxation. The steps of  $RP_{cycle}$  are outlined in Figure 4.1. The reason why we might expect this algorithm to outperform the rooting procedure that randomly adds clusters is that  $RP_{cycle}$  targets clusters that are identified as the cause of infeasibility at a given iteration of the rooting procedure. In very recent work that has come to our attention at the end of this research, Pop et al. [47] report computational experiments with the rooting procedure using a search strategy identical to that of our  $RP_{cycle}$ . In this dissertation, however, we look at the performance of two different search

<b>Step 0</b>	Set $W = \emptyset$
<b>Step 1</b>	Randomly select cluster $r \notin W$
<b>Step 2</b>	Set $W = W \cup \{r\}$ Add all global variables and constraints (2.71) - (2.74) for $k = r$ , to $p$ -root relaxation Solve integer $p$ -root relaxation
<b>Step 3</b>	Check if the solution of the $p$ -root relaxation is a GST If GST found Stop; optimal solution is found Else Go to Step 4
<b>Step 4</b>	Identify one cluster in the cycle from the solution of the $p$ -root relaxation Set $r =$ identified cluster from the previous step Go to Step 2.

Figure 4.1: Steps of our implementation of the Rooting Procedure ( $RP_{cycle}$ ).

strategies ( $RP_{rand}$  and  $RP_{cycle}$ ) in order to get a better sense of the impact that the choice of the root clusters has on the efficiency of the rooting procedure. Our computational experiments on a set of geographical TSPLIB instances are provided in Table 4.1. These results confirm our expectations and show a significant difference in the performance of the two versions of the rooting procedure. We can see that, within a two hour CPU time limit,  $RP_{rand}$  did not find the optimal solution in 11 out of 41 instances and, on average, required 2805.16 seconds of CPU time. Also, on average,  $RP_{rand}$  required 7.34 root clusters to be added to the  $p$ -root relaxation in order to find the optimal solution.  $RP_{cycle}$ , on the other hand, did not find the optimal solution in 9 instances, and required, on average, 2126.34 seconds of CPU time. The average number of root clusters needed to be added to the  $p$ -root relaxation in this case was 4.09. Additionally, in instances solved to optimality,  $RP_{cycle}$  never needed more than 8 root clusters to find the optimal solution, while  $RP_{rand}$  required 25 root clusters in the worst case.  $RP_{cycle}$  also provided better lower bounds than  $RP_{rand}$  in several instances. In particular, when compared over 11 instances that  $RP_{rand}$  did not solve to optimality, the lower bound provided by  $RP_{cycle}$  was on average 3.93% better than the one provided by  $RP_{rand}$ .

Problem name	$ E $	$RP_{rand}$			$RP_{cycle}$		
		Solution	Time (sec)	Root Clusters	Solution	Time (sec)	Root Clusters
15spain47	985	2310	11.99	8	2310	1.69	3
27europ47	1042	12971	18.37	4	12971	9.56	3
50gr96	4463	32*	7205.33	15	39	3126.64	7
35gr137	8251	20	237.11	12	20	38.28	5
34gr202	19018	-77	108.83	2	-77	146.84	2
10att48	1010	10861	1.30	4	10861	0.72	3
10gr48	1017	1219	0.63	3	1219	0.72	3
10hk48	995	4082	0.78	2	4082	1.92	4
11eil51	1158	74	1.67	2	74	2.42	3
12brazil58	1464	9159	1.37	4	9159	0.84	3
14st70	2248	157	21.27	6	157	8.27	3
16eil76	2660	85	91.12	5	85	59.30	3
16pr76	2661	46438	90.14	7	46438	19.19	2
20gr96	4292	103	47.80	7	103	14.84	4
20rat99	4609	295	2401.42	8	295	1787.91	6
20kroa100	4727	7874	1081.41	5	7874	820.27	6
20krob100	4716	8018	1595.61	13	8018	370.13	6
20kroc100	4714	7943	285.34	6	7943	152.77	4
20krod100	4716	7545	1006.37	7	7545	857.87	6
20kroe100	4702	8059	246.00	5	8059	98.12	4
20rd100	4703	2690	700.88	11	2690	124.70	4
21eil101	4776	74.17*	7200.81	7	84	5447.50	4
21lin105	5130	6638	130.81	5	6638	67.08	5
22pr107	5441	20271	22.06	7	20271	7.84	4
24gr120	6820	2132	6906.55	9	2132	6678.38	8
25pr124	7315	30065	3281.72	21	30065	129.44	5
26bier127	7191	58014	3915.13	12	58014	366.22	4
28pr136	8879	29819*	7201.52	5	30384.25*	7200.83	4
28gr137	8892	152	3710.14	7	152	999.10	4
29pr144	9952	39919	22.39	9	39919	15.89	7
30kroa150	10809	9013*	7201.95	8	9071.81*	7200.91	4
30krob150	10807	9206.58*	7201.05	4	9285*	7200.94	4
31pr152	11080	38949	4391.59	18	38949	108.47	6
32u159	12031	18181*	7202.02	7	18346*	7200.77	3
39rat195	18478	389*	7202.67	5	404.7*	7200.55	1
40kroa200	19409	10155.39*	7201.13	2	10158.84*	7210.75	4
40krob200	19430	9739*	7201.70	3	9729.81*	7200.87	2
40d198	18841	6674.33*	7201.69	4	6664.75*	7201.18	3
41gr202	19532	-31	1412.89	6	-31	337.84	3
45ts225	24650	54749*	7202.05	1	54749*	7201.54	1
46pr226	24626	55234	4046.97	25	55234	560.75	8

Table 4.1: Computational results for  $RP_{rand}$  and  $RP_{cycle}$  on TSPLIB instances with the center clustering procedure. (Solutions with an asterisk indicate lower bounds for instances where  $RP_{rand}$  and  $RP_{cycle}$  did not find the optimum within a two hour CPU time limit.)

## 4.2 A Simple Branch-and-Cut Algorithm

In Chapter 2 we pointed out that an alternative approach for solving the  $p$ -root relaxation of the *subel* formulation is to progressively add violated subtour elimination constraints (SECs). In this section, we elaborate on this idea and present a new branch-and-cut algorithm for the PCGMST problem.

First, note that in a given solution of the  $p$ -root relaxation violated SECs can be defined over local edge variables and/or over global edge variables (because there are fewer global edge variables than local edge variables). Although it is not apparent which choice is better, it is clear that the number of SECs defined over global edge variables can be significantly lower than the number of SECs over local edge variables. Consequently, we use SECs defined over global edge variables in our branch-and-cut algorithm. Another important decision that one needs to make is whether to add SECs at all nodes of the branch-and-bound tree or only at certain, designated, nodes. This issue turns out to be very important from an implementation perspective since the complexity of the separation problem may not be the same at different nodes of the branch-and-bound tree. In the  $p$ -root relaxation, for example, the separation problem for identification of violated SECs at any node of the branch-and-bound tree is a min-cut problem. At **incumbent** (integer) nodes, on the other hand, we can examine the set of edges selected in the solution and identify any cycles using a simple depth-first search. Once the cycles are identified, we can add the subtour elimination constraints defined over the nodes (or clusters) that are part of each of the identified cycles.

Aside from the advantage of solving a simpler separation problem, addition of cuts at incumbent nodes only may be more effective due to a fewer number of nodes at which the separation problem needs to be solved. We have tested both ideas by defining two branch-and-cut algorithms. The first algorithm,  $BCA_{inc}$ , adds SECs only at incumbent nodes. The steps of this procedure are outlined in the Figure 4.2.

The second algorithm,  $BCA_{all}$ , solves the min-cut problem at every node of the branch-and-bound tree, and whenever violated global SECs are found, two new global SECs are added to

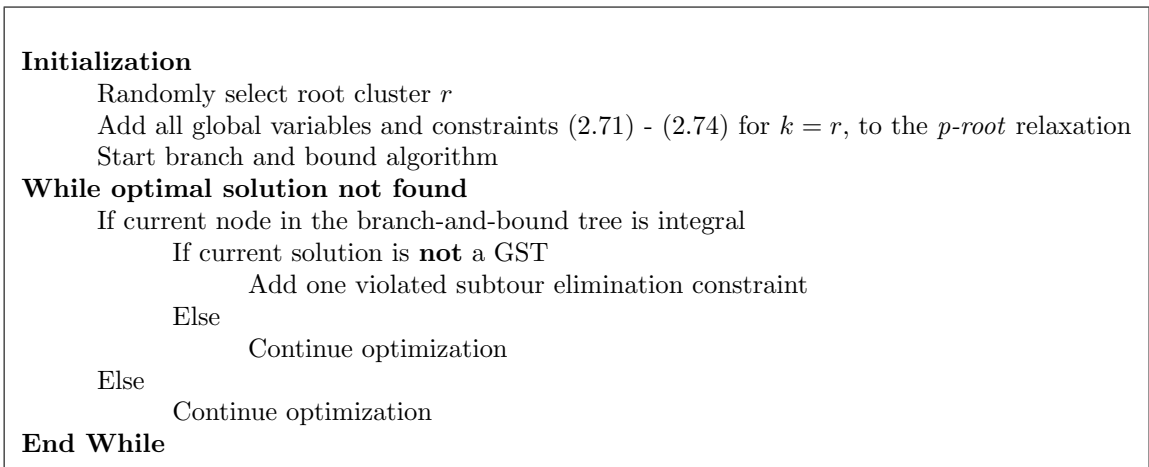


Figure 4.2: Steps of our Branch-and-Cut Algorithm ( $BCA_{inc}$ ).

an entire tree. In our implementation of  $BCA_{all}$  we have used the min-cut algorithm of Store [57] implemented in LEDA 4.5 to solve the separation problem. The comparison of these two procedures is provided in Table 4.2. We can see that the proposed algorithms have comparable performance with  $BCA_{inc}$  providing slightly better upper bounds, but also somewhat worse lower bounds. However, due to simplicity of the implementation of  $BCA_{inc}$ , we suggest its use for problems of moderate size.

We recognize here that more complex branch-and-cut algorithms (such as the one developed for the GMST problem by Feremans [24]) using a larger class of valid inequalities are likely to be more efficient than the exact procedures proposed here, especially in the case of large instances. However, the advantage of the exact procedures described in this chapter, especially  $BCA_{inc}$ , is that they do not require sophisticated separation routines and can be easily coded using commercial optimization software such as ILOG Concert Technology. In particular,  $BCA_{inc}$  can easily be implemented in a modeling language like AMPL. The importance of the simplicity of these procedures is emphasized by the fact that they find the optimal solutions for the PCGMST problem in networks of moderate size within reasonably short computational times. Further, the improvement in the time needed to solve the PCGMST problem using these procedures compared to the time needed to solve some alternative mathematical formulations (without using branch-and-cut)



Clustering Type	Number of Instances	$BCA_{inc}$	$BCA_{all}$	Avg Relative Difference $BCA_{inc}$ vs $BCA_{all}$	
		Avg. Time (sec)	Avg. Time (sec)	UB	LB
center	41	1606.23	1588.04	0.00 %	-0.03 %
grid, $\mu = 3$	32	2362.58	2177.03	-0.07 %	-0.31 %
grid, $\mu = 5$	32	2611.72	2293.01	-0.03 %	-0.09 %
grid, $\mu = 7$	32	1534.83	1450.86	0.01 %	-0.08 %
grid, $\mu = 10$	32	1189.16	1177.85	0.01 %	-0.12 %
Overall	169	1847.34	1729.41	-0.02 %	-0.12 %

Table 4.2: Comparison of Branch-and-Cut algorithms  $BCA_{inc}$  and  $BCA_{all}$  for the PCGMST problem. (Note: The TSPLIB instances used for these tests were modified by adding randomly generated integer node weights in the range  $[0, 10]$ .)

is remarkable. For example, the mathematical formulation that we used for the GMST problem in [30]<sup>1</sup> needed several days to solve several random instances (these instances are described in Chapter 3). Our  $BCA_{inc}$  procedure, on the other hand, solved most of these instances within two hours of CPU time.

### 4.3 Computational Results

The two exact procedures presented in this chapter were coded using Microsoft Visual C++ and Concert Technology 2.0 in CPLEX 9.0. As before, all computations were performed on a workstation with 2.66 GHz Xeon processor and 2GB RAM. The test instances used are the two sets of problems described in Chapter 3. However, we have modified these instances by adding randomly generated integer node weights in range  $[0, 10]$ .

The two metaheuristic procedures were applied with the same parameters as in Chapter 3, while for the exact procedures,  $BCA_{inc}$  and  $RP_{cycle}$ , we have used 2 hour time limit.

We first compared  $BCA_{inc}$  and  $RP_{cycle}$  over a set of geographical TSPLIB instances and obtained the following results. Out of 41 instances,  $BCA_{inc}$  did not find a solution in 8 instances, while  $RP_{cycle}$  did not find a solution in 9 instances. Overall, the average CPU time for  $BCA_{inc}$  was 1606.23 seconds, while  $RP_{cycle}$  required 2126.34 seconds on average. In 9 instances where  $RP_{cycle}$  did not find the optimum,  $BCA_{inc}$  provided a better lower bound in 8 instances with an

<sup>1</sup>This formulation was solved using CPLEX 7.1 on a Sun Microsystems Enterprise 250 with 2x400 MHz processors and 2 GB RAM.

average improvement of 1.72%.

Due to the superiority of  $BCA_{inc}$  over  $RP_{cycle}$ , in our further computational tests, we compared  $BCA_{inc}$  with the two metaheuristics — LS and GA that we presented in Chapter 3. These results are provided in Tables 4.3 through 4.6. The first three tables provide results for 130 TSPLIB instances where  $BCA_{inc}$  found the optimum within two hours of CPU time, and Table 4.6 provides results for the remaining 39 unsolved instances. Both metaheuristic procedures found optimal solutions in all 130 instances. The average CPU time required by LS in these instances was 10.23 seconds, while the average CPU time required by the GA was 4.7 seconds. Over all 169 TSPLIB instances, GA always provided solutions at least as good as the solutions provided by LS, and in eight instances GA was better than LS. In these instances, the upper bound provided by GA was, on average, 0.14% better than the upper bound provided by LS. The average time  $BCA_{inc}$  needed to solve 130 TSPLIB instances to optimality was 241.52 seconds. We found that in 24 of the remaining 39 instances where  $BCA_{inc}$  did not find the optimal solution, the upper bound provided by  $BCA_{inc}$  equals the best upper bound found by our metaheuristics. In the remaining 15 instances, the best upper bound found by our metaheuristics was, on average, 0.31% better than the one provided by  $BCA_{inc}$ . The average gap between the best upper bound and lower bound provided by  $BCA_{inc}$  was 8.94% over all 39 unsolved instances.

In the case of random instances, our branch-and-cut algorithm found the optimal solution for 41 out of 42 instances within two hours of CPU time (see Table 4.7). In instances where the optimal solution is known,  $BCA_{inc}$  needed 32.93 seconds to find the optimal solution. Over the same set of instances, the average CPU time for LS was 6.2 seconds, and 5.98 seconds for GA. Both LS and GA did not find an optimal solution for one of the problems where the optimal solution is known.

In order to test the sensitivity of our procedures with respect to assigned node prizes, we have repeated our experiments over all 169 TSPLIB instances using a different node-prize function with 5 possible prize values (0, 1, 10, 100, 1000). In this case,  $BCA_{inc}$  found the optimal solution in 166 out of 169 instances within two hours of CPU time, with an average CPU time of 113.95

seconds <sup>2</sup>. Both LS and GA found optimal solution in all 166 instances, with average CPU times of 14.37 and 9.56 seconds, respectively. In three instances where  $BCA_{inc}$  did not find the optimal solution, LS and GA provided the same solutions that were 12.65% from the best known lower bound, and 0.03% better than the upper bound provided by  $BCA_{inc}$ . These results indicate that higher node prizes (relative to edge costs) tend to make the problem easier to solve for  $BCA_{inc}$ . At the same time, the higher importance of node prizes seems to make problems slightly more difficult to solve for the metaheuristics in terms of computation time, but become easier for LS with respect to solution quality. The solution quality provided by our GA remained unchanged, as this metaheuristic provided optimal solutions for all TSPLIB problems where the optimal solution is known for both types of node prize functions used. A summary of the results for both prize functions is shown in Table 4.8.

---

<sup>2</sup>In this case, we had to adjust the mip gap in CPLEX settings due to higher magnitude of cost functions.

Problem name	$ E $	Optimal Solution	BCA Time (sec)	LS Time (sec)	GA Time (sec)
15spain47	985	2310	1.27	0.53	0.58
27europ47	1042	12971	66.27	1.27	1.59
50gr96	4463	39	130.13	8.69	10.77
35gr137	8251	20	7.84	5.24	6.42
34gr202	19018	-77	120.21	9.48	14.05
10att48	1010	10861	0.31	0.24	0.27
10gr48	1017	1219	0.44	0.22	0.25
10hk48	995	4082	0.38	0.27	0.28
11eil51	1158	74	1.11	0.28	0.31
12brazil58	1464	9159	0.58	0.41	0.42
14st70	2248	157	3.50	0.63	0.69
16eil76	2660	85	23.94	0.88	1.00
16pr76	2661	46438	19.39	1.02	1.05
20gr96	4292	103	3.73	1.58	1.91
20rat99	4609	295	467.03	1.88	2.00
20kroa100	4727	7874	259.11	2.05	2.17
20krob100	4716	8018	38.22	1.88	2.14
20kroc100	4714	7943	50.09	2.02	2.13
20krod100	4716	7545	331.48	2.13	2.13
20kroe100	4702	8059	75.11	1.92	1.97
20rd100	4703	2690	47.28	1.94	2.06
21eil101	4776	84	2874.41	2.06	2.28
21lin105	5130	6638	44.92	2.27	2.36
22pr107	5441	20271	8.91	2.70	3.02
24gr120	6820	2132	354.80	3.41	3.56
25pr124	7315	30065	50.66	3.33	3.83
26bier127	7191	58014	115.61	4.45	4.39
28gr137	8892	152	435.23	4.83	6.73
29pr144	9952	39919	10.25	5.89	6.28
31pr152	11080	38949	55.28	7.34	12.56
32u159	12031	18563	1990.41	8.72	10.73
41gr202	19532	-31	562.42	14.63	16.75
46pr226	24626	55234	104.83	182.95	23.49

Table 4.3: Computational results for  $BCA_{inc}$ , LS and GA on TSPLIB instances with the center clustering procedure where  $BCA_{inc}$  found the optimal solution within 2 hours of CPU time. Both LS and GA found the optimal solutions in all instances

Problem name	$\mu = 3$					Problem name	$\mu = 5$				
	E	Optimal Solution	BCA Time (sec)	LS Time (sec)	GA Time (sec)		Optimal Solution	BCA Time (sec)	LS Time (sec)	GA Time (sec)	
18att48	1062	16421	2.14	0.89	0.84	13att48	1025	13138	0.50	0.36	0.42
25eil51	1235	114	0.64	1.22	1.56	16eil51	1214	68	0.64	0.53	0.66
24st70	2329	155	2.50	1.81	2.03	16st70	2277	110	1.27	0.77	0.89
36eil76	2789	96	3.36	4.45	4.74	16eil76	2695	58	9.19	0.86	0.97
31pr76	2770	57885	205.66	3.36	3.30	16pr76	2661	29704	34.92	0.94	1.00
33gr96	4413	88	7.78	4.75	4.98	22gr96	4315	105	22.14	2.14	2.36
36rat99	4753	331	110.31	5.67	6.30	25rat99	4692	276	242.22	2.89	3.06
43kroa100	4844	11723	15.55	8.33	9.44	23kroa100	4748	7936	116.17	2.72	2.84
44krob100	4854	12329	117.69	8.94	9.84	25krob100	4743	7753	24.25	3.25	3.24
42kroc100	4847	12081	4.91	12.56	8.78	25kroc100	4762	7965	2428.92	3.24	3.17
42krod100	4846	11597	21.75	12.67	8.91	24krod100	4724	8635	420.38	2.67	2.86
42kroe100	4846	12102	90.50	14.50	9.08	25kroe100	4738	8307	10.34	2.84	3.03
36rd100	4801	3817	21.81	5.78	6.66	24rd100	4742	2950	6.77	2.83	2.94
36eil101	4912	68	45.31	5.74	6.73	25eil101	4812	62	279.70	3.08	4.84
42lm105	5340	9054	5.73	9.64	10.24	30lm105	5237	7265	62.08	5.61	5.13
45pr107	5542	23070	41.56	7.31	8.38	22pr107	5438	19765	7.56	2.53	2.92
42pr124	7440	37623	9.88	86.84	11.34	25pr124	7219	27023	7.81	3.67	3.86
49gr137	9122	89	558.09	137.09	14.61	26bier127	7173	58859	1287.97	4.42	4.11
48pr144	10084	43467	205.84	13.16	26.08	34pr136	8888	37590	4758.61	7.45	7.56
56krob150	10982	13548	1534.97	211.77	22.99	32gr137	8908	149	954.50	5.69	6.55
54pr152	11254	43995	33.86	162.83	20.34	30pr144	9928	36132	23.17	5.72	6.28
68gr202	19826	-115	562.06	135.16	39.02	41gr202	19303	-29	875.55	13.47	15.56

Table 4.4: Computational results for  $BCA_{inc}$ , LS and GA on TSPLIB instances with grid clustering procedure,  $\mu = 3$  and 5, where  $BCA_{inc}$  found the optimal solution within 2 hours of CPU time. Both LA and GA found the optimal solutions in all instances.

Problem name	$\mu = 7$					$\mu = 10$					
	$ E $	Optimal Solution	BCA Time (sec)	LS Time (sec)	GA Time (sec)	Problem name	$ E $	Optimal Solution	BCA Time (sec)	LS Time (sec)	GA Time (sec)
7att48	947	6631	0.20	0.13	0.14	7att48	947	6631	0.20	0.13	0.14
9eil51	1143	42	0.34	0.19	0.23	9eil51	1143	42	0.33	0.19	0.23
16st70	2277	110	1.27	0.77	0.89	9st70	2160	85	0.74	0.27	0.30
16eil76	2695	58	9.20	0.84	0.95	9eil76	2545	33	1.41	0.28	0.33
16pr76	2661	29704	34.91	0.94	1.00	9pr76	2532	20444	0.81	0.31	0.33
15gr96	4170	92	8.33	0.98	1.08	15gr96	4170	92	8.33	0.95	1.08
16rat99	4583	198	78.17	1.31	1.31	16rat99	4583	198	78.66	1.31	1.33
16kroa100	4647	5899	30.00	1.27	1.36	16kroa100	4647	5899	30.05	1.25	1.38
16krob100	4652	5962	62.96	1.36	1.34	16krob100	4652	5962	60.19	1.38	1.36
16kroc100	4651	5463	11.72	1.34	1.31	16kroc100	4651	5463	11.81	1.36	1.31
16krod100	4647	5837	85.66	1.34	1.41	16krod100	4647	5837	85.50	1.36	1.41
16kroe100	4607	6367	5.08	1.27	1.31	16kroe100	4607	6367	5.08	1.28	1.31
16rd100	4606	2219	8.49	1.24	1.31	16rd100	4606	2219	8.49	1.25	1.31
16eil101	4757	36	94.25	1.20	1.31	16eil101	4757	36	94.38	1.20	1.33
16lin105	5005	4466	43.20	1.34	1.36	16lin105	5005	4466	43.08	1.34	1.36
16pr107	5322	17445	2.27	1.42	1.55	12pr107	5196	16680	0.84	0.80	0.84
19pr124	7077	23057	10.70	2.16	4.08	14pr124	6896	18491	28.30	1.19	3.67
19bier127	7039	51974	107.88	2.25	4.19	14bier127	6240	43702	73.72	1.25	3.47
20pr136	8724	22432	1645.95	2.39	2.72	16pr136	8548	21651	132.02	1.47	1.77
22gr137	8746	122	1678.47	2.78	9.36	15gr137	8504	95	64.78	1.41	2.39
21pr144	9762	33847	3.92	3.02	3.16	16pr144	9508	32434	1.84	1.97	1.84
25kroa150	10703	7786	766.13	4.89	5.09	16kroa150	10506	5149	135.03	2.14	2.11
24pr152	11008	35308	34.98	4.52	4.52	16krob150	10455	5390	142.30	2.05	2.03
23u159	11896	12544	581.69	4.66	4.20	16pr152	10828	33249	6.36	2.05	2.16
31gr202	18872	-8	217.19	8.09	19.80	23u159	11896	12544	581.11	4.63	4.19
33pr226	24355	48054	391.33	13.53	11.95	27pr226	24137	43228	190.14	8.16	7.94
-	-	-	-	-	-	21gr202	17904	42	267.42	3.91	7.20

Table 4.5: Computational results for  $BCA_{inc}$ , LS and GA on TSPLIB instances with grid clustering procedure,  $\mu = 7$  and 10, where  $BCA_{inc}$  found the optimal solution within 2 hours of CPU times. Both LA and GA found the optimal solutions in all instances.

Problem name	E	BCA		LS		GA	
		LB	UB	Soln	Time (sec)	Soln	Time (sec)
TSPLIB instances, center clustering							
28pr136	8879	30406.9	34003	34003	5.59	34003	5.70
30kroa150	10809	9330.25	9655	9655	7.33	9655	11.28
30krob150	10807	9568.43	9896	9896	6.78	9896	11.53
39rat195	18478	406.117	541	532	36.49	532	18.41
40kroa200	19409	10597.5	11442	11442	17.81	11442	18.28
40krob200	19430	10123.5	11080	11047	17.59	11047	18.72
40d198	18841	6662.5	6844	6844	17.19	6844	42.23
45ts225	24650	54554.8	62036	62080	196.02	62016	30.56
TSPLIB instances, grid clustering, $\mu = 3$							
50bier127	7729	70052.6	70965	70965	135.44	70965	14.86
60pr136	9064	49394	52569	52575	20.47	52562	22.13
57kroa150	11005	13318.8	13796	13796	206.31	13796	32.17
58u159	12321	23212.3	23916	23916	229.64	23916	23.47
81rat195	18745	635.438	646	649	67.33	646	61.05
72kroa200	19636	13613.6	14574	14533	246.41	14529	59.02
76krob200	19661	14028.1	14930	14930	328.14	14930	74.25
67d198	19101	7588.51	7956	7956	58.47	7956	44.02
75ts225	24900	73343.2	78639	78609	68.74	78609	136.94
84pr226	25118	58815.4	62156	62052	54.91	62052	74.66
TSPLIB instances, grid clustering, $\mu = 5$							
36kroa150	10868	9549.6	9929	9929	10.70	9929	31.05
36krob150	10870	9226.83	9580	9580	10.00	9580	29.55
33pr152	11083	37505.3	37985	37985	9.03	37985	16.19
32u159	12046	16498.4	16871	16871	8.03	16871	9.52
49rat195	18600	450.332	514	515	263.28	513	44.69
47kroa200	19464	11020.6	11383	11383	25.67	11383	23.22
48krob200	19511	10261.1	10866	10866	26.22	10866	24.27
40d198	18772	6527.09	6892	6892	15.88	6892	39.69
45ts225	24726	46446.4	60900	60496	210.20	60431	27.77
50pr226	24711	55338.1	56444	56444	265.52	56444	25.84
TSPLIB instances, grid clustering, $\mu = 7$							
25krob150	10725	6841.77	7174	7174	5.47	7174	5.00
36rat195	18454	349.141	409	408	18.27	408	15.25
35kroa200	19335	8016.5	9507	9476	14.61	9474	15.02
36krob200	19362	8802.5	9581	9580	14.97	9580	15.38
32d198	18372	5836.9	6329	6329	136.39	6329	17.83
35ts225	24544	42156.9	50753	50636	20.53	50635	15.73
TSPLIB instances, grid clustering, $\mu = 10$							
25rat195	18225	257.55	313	313	7.69	313	6.41
25kroa200	19100	6008.25	6754	6754	7.75	6754	22.11
25krob200	19082	6094.43	6801	6801	7.67	6801	21.06
25d198	18149	5600.93	6067	6053	6.66	6053	7.59
25ts225	24300	34771.1	40199	40187	8.64	40187	7.81

Table 4.6: Computational results for  $BCA_{inc}$ , LS and GA on TSPLIB instances where the optimal solution is unknown. (The time limit for  $BCA_{inc}$  was 2 hours.)

Problem characteristics			BCA		LS		GA	
$K$	$ V $	$ E $	Soln	Time (sec)	Soln	Time (sec)	Soln	Time (sec)
15	120	2,000	-54	7.63	-54	1.66	-54	1.66
		3,000	-69	2.77	-69	1.70	-69	2.00
		6,000	-83	3.16	-83	1.66	-83	1.59
	150	3,000	-82	0.33	-82	2.14	-82	1.45
		5,000	-80	0.33	-80	2.08	-80	1.91
		9,000	-98	19.24	-98	2.22	-98	1.94
	180	4,000	-81	8.30	-81	2.53	-81	2.30
		7,000	-96	6.60	-96	2.52	-96	2.39
		14,000	-94	11.36	-94	2.39	-94	2.31
20	120	1,500	-76	2.95	-76	2.70	-76	2.75
		3,000	-100	8.91	-100	2.86	-100	2.92
		6,000	-112	85.30	-112	2.66	-112	2.73
	160	3,000	-96	3.80	-96	4.09	-96	3.13
		5,000	-94	60.56	-94	3.78	-94	3.72
		10,000	-131	13.08	-131	3.52	-131	3.44
	200	5,000	-101	43.36	-101	4.94	-101	4.72
		10,000	-108	42.48	-108	4.58	-108	3.64
		15,000	-128	1.80	-128	4.50	-128	4.45
25	150	3,000	-89	88.73	-85	5.44	-89	5.67
		6,000	-111	19.47	-111	5.31	-111	4.14
		9,000	-130	3.81	-130	5.20	-130	4.78
	200	5,000	-111	12.00	-111	7.11	-111	6.99
		10,000	-134	60.33	-134	7.13	-134	6.59
		15,000	-148	12.39	-148	7.14	-147	6.56
30	120	2,000	-89	5.34	-89	5.78	-89	5.45
		3,000	-144	2.86	-144	6.17	-144	7.72
		6,000	-159	1.58	-159	6.30	-159	5.91
	150	3,000	-118	30.67	-118	7.22	-118	7.08
		6,000	-164	8.83	-164	7.14	-164	6.89
		9,000	-139	18.38	-139	7.09	-139	6.84
	180	4,000	-146	11.73	-146	9.30	-146	7.58
		7,000	-151	15.50	-151	9.17	-151	8.48
		14,000	-153	189.02	-153	8.38	-153	7.16
40	120	1,500	-112	2.58	-112	10.05	-112	9.33
		3,000	-157	22.00	-157	9.75	-157	9.45
		6,000	-180	12.83	-180	9.05	-180	9.58
	160	3,000	-142	9.25	-142	13.86	-142	12.77
		5,000	-170	20.14	-170	13.36	-170	12.70
		10,000	-205	6.97	-205	12.49	-205	11.36
	200	5,000	-177	150.20	-177	18.19	-177	17.00
		10,000	-195*	7,200.00	-194	16.94	-194	17.36
		15,000	-223	323.56	-223	15.33	-223	15.91

Table 4.7: Computational Results for  $BCA_{inc}$ , LS and GA on Random Instances (\* -  $BCA_{inc}$  did not complete search for this problem within 2 hours of CPU time. Upper bound at the time search was terminated was -194.)



Clustering Type	Number of Instances	Node Prizes $\{0, 1, \dots, 10\}$				Node Prizes $\{0, 1, 10, 100, 1000\}$			
		<i>BCA<sub>inc</sub></i>		<i>LS</i>	<i>GA</i>	<i>BCA<sub>inc</sub></i>		<i>LS</i>	<i>GA</i>
		opt. sol.	Avg. Time	Avg. Time	Avg. Time	opt. sol.	Avg. Time	Avg. Time	Avg. Time
center	41	33	250.16	8.70	4.55	41	55.89	8.54	7.17
$\mu = 3$	32	22	163.72	38.84	10.74	31	348.13	52.45	23.14
$\mu = 5$	32	22	526.12	3.53	3.83	31	56.82	7.46	10.18
$\mu = 7$	32	26	227.47	2.51	3.34	31	34.66	3.46	4.80
$\mu = 10$	32	27	76.03	1.66	2.00	32	93.63	2.24	3.50
Overall	169	130	241.53	10.23	4.70	166	113.95	14.37	9.56

Table 4.8: Summary of computational results for two different node-prize functions in instances where *BCA<sub>inc</sub>* found optimal solution within a 2 hour CPU time limit. (Our LS did not find optimal solution in 8 instances for the first node-prize function. In all other instances both LS and GA found optimal solutions.)

#### 4.4 Conclusion

In this chapter, we have presented two exact procedures for the prize-collecting generalized minimum spanning tree problem. One of them is a new implementation of the Rooting Procedure. We have found that this procedure performs well when a more deliberate choice of root clusters is implemented. We have also presented a novel branch-and-cut algorithm for the PCGMST problem, that works with integer incumbent nodes of the branch-and-bound tree. Our computational experiments suggest that these two procedures are a good choice for networks of small and moderate size. For larger networks, the metaheuristic procedures, such as genetic algorithm or local search are a better alternative. These metaheuristic procedures have shown excellent performance for two different types of functions used for the node prizes. In particular, out of 299 TSPLIB instances for which the optimum is known, our local search found optimum in 291 instances, while our genetic algorithm found optimal solutions in all 299 instances. For the random instances, both algorithms found optimal solution for 40 out of 41 instances for which optimal solution is known.

## Chapter 5

### Design of Optical Networks with Wavelength Division Multiplexing

#### 5.1 Introduction

Optical networks with wavelength division multiplexing (WDM) belong to the second generation of optical networks, designed to take advantage of the large optical network bandwidth. The design of WDM networks differs from classical telecommunication network design in the sense that here we need to work with multiple network layers and provide feasible routing of traffic in each of the layers. Over the past 10 years, this problem has been studied by many researchers. However, the solution procedures developed so far for this problem often work with very different restrictions on the optical network, which makes it difficult to compare their efficiency. The following four sections of this chapter explain the motivation behind different restrictions imposed in the WDM optical network design, and discuss the impact of these restrictions on the complexity of the corresponding design problem. Specifically, in Section 5.2 important aspects of WDM design are discussed. In Section 5.3 a formal problem definition is provided. After that, two mathematical formulations for the design of WDM optical networks are presented in Section 5.4, and the impact of different network restrictions on these formulations is discussed in Section 5.5. This chapter concludes with Section 5.6, which provides a literature review of heuristic and exact solution approaches developed for different versions of the WDM optical network design problem.

#### 5.2 Important aspects of WDM optical network design

Typical telecommunication network problems modelled and solved using different optimization techniques work with a network structure based on a well-known 7-layer ISO (International Standards Organization) standard (Figure 5.1). In this structure, the physical layer represents equipment, such as optical fibers, used to transmit signals. The data link layer processes signals

<b>Layer 7</b>	Application
<b>Layer 6</b>	Presentation
<b>Layer 5</b>	Session
<b>Layer 4</b>	Transport
<b>Layer 3</b>	Network
<b>Layer 2</b>	Data Link
<b>Layer 1</b>	Physical

Figure 5.1: ISO standard for telecommunications network structure

into the form in which signals will be transmitted in the physical layer, so it includes signal framing, multiplexing and demultiplexing. The network layer provides routing of requests from higher layers, while the transport layer ensures reliable transfer of traffic requests. The session layer handles user access to the network in terms of how the session is established (beginning, ending) and security issues. The presentation layer processes signals from the user side to the form in which they will be transmitted through the network. Finally, the application layer deals with the connection of specific software applications with the network.

Today's networks, however, have a more complex structure. This complexity comes, in part, from the fact that today's optical networks are used to provide service for many different client layers (such as ATM, SONET, or SDH), which instead of direct routing in the physical layer, may use one of the lower server layers for the routing of its own traffic. In other words, these networks may have multiple sets of the first three layers, which are recursively embedded into each other. (To be more precise in technical terms, each higher client layer uses the next lower server layer as its data link layer [53].) This difference between the ISO structure of the first three layers and the structure of today's optical networks is illustrated in Figure 5.2. The embedded layered network

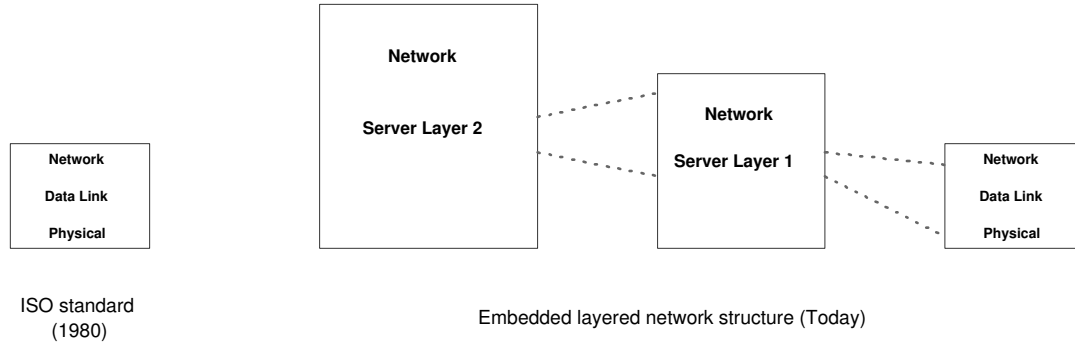


Figure 5.2: Change in the complexity of telecommunications networks

structure shown in Figure 5.2, occurs, for example, in networks where an ATM layer uses service of the lower SONET or SDH layer. In this case, the optical layer (with physical layer represented by optical fibers) provides optical paths called **lightpaths** to the SONET or the SDH layer. The SONET or SDH layer then use lightpaths to route their own connections, which are then used to serve the ATM layer. So, in a way (not literally), lightpaths take the role of the physical layer in the SONET or SDH layer, and, similarly, the SONET or SDH connections take the role of the physical layer for the ATM layer. More details and additional examples of embedded layered network structures can be found in the text by Ramaswami and Sivarajan [53].

Although there could be multiple server layers nested in each other (as in the example of the ATM layer over the SONET or SDH), most of the current research dealing with the design of WDM networks considers the existence of a single server layer only (commonly referred to as logical layer, logical topology or virtual topology). In this dissertation we also focus on single server layer optical networks only.

### 5.2.1 Physical Topology

The physical topology of a WDM network consists of physical nodes interconnected by fiber links. Each node may contain different equipment including transmitters and receivers, multiplexers and demultiplexers, amplifiers, wavelength changers, etc. However, the only node equipment usually considered in the optimization models for the WDM network design includes transmitters,

receivers and wavelength changing devices. Specific characteristics of each of these devices depends on the requirements or restrictions in a given network. For example, certain types of transmitters and receivers are tunable to all wavelengths, while the other ones are tunable to just a specific wavelength. Usually it is considered that transmitters and receivers are tunable to all wavelengths, but there are studies that consider both of these cases (see [62] for example). Further, some WDM optical networks provide full wavelength conversion at all nodes of the network, and some WDM optical networks provide no wavelength conversion at all. The existence of wavelength converters in the network represents a very important aspect of optical network design, as it significantly changes the complexity of the problem. That is, when there is no wavelength conversion, we also need to make sure that each lightpath uses exactly one wavelength along its propagation path in the physical layer. Although it has been recognized that sparse wavelength conversion is sufficient in most networks (see [12], [13]), most papers on WDM network design problem assume that either all the nodes in the network have wavelength conversion ability (see [6], [54]), or that all nodes in the network are without wavelength conversion ability (as in [35], [37], [44]). (See Ramaswami and Sivarajan [53] for more details on the tradeoffs between optical crossconnects with and without wavelength conversion capability.) While there are examples of models for optical networks with sparse wavelength conversion (see [52] for example), these types of optical networks are usually not studied as the use of the sparse wavelength conversion significantly increases the problem complexity. This dissertation focuses only on optical networks with wavelength converters available at all nodes of the network. Our choice is partly driven by the results of previous studies for WDM optical network design, which indicate that wavelength conversion may be desirable in WDM networks as a way of resolving equipment compatibility issues [6].

In addition to transmitters, receivers, and wavelength changers, other types of node equipment may be considered as well. For example, Farahat et al. [22] considered the use of different optical crossconnect (OXC) switches that receive/transmit lightpaths and allow lightpaths to go through node without opto-electronic conversion.

With regard to the restrictions imposed by optical fibers, usually two factors are considered

- fiber capacity and the number of wavelengths/lightpaths that can be supported by a given fiber. Normally, the number of wavelengths supported by a given fiber depends on the characteristics of the fiber and transmission requirements. Consequently, in most of the research on WDM network design it is considered that the number of wavelengths in the network is finite. Some studies of the WDM optical network design problem, however, consider that the number of fibers is sufficiently large and treat the number of available wavelengths as infinite (see [36] for example), which then makes the whole design problem easier to solve. This assumption is also sometimes justified by the fact that the number of wavelengths that can be used in a fiber has been gradually increasing in the recent past. (In the year 2000 a single fiber capacity was about 100 wavelengths [21].) However, a computational study by Ramaswami and Sivarajan [54] indicates that even when the number of available wavelengths is not an issue and the constraint on the number of wavelengths can be relaxed, the WDM optical network design problem can be solved to optimality only for networks with up to 6 nodes.

Finally, an important issue related to the physical topology is whether the network already exists, or has a possibility of expanding, or is just being built. Although some papers take network expansion into consideration (see for example [6]), most often it is assumed that the physical topology is given and fixed [44], [54]. Sometimes, it is considered that only a part of the physical topology is fixed, and a part of it subject to change. For example, Hu and Leida [35], and Konda and Chou [36], assumed that the number of transmitters and receivers available at the nodes is unknown and needs to be determined, while the remaining part of the physical topology is fixed.

### 5.2.2 Logical Topology

Logical topology can be thought of as a network of physical nodes connected by a set of virtual edges, or lightpaths, between the nodes. This topology is an optical layer that provides service to client layers residing over the optical layer, such as IP, ATM, SONET/SDH [53]. In order to design a logical topology, it is necessary to determine which nodes should be connected by the lightpaths, and how these lightpaths should be routed in the physical layer. Additionally,

we need to decide whether we should establish symmetrical lightpaths throughout the network (that is, we need to decide whether for each specific lightpath from node  $i$  to node  $j$ , there should also be a corresponding, symmetrical lightpath, lightpath from node  $j$  to node  $i$ ). In practice, almost all higher layer protocols, including IP and SONET, assume bidirectional physical layer links that can simultaneously carry traffic in both directions [53], and, therefore, it is not uncommon to have symmetrical lightpaths established across the network. However, asymmetric traffic requests tend to be more efficiently served using asymmetrical lightpaths. Additionally, asymmetric traffic requests are sometimes more difficult to configure over the bidirectional links, which then warrants the use of unidirectional links that are more convenient for configuration of asymmetrical lightpaths. We will assume that physical links in the network are unidirectional, and that symmetry of lightpaths is not required.

### 5.2.3 Traffic Requirements

Traffic carried over WDM optical networks can be represented in several different forms. In some cases, traffic demand between pairs of nodes can be represented in terms of the number of lightpath requests. In other cases, traffic demand can be represented as a fraction of lightpath capacity (measured in packets/second for example). In the latter case, the WDM optical network design includes the traffic grooming problem that deals with multiplexing multiple traffic demands over a single lightpath. Additionally, when considering traffic grooming as part of WDM optical network design, traffic may be considered in terms of the number of separate connections between each pair of nodes. In this case, each connection request represents a bandwidth requirement. In certain cases traffic demand is granular, that is, it can be of certain pre-specified size only. (For example, traffic demand may be measured in terms of the number of OC-3, OC-12, OC-48 connections (as in [62]).)

Another issue related to traffic routing over the logical topology is whether bifurcation of traffic should be allowed or not (that is whether traffic of a single commodity can be sent over multiple paths or not). As indicated by Ramaswami and Sivarajan [53], bifurcation of traffic is not

a problem if the traffic is represented by IP packets, but the same does not hold for SONET circuits. However, as pointed out by Zhu and Mukherjee [62], bifurcation of traffic should not be permitted in networks that allow grooming of the higher speed connections. In this case, every connection request needs to be carried as is without division into lower-speed connections that could then be carried on separate paths. This aspect of WDM optical network design has been modelled in [35] and [62]. It is often considered that all traffic requests for a single origin-destination pair can be treated as a single commodity (see [35] and [38]), but other studies (as in [62]), consider that each origin-destination pair may have multiple low-speed requests, and it is only required that each of these low-speed requests is not bifurcated. (The computational study carried in [62] did, however, indicate that most of common origin-destination connections are routed together on the same lightpaths, even when there is a choice of routing common origin-destination connections over different lightpaths.)

Finally, although most of the research on WDM network design has so far considered static traffic requirements only, it is recognized that traffic in the IP environment has a dynamic and stochastic nature (see [43]). Some recent work on dynamic traffic includes studies of Gençata and Mukherjee [29], and Fard et al. [23].

#### 5.2.4 Design Objectives

Some of the common design objectives used for WDM optical networks include minimization of the network-wide average packet delay, maximization of the traffic matrix scale factor, minimization of the congestion, minimization of the number of transponders, minimization of the total lost traffic, and minimization of the total weighted lost traffic. The rationale for using each of these objectives is discussed next.

**Minimizing the network-wide average packet delay.** This design objective targets the quality of service provided by a given optical network. For example, in [44] two components of packet delay are considered - propagation and queueing delay. Propagation delay is defined as the sum of propagation delays over all used links multiplied by the total traffic carried over each individual



link, while the queueing delay is defined using an M/M/1 queueing model for each lightpath, and has a non-linear form. However, as noted in [44], the queueing delay can usually be disregarded, since propagation delay has a greater effect than queueing delay, as long as links are not too close to full utilization.

**Maximizing the traffic matrix scale factor.** This design objective can be used to ensure that the network is set up so that there is maximum capacity available for future demands. In [44], this design objective function was modelled as a nonlinear function representing the maximum amount of traffic that goes through any lightpath.

**Minimizing the congestion (maximum load on any link).** This design objective is used when the goal is to balance network congestion evenly throughout the network (see [54] for example) .

**Minimizing the number of transponders.** This design objective is of importance in networks that are subject to expansion, as the cost of transponders and optical amplifiers has a major role in a nationwide optical network [35].

**Minimizing the total lost traffic (maximizing the network throughput).** This objective is of importance in situations where all of the traffic cannot be served, and the goal is simply to provide service for as much traffic as possible. (An example of a model using this design objective can be found in [62]).

**Minimizing the total weighted lost traffic.** This objective is similar in nature to the previous one, but with more value given to larger traffic demands. In [62], for example, a network revenue model was considered, where each connection was assigned a weight determined as the product of the shortest path distance between the end nodes and the bandwidth requirement factor (bandwidth requirement was scaled between 0 and 1) for that connection. Experimental findings on a 6 node network indicated that, for this objective function, packing the connections for the same origin-destination pair does not represent the most efficient way of traffic grooming.

In this dissertation, we consider single objective WDM optical network design models only. However, we recognize that in certain cases, multiple objectives need to be considered in WDM op-

tical network design. For example, in [38] and [39], two objectives are considered: minimization of the weighted delay (measured as average weighted hop count of each lightpath) and maximization of the network throughput. In [50], three objectives are considered: minimization of the average propagation delay (or minimization of the average hop count), maximization of the network throughput, and minimization of the number of transceivers. In [50], these multiple objectives were evaluated together using the notion of Pareto optimal sets within a multi-objective evolutionary algorithm.

### 5.3 Problem Statement

The general WDM optical network design problem can be stated in the following way. Given the restrictions on the physical and logical topology of the WDM network, and traffic requests between all pairs of nodes, determine the number and routing of the lightpaths in the physical topology, and routing of the traffic in the logical topology so that the desired objective is minimized/maximized. Due to the computational intractability of mathematical formulations for this problem, the WDM optical network design problem is often solved sequentially using two separate problems: logical topology design problem and lightpath routing problem. The **Logical topology design problem** (LTD problem), or lightpath topology design problem can be stated as follows. Given the number of transmitters and receivers at each node of the network, capacity of the lightpaths, and traffic demand, determine the number of lightpaths to be established between each pair of nodes and routing of traffic over the established logical topology, so that a desired objective is minimized/maximized.

When traffic requests have lower capacity requirements than the capacity of the lightpaths used to carry them, and when there is an option of multiplexing the signal with other low-speed requests, the problem of routing the traffic over the established logical topology is referred to as the **traffic grooming problem**. (The traffic grooming problem, basically, deals with the packing of low speed connections into the wavelengths and their routing over the logical topology.) Depending on the original form of traffic requests, and whether bifurcation of flow is allowed, the

traffic grooming problem may be a special form of the bin packing problem where bin capacity is defined by the capacity of lightpath. In this case traffic requests represent items that are supposed to be placed in bins, and they can be of identical size (if represented by traffic streams of identical size [36]), or different size (if traffic demand is granular as in [43], [62] and [15] (i.e., it can take one of the finite number of pre-specified forms)), which are then combined into higher speed connections.

Traffic grooming has received significant attention in research related to ring networks, and its importance in general mesh networks is also increasing due to the growth of Internet traffic [43]. Some recent papers on traffic grooming within WDM mesh networks can be found in [35], [36], and [38].

The second part of the WDM network design problem, the **lightpath routing problem**, deals with routing of lightpaths in the physical topology and can be defined in the following way. Given the physical and logical topology (number of lightpaths between all pairs of nodes), determine a routing of lightpaths in the logical topology so that the desired objective is minimized/maximized.

When wavelength assignment needs to be taken into consideration, this problem is referred to as **routing and wavelength assignment (RWA)** problem, which in addition to routing of lightpaths, also requires that each lightpath is assigned a wavelength without violation of the wavelength requirements in a given network. As mentioned earlier, the wavelength assignment problem is not relevant in networks with wavelength changers, as lightpaths can always be reassigned to a new wavelength at every node in the network. Therefore, the wavelength assignment problem is not considered in this dissertation.

As previously mentioned, most of the previous research work sequentially solves the WDM optical network design problem using the logical topology design problem and the lightpath routing problem. In this dissertation, we focus on development of exact procedures that *integrate* these two problems and solve them simultaneously.

Figure 5.3 illustrates an example of WDM optical network design problem where we need to determine the logical topology and traffic routing in a five-node optical network so that the total number of transmitters and receivers in the network is minimized. The physical topology of the

optical network is shown in Figure 5.3 (a). Each arc in this network represents an optical fiber with a capacity of 3 lightpaths. We are also given a set of 10 commodities and the corresponding traffic requests (expressed as fractions of individual lightpath capacity).

One possible feasible solution for this example is to establish a single lightpath for each commodity, which would then mean that we need total of 20 transmitters and receivers in the network. (Recall that each lightpath requires one transmitter and one receiver. Therefore, in order to establish 10 lightpaths, one for each commodity, we need total of 20 transmitters and receivers.) The logical topology and traffic routing for this solution are shown in Figure 5.3 (c). (The light arcs represent lightpaths, and the dark arcs represent flow paths for each commodity.) In order to completely specify solution for this problem, we also need to determine the routing of lightpaths in the physical layer. Since, in this example, there is sufficient fiber capacity in the underlying physical topology, we can route each lightpath over a single direct optical link for each pair of demand nodes. For example, the lightpath between nodes 2 and 1 carrying traffic for commodity (2, 1) can be routed over optical fiber (2, 1) in the physical layer. However, notice that, depending on a specific problem, the lightpath routing problem may be much more complex. For example, if there was no direct optical fiber between nodes 2 and 1, we would have to either use multiple physical links for the lightpath between nodes 2 and 1, or carry the flow of commodity (2, 1) over multiple lightpaths (in some instances it is also possible that the flow of a specific commodity is carried over multiple lightpaths, where one or more of these lightpaths use multiple physical links).

Now, going back to the network design objective for this example, we should ask a question of whether it is possible to serve all the demand with a smaller number of transmitters and receivers. Just by examining demand at each origin and destination node we see that *at least* 16 transmitters and receivers are needed. (We can determine this number by solving the bin packing problem at each origin and destination node. For example, the minimum number of lightpaths originating at node 1 is equal to the minimum number of lightpaths needed to carry all the demand originating from this node without bifurcation of flow. This is equivalent to solving the bin packing problem

where the traffic request of each commodity corresponds to an item that needs to be placed in one of the bins, and the bins correspond to lightpaths originating at node 1. By examining the demand originating at node 1, we see that two lightpaths are needed to carry all the traffic originating at that node. Following the same logic for all other traffic origin and destination nodes, we can determine the minimum number of transmitters and receivers needed in this network.)

However, we now need to answer the question of whether it is possible to establish a logical topology and traffic routing so that only 16 transmitters and receivers are needed. It turns out that such a solution does exist, and we show the corresponding logical topology and traffic routing in Figure 5.3 (d). We can see that in order to minimize the total number of transmitters and receivers in the network, 2 lightpaths from the solution in Figure 5.3 (c) are eliminated (these are lightpaths (1, 2) and (1, 5)). As a result, some commodities had to be combined on certain lightpaths. (Specifically, commodity (1, 2) is now using two lightpaths (1, 4) and (4, 2), which are also used by commodities (1, 4) and (4, 2). Similarly, commodity (1, 5) is now using two lightpaths (1, 3) and (3, 5), which are also used by commodities (1, 3) and (3, 5).)

#### 5.4 Arc Based Mathematical Formulations for the WDM Optical Network Design

In this chapter, we will present two arc-based formulations for the WDM optical network design problem that differ in the types of variables used to define the flow of different commodities over the lightpaths in the network. In the first formulation, MIP-ARC-GLOBAL, the flow of a given commodity is defined in the logical topology only, and does not provide information regarding the exact propagation path of a given commodity in the physical topology. Consequently, the flow variables specify only origins and destinations of the lightpaths used. The second formulation, MIP-ARC-LOCAL, uses flow variables that specify the exact propagation of the flow of a given commodity both in the logical and in the physical topology.

In both of these formulations we have assumed that individual traffic requests do not exceed the capacity of a lightpath. Consequently, we have scaled all traffic requests by the capacity of a single lightpath, so that a bandwidth requirement of one traffic unit (TU) is equal to the capacity

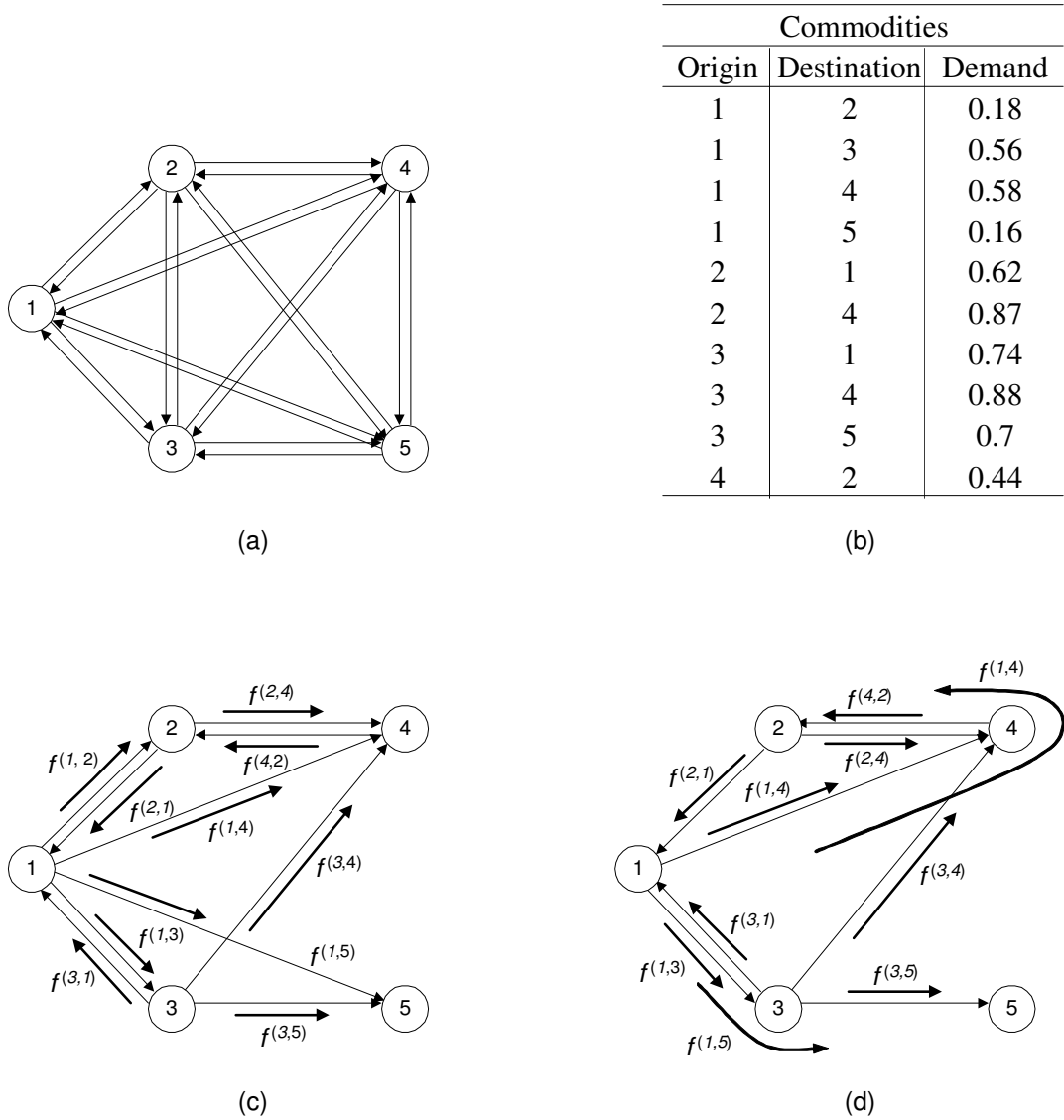


Figure 5.3: A five-node example of the WDM optical network design problem. (a) Physical Topology. (b) List of commodities and the corresponding demand. (c) Feasible logical topology and traffic routing. (d) Optimal logical topology and traffic routing.

of a lightpath. We will first introduce notation used in this chapter, and then discuss formulations MIP-ARC-GLOBAL and MIP-ARC-LOCAL.

**Notation**

$G = (V, A)$  - The physical topology defined over a set of nodes  $V$  connected by a set of arcs (direct fiber connections)  $A$

$T^{(s,d)}$  - Total (scaled) demand between nodes  $s$  and  $d$

$\Omega$  - Set of demands

$\Lambda$  - Set of all possible lightpath origin-destination pairs. When it is possible to set up lightpaths between all pairs of nodes in the network we have  $\Lambda = V \times V$ .

$L_{ij}$  - Number of wavelengths available between nodes  $i$  and  $j$  that are directly connected by optical fibers. When multiple fibers are available, this number represents a product of the number of fibers and the number of wavelengths available at each fiber.

$\Delta_t^i$  - Number of transmitters available at node  $i$

$\Delta_r^j$  - Number of receivers available at node  $j$ .

**MIP-ARC-GLOBAL formulation:**

**Variables:**

$H^{(s,d)}$  - Amount of traffic of commodity  $(s, d)$  that is not served. These variables are referred to as **lost traffic** variables

$Y^{(i,j)}$  - Number of lightpaths with origin at node  $i$  and destination at node  $j$  included in the logical topology. These variables will be referred to as **global lightpath variables**, as they do not provide the information regarding exact propagation paths of the lightpaths in the physical topology

$X_{(l,m)}^{(i,j)}$  - Number of lightpaths with origin at node  $i$  and destination at node  $j$ , using physical arc  $(l, m)$ .

$f_{(i,j)}^{(s,d)}$  - Amount of flow (traffic) of commodity  $(s, d)$  carried over the lightpaths with origin at node  $i$  and destination at node  $j$

MIP-ARC-GLOBAL:

$$\text{Min} \sum_{(s,d) \in \Omega} T^{(s,d)} H^{(s,d)} \quad (5.1)$$

Subject to:

$$\sum_{j:(i,j) \in \Lambda} Y^{(i,j)} \leq \Delta_t^i \quad \forall i \in V \quad (5.2)$$

$$\sum_{i:(i,j) \in \Lambda} Y^{(i,j)} \leq \Delta_r^j \quad \forall j \in V \quad (5.3)$$

$$\sum_{(i,j) \in \Lambda} X_{(l,m)}^{(i,j)} \leq L_{lm} \quad \forall (l,m) \in A \quad (5.4)$$

$$\sum_{l:(l,k) \in A} X_{(l,k)}^{(i,j)} - \sum_{m:(k,m) \in A} X_{(k,m)}^{(i,j)} = \begin{cases} Y^{(i,j)} & \text{if } k = j \\ -Y^{(i,j)} & \text{if } k = i \\ 0 & \text{otherwise} \end{cases} \quad \forall (i,j) \in \Lambda, k \in V \quad (5.5)$$

$$\sum_{i:(i,k) \in \Lambda} f_{(i,k)}^{(s,d)} - \sum_{j:(k,j) \in \Lambda} f_{(k,j)}^{(s,d)} = \begin{cases} 1 - H^{(s,d)} & \text{if } k = d \\ H^{(s,d)} - 1 & \text{if } k = s \\ 0 & \text{otherwise} \end{cases} \quad \forall (s,d) \in \Omega, k \in V \quad (5.6)$$

$$Y^{(i,j)} - f_{(i,j)}^{(s,d)} \geq 0 \quad \forall (s,d) \in \Omega, (i,j) \in \Lambda \quad (5.7)$$

$$Y^{(i,j)} - \sum_{(s,d) \in \Omega} T^{(s,d)} f_{(i,j)}^{(s,d)} \geq 0 \quad \forall (i,j) \in \Lambda \quad (5.8)$$

$$f_{(i,j)}^{(s,d)} \in R_+^1 \quad \forall (i,j) \in \Lambda, (s,d) \in \Omega \quad (5.9)$$

$$H^{(s,d)} \in R_+^1 \quad \forall (s,d) \in \Omega \quad (5.10)$$

$$Y^{(i,j)} \in R_+^1 \quad \forall (i,j) \in \Lambda \quad (5.11)$$

$$X_{(l,m)}^{(i,j)} \in Z_+^1 \quad \forall (l,m) \in A, (i,j) \in \Lambda \quad (5.12)$$

The first expression in the above formulation is the objective function that minimizes the total lost traffic. Constraints (5.2) and (5.3) are degree constraints for each node. They ensure that the total number of lightpaths originating/terminating at any node is limited by the number of transmitters/receivers at that node. Constraint (5.4) provides a bound on the number of lightpaths that can be established over a single arc. Constraint (5.5) is the flow balance constraint for the



lightpaths. This constraint also guarantees that if the  $X_{(l,m)}^{(i,j)}$  variables are integer, then the  $Y^{(i,j)}$  variables are integer as well. Constraint (5.6) is the flow balance constraint for the traffic routed over established lightpaths. Constraint (5.7) limits traffic to lightpaths established in the logical topology. Constraint (5.8) limits the total amount of flow that can be sent over any single lightpath. Note that constraint (5.7) is redundant given the presence of constraint (5.8), but it is used here to strengthen the formulation.

The MIP-ARC-GLOBAL formulation closely resembles the formulation presented by Banerjee and Mukherjee in [6]. They also define additional constraints to ensure that the entire capacity of any given lightpath is not used, and they include constraints that limit propagation delay of the lightpaths in the physical topology. The objective function in their formulation was the minimization of the average logical hop distance.

#### **MIP-ARC-LOCAL formulation:**

##### **Variables:**

$H^{(s,d)}$  - Amount of traffic of commodity  $(s, d)$  that is not served. These variables are referred to as **lost traffic** variables

$Y^{(i,j);k}$  - Indicator variable equal to 1 if the  $k^{th}$  lightpath with origin at node  $i$  and destination at node  $j$  is used. These variables will be referred to as **local lightpath variables** in reference to the fact that they correspond to the specific lightpaths whose propagation paths in the physical topology are specified in the solution of the MIP-ARC-LOCAL formulation. (Note: The total number of lightpaths with origin at node  $i$  and destination at node  $j$  needs to be specified in this formulation. We can use the maximum number of transmitters and receivers, or some other criteria to specify this number.)

$\kappa_{(i,j)}$  - set of possible lightpaths with origin at node  $i$  and destination at node  $j$  ( $\kappa_{(i,j)} = \{1, 2, \dots, |\kappa_{(i,j)}|\}$ )

$X_{(l,m)}^{(i,j);k}$  - Indicator variable equal to 1 if the  $k^{th}$  lightpath, with origin at node  $i$  and destination at node  $j$  is using physical arc  $(l, m)$ .

$f_{(i,j);k}^{(s,d)}$  - Amount of flow of commodity  $(s, d)$  carried over the  $k^{th}$  lightpath with origin at node  $i$

and destination at node  $j$

MIP-ARC-LOCAL:

$$\text{Min} \sum_{(s,d) \in \Omega} T^{(s,d)} H^{(s,d)} \quad (5.13)$$

Subject to:

$$\sum_{j:(i,j) \in \Lambda, k \in \kappa_{(i,j)}} Y^{(i,j);k} \leq \Delta_t^i \quad \forall i \in V \quad (5.14)$$

$$\sum_{i:(i,j) \in \Lambda, k \in \kappa_{(i,j)}} Y^{(i,j);k} \leq \Delta_r^j \quad \forall j \in V \quad (5.15)$$

$$\sum_{(i,j) \in \Lambda; k \in \kappa_{(i,j)}} X_{(l,m)}^{(i,j);k} \leq L_{lm} \quad \forall (l,m) \in A \quad (5.16)$$

$$\sum_{l_1:(m,l_1) \in A} X_{(m,l_1)}^{(i,j);k} - \sum_{l_2:(l_2,m) \in A} X_{(l_2,m)}^{(i,j);k} = \begin{cases} Y^{(i,j);k} & \text{if } m = i \\ -Y^{(i,j);k} & \text{if } m = j \\ 0 & \text{otherwise} \end{cases} \quad \forall (i,j) \in \Lambda; k \in \kappa_{(i,j)}, m \in V \quad (5.17)$$

$$\sum_{j_1:(i,j_1) \in A, k \in \kappa_{(i,j_1)}} f_{(i,j_1);k}^{(s,d)} - \sum_{j_2:(j_2,i) \in A, k \in \kappa_{(j_2,i)}} f_{(j_2,i);k}^{(s,d)} = \begin{cases} 1 - H^{(s,d)} & \text{if } i = s \\ H^{(s,d)} - 1 & \text{if } i = d \\ 0 & \text{otherwise} \end{cases} \quad \forall (s,d) \in \Omega, i \in V \quad (5.18)$$

$$Y^{(i,j);k} - f_{(i,j);k}^{(s,d)} \geq 0 \quad \forall (s,d) \in \Omega, (i,j) \in \Lambda; k \in \kappa_{(i,j)} \quad (5.19)$$

$$Y^{(i,j);k} - \sum_{(s,d) \in \Omega} T^{(s,d)} f_{(i,j);k}^{(s,d)} \geq 0 \quad \forall (i,j) \in \Lambda; k \in \kappa_{(i,j)} \quad (5.20)$$

$$f_{(i,j);k}^{(s,d)} \in R_+^1 \quad \forall (s,d) \in \Omega, (i,j) \in \Lambda; k \in \kappa_{(i,j)} \quad (5.21)$$

$$H^{(s,d)} \in R_+^1 \quad \forall (s,d) \in \Omega \quad (5.22)$$

$$X_{(l,m)}^{(i,j);k} \in B^1 \quad \forall (i,j) \in \Lambda, k \in \kappa_{(i,j)}, (l,m) \in A \quad (5.23)$$

$$Y^{(i,j);k} \in R_+^1 \quad \forall (i,j) \in \Lambda; k \in \kappa_{(i,j)} \quad (5.24)$$

The main difference between formulations MIP-ARC-GLOBAL and MIP-ARC-LOCAL is the presence of index  $k$ , both in the lightpath variables and in the flow variables. Since the lightpaths

variables now correspond to the specific lightpaths, both  $Y^{(i,j);k}$  and  $X_{(l,m)}^{(i,j);k}$  variables are defined as binary variables. With this exception, constraints (5.14) to (5.24) have the same interpretation as the constraints in the MIP-ARC-GLOBAL formulation. An apparent drawback of the MIP-ARC-LOCAL formulation is that it requires the number of possible lightpaths between all pairs of nodes to be known in advance. This is an important issue, since that number has a direct impact on the number of variables and constraints in the MIP-ARC-LOCAL formulation. Although this formulation may seem to be less desirable than the MIP-ARC-GLOBAL formulation, we will see in the next section that the MIP-ARC-LOCAL formulation has certain advantages when bifurcation of flow is not allowed in the network. (Notice that both formulations MIP-ARC-LOCAL and MIP-ARC-GLOBAL allow bifurcation of the flow.)

In the next section we will discuss how different network restrictions encountered in the optical network design influence the complexity of the WDM optical network design problem.

## 5.5 Impact of Different Network Restrictions on the Complexity of Mathematical Formulations for the WDM Optical Network Design Problem

For easier differentiation of optical networks with different properties, the following three-letter notation will be used for different network settings:  $\cdot / \cdot / \cdot$ . The first index refers to the logical topology described by one of the following two letters:

*F* - Fixed - indicates that the logical topology is fixed, i.e., exact propagation paths of all lightpaths are known. Typically, fixed logical topologies include only a small subset of possible lightpaths that are heuristically selected.

*U* - Unknown - indicates that no information on the number or propagation paths of the lightpaths is provided.

The second index refers to transmitters and receivers in the network, which could be:

*F* - Fixed - indicates that the number of transmitters and receivers is limited in the network, and their number at each node is known,

*U* - Unknown - indicates that the number of transmitters and receivers in the network needs to be

Logical Topology	T/R	Bifurcation	
		A	N
F	X	ODMCF	ODIMCF
	F	OD MCF with <i>deg</i> constr.	ODIMCF with <i>deg</i> constr.
	U	OD MCF with <i>deg</i> constr.	ODIMCF with <i>deg</i> constr.
U	X	2-layer ODMCF	2-layer ODIMCF
	F	2-layer ODMCF with <i>deg</i> constr.	2-layer ODIMCF with <i>deg</i> constr.
	U	2-layer ODMCF with <i>deg</i> constr.	2-layer ODIMCF with <i>deg</i> constr.

Table 5.1: Impact of different network restrictions on the nature of the corresponding network design problem

determined as a part of the network design, or

*X* - Relaxed - indicates that the number of transmitters and receivers in the network is not considered to be constraint.

The last index refers to the bifurcation of flow in the network, which could be:

*A* - Allowed, or

*N* - Not allowed.

Using this notation, the formulations presented in the previous section represents a model for WDM optical network with *U/F/A* setting. Additionally, whenever degree, propagation delay, or lightpath capacity constraints are part of the network restrictions, we will use the shorthand notation *deg/del/cap*, indicating that these restrictions need to be taken into consideration in the design of a specific network. We will next discuss the impact of different network restrictions on the formulations presented in the previous section.

First, note the close resemblance of the formulations MIP-ARC-GLOBAL and MIP-ARC-LOCAL to the standard origin-destination multicommodity flow problem (ODMCF problem). Actually, if we assume that the logical topology is given, the problem reduces to the ODMCF problem with additional degree constraints, a problem that can be efficiently solved using column generation. When the bifurcation of flow is not allowed, the WDM optical network design problem with a fixed logical topology becomes a standard origin-destination integer multicommodity flow (ODIMCF) problem. Table 5.1 provides a summary of the type of multicommodity flow (MCF) problems that need to be solved depending on different settings in a given WDM optical network. Notice that network settings with fixed and unknown number of transmitter and receivers corre-

respond to the same type of the MCF problem. The only difference between the formulations for these network settings is that when the number of transmitters and receivers needs to be determined as a part of the design problem, we need to add variables corresponding to the number of transmitters and receivers at each node.

Now, note that the requirement that there is no bifurcation of flow, significantly complicates the MIP-ARC-GLOBAL formulation. Not only do the flow variables need to be defined as binary variables, but this additional restriction does not suffice to ensure no bifurcation of flow. This is due to the fact that in our current formulation flow variables are defined over global lightpath variables, and nothing ensures that flow of a given commodity will not be split among different lightpaths with the same origin and destination. Ensuring no bifurcation can be achieved through the addition of the following set of new constraints.

$$Y^{(i,j)} - B_h^{(i,j)} S_h^{(i,j)} \geq 0 \quad \forall (i,j) \in \Lambda, h \in H \quad (5.25)$$

$$C_h - \sum_{(s,d) \in h} f_{(i,j)}^{(s,d)} + S_h^{(i,j)} \geq 1 \quad \forall (i,j) \in \Lambda, h \in H \quad (5.26)$$

$$S_h^{(i,j)} - f_{(i,j)}^{(s,d)} \leq 0 \quad \forall (i,j) \in \Lambda, h \in H, (s,d) \in h \quad (5.27)$$

$$S_h^{(i,j)} \in B^1 \quad \forall (i,j) \in \Lambda, h \in H \quad (5.28)$$

These constraints are bin packing (*pack*) constraints, where the number of lightpaths represents the number of bins, the capacity of each lightpath represents the size of each bin, and demands of commodities that are supposed to be carried by a given set of lightpaths represent the size of individual items that need to be placed in the bins. More specifically, the term  $B_h^{(i,j)}$  in the constraint (5.25) represents a pre-calculated number of lightpaths needed to carry the traffic of all commodities  $(s,d) \in h$  without bifurcation. Set  $h$  represents one of all possible subsets of commodities in the network (set  $H$  denotes the set of all subsets of commodities). The term  $S_h^{(i,j)}$  in the constraint (5.25) represents an indicator variable equal to one, only if all commodities that belong to set  $h$  use global arc  $(i,j)$ . Constraints (5.26) and (5.27) ensure proper definition of the indicator variable  $S_h^{(i,j)}$ . The term  $C_h$  in the constraint (5.26) represents the cardinality of set  $h$ . Consequently, constraint (5.26) guarantees that variable  $S_h^{(i,j)}$  will be equal to one if all

commodities from set  $h$  use arc  $(i, j)$ . Constraint (5.27) guarantees that  $S_h^{(i,j)}$  will be equal to zero if at least one of the commodities from set  $h$  is not using global arc  $(i, j)$ .

The MIP-ARC-LOCAL formulation has an advantage over the MIP-ARC-GLOBAL formulation in the sense that we do not need to add *pack* constraints in order to ensure no bifurcation of flow (recall that the lightpath variables in the MIP-ARC-LOCAL formulation correspond to specific lightpaths). Instead, it is sufficient to define the flow variables as binary variables. Also, notice that in order to add the packing constraints, it is necessary to know the number of possible lightpaths between all pairs of nodes in advance (the same problem that we discussed for the MIP-ARC-LOCAL formulation). Another issue related to the MIP-ARC-GLOBAL formulation is that this formulation cannot be used for networks where we need to place a limit on the number of physical hops for each individual flow path. The reason is that the flow variables are not mapped to the local lightpath variables, so there is no way to determine the number of physical hops used by each individual flow path.

A special case of the WDM optical network design problem with infinite number of wavelength changers arises in situations when it is reasonable to assume that the number of lightpaths does not represent an actual restriction (and can be therefore relaxed), and there are no additional restrictions related to the links of the physical network. In this case, the physical topology can be completely discarded and replaced by an auxiliary network with edges represented by lightpaths. The WDM optical network design problem in this case becomes a single layer ODIMCF problem with *deg* constraints. This problem was studied by Ramaswami and Sivarajan [54], but due to the size of the formulation, they were able to solve problems on networks with 6 nodes only. Similarly, Chang et al. [11], provided a mathematical formulation for the low-earth orbit (LEO) satellite network design problem (which turns out to be equivalent to the WDM optical network design problem with infinite number of wavelengths, wavelength changers, and bifurcation of flow allowed), but no computational tests were provided due to the NP-hard nature of the problem.

Finally, it should be noted that the WDM optical network design problem is much easier to solve when demand can be expressed in terms of lightpath requests. Höller and Voß [34] have

recently shown that in such networks, exact mathematical formulations can be solved to optimality for significantly larger problems, compared to the case when demands are expressed as fractions of the lightpath capacity. It is noteworthy that the mathematical formulation proposed in [34] had additional restrictions that are not present in the WDM optical network design problem studied in this dissertation. Specifically, in order to increase reliability of the network, two node disjoint paths were required for each commodity. Additionally, physical hop constraints were imposed on all flows.

Finally, we point out that a slightly different form of a two-layer network design appears in ATM networks. This problem has been studied by Ball and Vakhutinsky [3, 4]. However, the two-layer network design problem studied in [3, 4] differs from the one studied in this dissertation in several important aspects. First, the number and the capacity of logical paths is only limited by the total capacity of physical links used. This is an important aspect of the problem, since it implies that the logical path variables can be defined as general integers, meaning that there is no need to generate multiple lightpaths with the identical propagation path in the physical layer. They also allow bifurcation of flow and do not consider any node equipment requirements. On the other hand, traffic requests in the model of Ball and Vakhutinsky are grouped into classes that have the same origin and destination, and the same peak and mean bit rates, which in turn significantly increases the number of commodities and the number of the corresponding flow variables.

For this problem, Ball and Vakhutinsky [3] proposed two models. One for the problem described above, and the other for an extended model, where the network is supposed to be fault tolerant to single link failures. In both models a bicriteria objective that maximizes the overall network throughput and minimizes the amount of traffic flow going through logical path switches was used (the two criteria were expressed as a linear combination of the two objectives, with a use of pre-defined weight for the second objective).

In [3], Ball and Vakhutinsky recognized the computationally hard nature of the problem, and proposed several improvement and simplification strategies, including variables aggregation, solving the linear relaxation instead of the complete MIP formulation, development of heuristic

procedures that would take advantage of the results obtained by solving the linear relaxation, and development of valid inequalities that would strengthen the linear relaxation.

In [4], Ball and Vakhutinsky proposed several classes of valid inequalities that exploit the bandwidth based grouping of traffic requests, and the fact that, by the definition of their problem, at least a portion of all traffic requests has to be served.

They proposed a two-step procedure to solve this problem. In the first step, they solve a multicommodity flow problem and use the result to identify a *single* logical path for each pair of nodes in the network. For the identified sets of logical paths, they then iteratively solve the linear relaxation of the original MIP formulation using a different weight for the second objective in each iteration. Within each iteration, a rounding heuristic is applied several times to a solution found in the previous step (each time a different rounding threshold is used). The values of logical path variables found using the rounding heuristic are then fixed in the original MIP formulation, which is then solved using commercial optimization software CPLEX. The final solution is determined as the one that is best out of all threshold values. Also, based on all the different values of weight for the objective, a subset of non-dominated solutions is selected.

In their computational tests on two network configurations, a grid network with 8 nodes and 10 undirected links, and a geographically distributed network with 12 nodes and 20 undirected links, the proposed procedure provided results that had an integrality gap between 1.86% and 10.53%. They also found that the addition of a second logical path for each pair of nodes improves the quality of results, but not significantly.

We will next discuss existing solution approaches for the WDM optical network design problems that are more closely related to the problem studied in this dissertation.

## 5.6 Existing solution approaches for the design of WDM optical networks

Since the focus of this dissertation is on WDM optical networks with wavelength changers, we will provide a review of only those procedures that are relevant for the problem studied in this dissertation. A more comprehensive survey of the existing solution approaches for the WDM optical network design under different network settings can be found in the survey by Dutta and



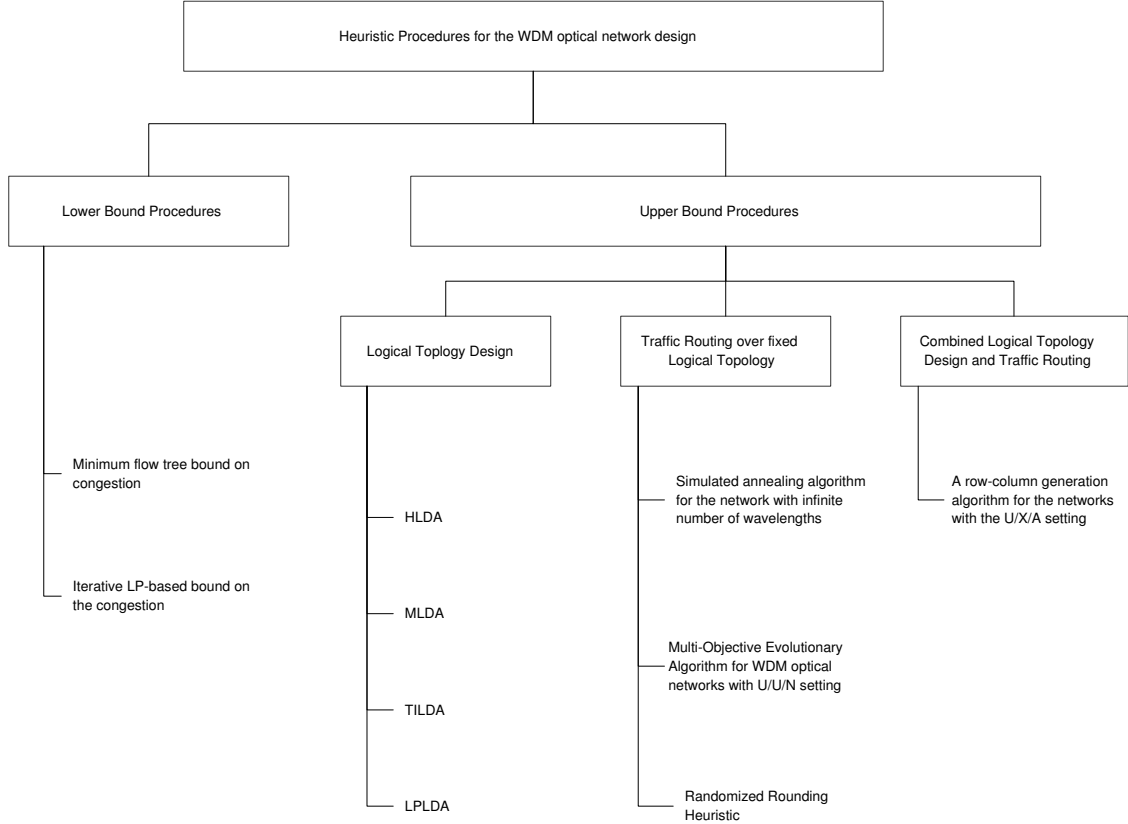


Figure 5.4: Heuristic Procedures for the WDM optical network design problem

Rouskas [21]. Our review is organized in three parts. The first and the second part review lower and upper bound procedures respectively, and the last part reviews exact solution procedures. A list of all heuristics and exact solution procedures reviewed in this chapter is provided in Figures 5.4 and 5.5 respectively.

### 5.6.1 Lower Bound Procedures

The lower bound procedures reviewed in this section provide a lower bound on the network congestion. These bounds can be used to evaluate the quality of the upper bound procedures when the network design objective is to minimize the congestion (defined as maximum load on any given lightpath in the network).

1. **Minimum Flow Tree Bound (MFTB).** This lower bound was proposed by Ramaswami and Sivaraman in [54], and it applies to any logical topology with maximal degree  $\Delta_l$  and states that the minimum congestion in the network satisfies the following inequality:

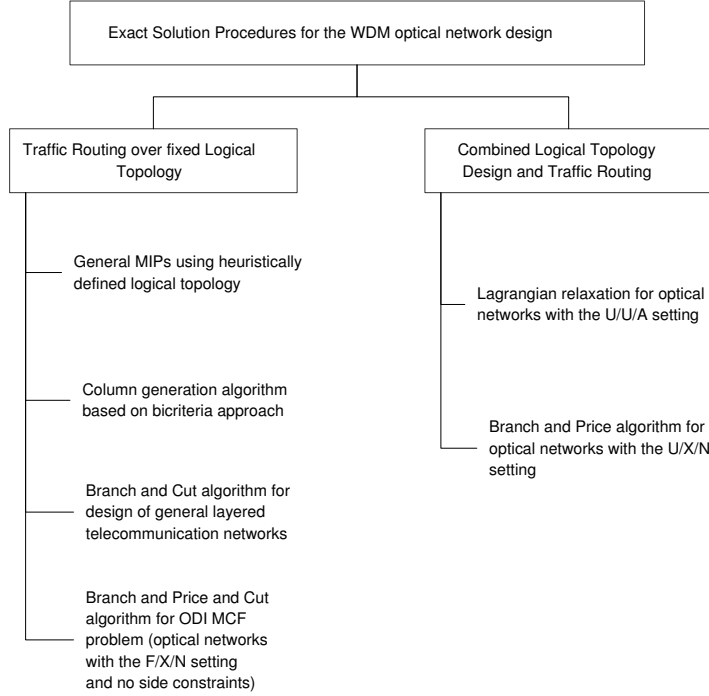


Figure 5.5: Exact Procedures for the WDM optical network design problem

$$\lambda_{max} \geq r\bar{H}_{min}/E_l$$

where  $\lambda_{max}$  represents the minimum possible congestion in the network,  $r$  represents the total arrival rate of traffic to the network, and  $E_l$  represents the number of logical links in the network.  $\bar{H}_{min}$  represents a lower bound on the average number of logical hops in the network. (Calculation of this lower bound is based on the observation that for each source node  $i$ , the number of one-hop destinations is limited by the number of transmitters at node  $i$ . Similarly, the number of two-hop destinations is limited by the square of the number of transmitters at node  $i$  etc.)

Computational experiments in [54] indicate that the MFTB represents a very weak lower bound when applied to the WDM networks with infinite number of wavelengths and full wavelength conversion. For the five instances of the NSFNET example, where the optimal solution was known, the MFTB had an average gap of 64.64% from the optimal solution value. This lower bound has been, however, used in two iterative LP procedures, which provide stronger lower bounds on the congestion. We describe these procedures next.

2. **Iterative LP procedure for the lower bound on the congestion.** This procedure was proposed by Ramaswami and Sivarajan [54] for WDM optical networks with infinite number of wavelengths and allowed flow bifurcation. It is based on the use of a simple cutting plane (originally defined by Bienstock and Günlück [9]) to iteratively solve the LP relaxation defined in [54]. In each iteration, the objective value from the previous iteration is used to make an update on the congestion. Using the notation for the MIP-ARC-LOCAL formulation from the previous section, the cutting plane added is of the following form:

$$\lambda_{max} \geq \sum_{(s,d) \in \Omega} f_{(i,j);k}^{(s,d)} + \lambda_{max}^{(L)} (1 - Y^{(i,j);k}) \quad \forall (i,j) \in \Lambda, k \in \kappa_{(i,j)} \quad (5.29)$$

where  $\lambda_{max}$  is the objective minimized in a current iteration, and  $\lambda_{max}^{(L)}$  is the lower bound obtained in the previous iteration (i.e., the previous objective). Note that this cutting plane may be helpful only in instances where global lightpath variables have fractional values.

Computational results for two sets of instances with different traffic patterns for the NSFNET example, indicated that this procedure provided results 0% - 5.19% from the best known upper bound, with an average gap of 0.75%. For a sparse traffic matrix, this procedure provided results 0% - 18.79% from the best known upper bound and the average gap of 4.27%. Overall, the gap from the MIP upper bound was in range 4.57% - 39.68% and 15.80% on average.

Observe that the cutting plane (5.29) can be used in optical networks with different settings. It is, for example, valid both for networks with and without bifurcation of flow, as well as, finite and infinite number of wavelengths. Additionally, it can be used in networks without wavelength conversion. For example, Krishnaswamy and Sivarajan [37] have used the same type of cutting plane and lower bound procedure for WDM optical networks with finite number of wavelengths and without wavelength conversion. In their study, this procedure provided the following results. On a 6 node network this procedure provided an average gap of 5.58%, when compared to optimal solution. For two different traffic patterns of the NSFNET network the average gap between the lower and the best known upper bound was

10.87%.

### 5.6.2 Upper Bound Procedures

In this section, we review upper-bound heuristics for pre-specified and arbitrary logical topologies. Again, heuristics for networks without wavelength conversion are not included, as these networks are not studied in this dissertation. The upper bound heuristics reviewed here are classified based on the type of the subproblem of the WDM optical network design for which they are defined, and include heuristics for: LTD problem, traffic routing problem with fixed LTD, and complete WDM optical network design problem (i.e., the simultaneous solution of the LTD and lightpath routing problem).

Heuristics for the LTD problem are useful for two reasons. First, they can be used to define a fixed logical topology, which, further, can be used to obtain potentially stronger upper bounds. This can be achieved by using the logical topology determined by one of the LTD algorithms as input for the MIP that solves the traffic routing problem over the fixed logical topology. Second, logical topologies determined using these heuristics can be used as a source of good lightpaths that could be used in a column generation framework. We review four such procedures that were developed by Ramaswami and Sivaraman [54].

1. **HLDA** - Heuristic Logical Topology Design Algorithm that minimizes congestion. This algorithm first attempts to establish one-hop lightpaths for commodities with the largest traffic. Once no additional one-hop lightpaths can be created, the traffic for remaining commodities is routed using residual capacity on the already established lightpaths.
2. **MLDA** - Minimum-delay Logical Topology Design Algorithm. This algorithm is used in a special case when the number of transceivers is larger than the degree of the physical topology. The algorithm starts by adding as many direct (one-hop) lightpaths as possible, and then implements steps identical to HLDA.
3. **TILDA** - Traffic Independent Logical Topology Design Algorithm. This algorithm is designed for situations where the goal is to minimize the number of wavelengths required. It

starts by establishing lightpaths between all one-hop neighbors in the physical topology, then the two-hop neighbors, and so on.

4. **LPLDA** - LP Relaxation Logical Topology Design Algorithm. This algorithm is based on the rounding procedure applied to the solution of the iterative LP-relaxation algorithm presented in [54]. Computational experiments performed on the NSFNET network for 2 different traffic patterns indicated that MLDA and LPLDA provide the best results.

Upper bounding procedures for the traffic routing in networks with fixed logical topology design:

1. **Iterative Simulated Annealing Algorithm.** Chang et al. [11] developed an iterative procedure for the low earth orbit satellite networks (as mentioned before the design of LEO satellite networks is similar to the design of the WDM optical networks with infinite number of wavelengths, wavelength conversion, and bifurcation of flow allowed). The simulated annealing part of the procedure is used to perform updates of the logical topology in terms of the edges available for use. In the second part of the procedure, the traffic grooming problem is solved. Both parts are repeated until termination criteria are met. We have classified this heuristic as the one with the fixed logical topology, because the problem of lightpath routing is not solved here (recall that when we have infinite number of wavelengths there is no need to solve the lightpath routing problem).
2. **Multi-Objective Evolutionary Algorithm - MOEA.** Prathombutr et al. [50] studied the design of the WDM optical network with SONET/SDH streams with and without wavelength conversion, and with the  $U/U/N$  setting. The proposed solution procedure works with a reduced search space in the sense that only a subset of all possible lightpaths can ever be considered for the network design. Prathombutr et al. consider three different objective functions: traffic throughput, number of transceivers, and average propagation delay. Traffic demand is represented in terms of the OC-x traffic requests, and demand between any two nodes may consist of many requests of different size. While all requests for a given source destination pair do not need to be routed over a unique path, a single request must follow

(1, 2)		(1, 3)		(2, 1)		(2, 3)		(3, 1)		(3, 2)	
1	0	0	2	1	0	2	1	1	2	0	1

Figure 5.6: An example of the chromosome representation used in the Multi-Objective Evolutionary Algorithm [50]

a unique path. The set of possible paths is calculated in advance either as K-shortest paths in the physical topology, or as randomly generated paths. The proposed algorithm is a multi-objective evolutionary algorithm (MOEA), in which each chromosome represents a virtual topology encoded by a  $|V| * |V - 1| * |\Delta_i|$  array of numbers. Each gene is defined by  $|\Delta_i|$  numbers (path indices), each representing an index of the lightpath using a given transmitter. For example, the chromosome in Figure 5.6, represents the virtual topology of a 3 node network with 2 transmitters available at each node. So, it has 6 genes, one for each pair of nodes. The first gene represents lightpaths between nodes 1 and 2, and indicates that the first transmitter at node 1 is used by lightpath 1, while the second transmitter is not used for any lightpaths between nodes 1 and 2. Similarly, the second gene represents the lightpaths between nodes 1 and 3, and indicates that the first transmitter from node 1 is not used for any lightpaths between nodes 1 and 3, while the second transmitter is used for lightpath 2. If the generated chromosome violates any of the degree constraints, lightpaths carrying the least amount of traffic are removed one by one until all degree constraints are satisfied. Traffic routing and wavelength assignment (in case of networks without wavelength conversion) is then performed using heuristic algorithms. Specifically, traffic routing is performed using the shortest path algorithm with priority given to the larger traffic streams and single-hop paths. Each chromosome is evaluated based on three objective functions, and is then heuristically compared to other chromosomes during the selection stage of the algorithm. Selection is followed by crossover and mutation. Computational experiments performed on a 6-node network and the NSFNET network for a single set of traffic demands, and varying number of transmitters and receivers, and number of wavelengths available at each fiber, indicate that MOEA outperforms two heuristics presented by Zhu and Mukherjee [62].

### 3. Randomized rounding

This heuristic was developed by Banerjee and Mukherjee [5]. In this procedure, it is assumed that only origins and destinations of the lightpaths are pre-specified. The randomized rounding is part of a more complex procedure that determines both the lightpath routing and the wavelength assignment in optical networks without wavelength changers. However, the randomized rounding heuristic can be used as a stand-alone procedure for the networks with wavelength changers. This heuristic is based on a three-phase approach where first an LP relaxation of the multicommodity flow problem is solved. The fractional solution found in the first phase is then used to identify a set of paths that could be used to serve each commodity. This stage (referred to as **path stripping**) is then followed by a **randomization** phase where one of the paths identified in the path stripping phase is randomly selected using path weights as selection probabilities (Each path in this set is assigned a weight that is based on the flow over edges that constitute the path.)

Upper bounding procedures for the combined (simultaneous) logical topology design and lightpath routing (complete WDM optical network design) are not common in literature. However, one such procedure has been recently proposed by Belotti and Malucelli [8]. They propose a column generation algorithm for the two-layer telecommunication networks with the following setting. Upper layer demand needs to be routed over the paths of the lower layer, which are routed in the physical topology. The only constraints considered are the lower layer path capacity and the capacity of the physical edges in terms of the total flow supported. In case that demands are more than one TU, this procedure allows flow bifurcation. The proposed column generation starts with a subset of lower and higher layer paths and progressively solves the problem. If the solution found by the column generation is fractional a heuristic rounding algorithm is applied in search for a feasible integer solution. The column generation is then repeated using the modified reduced costs that favor use of physical edges not close to its capacity. More details on this procedure are provided in Chapter 6.

### 5.6.3 Exact Solution Procedures

As in the case of upper bounding procedures, we classify the exact solution procedures based on the type of problem solved and include algorithms for: traffic routing problem over fixed LTD, and combined LTD and traffic grooming problem.

We first discuss exact procedures for the traffic routing in the networks with a fixed logical topology design:

#### 1. **General MIPs using heuristically defined logical topology**

These upper bounds are based on the two-step WDM optical network design approach. In the first step, the logical topology is heuristically determined, and then in the second step traffic routing problem is solved using the exact MIP. An example of this approach is the K-shortest path based MIP developed by Banerjee and Mukherjee [6]. This procedure is based on the idea of search space reduction in the sense that only a small subset of all possible lightpaths is considered. For each  $(s, d)$  pair, only those variables that represented lightpath  $(s, d)$  over physical edges contained in the K shortest paths were included in the formulation. Similarly, for each  $(s, d)$  pair only those variables that represented traffic carried between nodes contained in the K shortest paths were included in the formulation. The computational experiments performed in [6] included three networks: 14-node NSFNET network, 15-node PACBELL network, and 20-node randomly generated network. In all tests it was assumed that there is only one fiber on every physical link and that the number of transmitters is equal to the number of receivers and constant across the network.

#### 2. **Column generation algorithm based on a bi-criterion approach.**

This procedure was proposed by Haque et al. in [33], and it solves the WDM design problem in two-stages. First, the logical topology is defined heuristically (using a modification of the HLDA algorithm), and then optimal routing of traffic is performed over the defined logical topology. For the second stage, a column generation algorithm with a bi-criterion approach is developed. Computational experiments in [33] indicate that this approach efficiently solves problems with



up to 50 nodes. In this computational study, Haque et al. compared results of their column generation algorithm for different logical topologies obtained using several different heuristic procedures. They have found that depending on the quality of the logical topology provided as an input, the final result may differ in up to 15% improvement of the network throughput.

### 3. **Branch-and-Cut algorithm for the design of general layered telecommunication networks.**

Dahl et al. (1999) considered a general problem for the layered telecommunication networks. They looked at a two-layer network defined by the physical network  $N = (V, L)$ , and the **pipe** graph  $G = (V, E)$  that consists of a **pre-specified** set of pipes installed over the edges of graph  $N$ . Given the demand matrix, the objective is to route all the traffic, so that each commodity uses exactly one pipe path, and the fixed pipe installation cost and the cost of routing flows over pipes is minimized. In this problem, only a limited number of pipes may be established over any given physical edge, and, there is, also, a limit on the capacity of pipes used. So, the sum of all flows using a given pipe must be less than its capacity. Additionally, it was assumed that the demand of commodities can have only two possible values. The proposed solution procedure is an efficient branch-and-cut algorithm that uses knapsack, strengthened cut, and hypomatchable inequalities (see [15] for details). The computational experiments were performed on problems with up to 62 nodes and 81 edges in the physical graph. These results indicate that when the pipe installation cost is set to very small values, the proposed branch-and-cut algorithm performs quite well (all the test problems were solved to optimality at the root node in quite short CPU times). However, the procedure turned out to be less effective in the case of high pipe installation cost.

### 4. **Branch-and-Price-and-Cut algorithm for the ODIMCF problem corresponding**

**to optical networks with  $F/X/N$  setting and *cap* constraints.** This procedure was developed by Barnhart et al. [7], and is described in more detail in Chapter 6, but basically it represents an exact approach that can be used to solve the network design problem for WDM optical networks with fixed logical topology and *cap* constraints.

We now provide a review of exact solution procedures for the combined logical topology design

and traffic routing problem:

1. **Lagrangian relaxation for the optical networks with  $U/U/A$  setting.** Recently, Farahat et al. [22] considered the following WDM optical network design problem. It is assumed that there is a limited number of fibers available in the network and that the capacity of each fiber is 1 lightpath. As the wavelength assignment problem is not solved here, this is equivalent to the situation where a single fiber is available with a limited number of wavelengths. It is assumed that traffic bifurcation is allowed, and traffic demand is not necessarily symmetric. The direction of fibers in the networks is not known in advance and is determined through optimization. The objective of network design in this case is to minimize the cost of node equipment (transceivers and fiber boxes with cost dependent on the total traffic that needs to be processed at a given node<sup>1</sup>). An initial MIP with traffic flow variables is proposed and modified using aggregation of traffic flow by source node. The solution procedure consists of applying Lagrangian relaxation to constraints on the limited number of fibers, and adding simple Gomory cuts. This solution procedure was tested over a group of 28 problems defined on a network with 13 nodes and 21 physical edges. The effectiveness of the proposed procedure appears to be dependant on the total demand that needs to be served in the network, and gaps between the lower and upper bounds vary between 1.3% and 9.0%, with CPU time limited to 600 seconds (on 1.3 GHZ Pentium III processor and using CPLEX 7.5). The feasibility check (with respect to original constraints on the number of fibers available at each physical link) applied to the best integer solution found solving the Lagrangian relaxation provided feasible solutions in all 28 test problems, indicating optimality of solutions found.
2. **Branch-and-Price algorithm for optical networks with  $U/X/N$  settings.** The branch-and-price algorithm developed by Sung and Song [58] represents the first exact procedure for the simultaneous design of the logical topology and lightpath routing with no bifurcation allowed. However, the procedure is based on some strong simplifying assumptions, including the one that allows only a single lightpath between any two pairs of nodes. Additionally

---

<sup>1</sup>Only traffic on lightpaths originating and terminating at a given node is processed and counted towards the capacity of the installed fiber box

lightpath capacity and degree constraints were not considered either. We will discuss this procedure in greater detail in Chapter 6.

## 5.7 Conclusion

It is clear that solving the entire WDM network design problem represents a significant challenge, but it is also evident that not many exact optimization techniques have been applied to this problem so far. Earlier studies by Haque et al. [33], and Sung and Song [58] indicate that the application of more advanced IP techniques, such as column generation have a significant potential for solving this problem to provable optimality. In the next chapter, we explore further the application and effectiveness of column generation techniques when applied to the WDM optical network design problem. We then propose two branch-and-price algorithms for the WDM optical network design problem and demonstrate the applicability of these procedures for alternative design objectives. Finally, in Chapter 7, we apply our procedures to WDM optical networks with alternative network settings where bifurcation of flow is allowed, and/or where an ample fiber capacity exists, and there is no limit on the number of available wavelengths on the physical links in the network.

## Chapter 6

### Branch-and-Price Algorithms for WDM Optical Network Design

#### 6.1 Introduction

Branch-and-price algorithms are procedures commonly used to solve problems with a large number of variables. They combine the linear programming concept of column generation and integer programming concept of branch-and-bound. Column generation itself is based on a very simple idea. Instead of working with a complete linear programming problem, we can solve the problem that includes only a subset of original variables, and then perform a check to see whether any other variables need to be added to this model in order to find the optimal solution to the original problem. Although column generation is simple in principle, its implementation requires careful consideration of many important issues. We discuss some of these issues and provide a more formal interpretation of column generation in Section 6.2 of this chapter.

The remaining part of this chapter is organized as follows. In Section 6.3, path-based versions of the MIP-ARC-LOCAL and MIP-ARC-GLOBAL formulations presented in Chapter 5 are given. In Section 6.4 we provide a more detailed review of the relevant column generation and branch-and-price algorithms presented in the literature and discuss their relationship to the WDM optical network design problem. In Section 6.5, we propose branch-and-price algorithms for two mathematical formulations for the WDM optical network design problem. We show that depending on our choice of constraints for our formulations, the proposed branch-and-price algorithms can be used to obtain exact solutions, as well as valid lower and upper bounds for this problem. We also discuss the applicability of our procedures to WDM optical networks with alternative design objectives. In Section 6.6, we discuss two classes of valid inequalities that can be used in our branch-and-price framework.

Finally, in Section 6.7, computational results on four different classes of problems are presented.

## 6.2 Review of the Branch-and-Price Algorithms

Column generation is closely related to the Dantzig-Wolfe decomposition principle [16] developed for the large-scale linear programming problems that have decomposable, block-angular structure of the constraint matrix. The Dantzig-Wolfe decomposition principle uses the fact that block-angular constraint matrices can be decomposed into a finite number of smaller constraint blocks, such that each constraint block uses an exclusive set of the original variables. This further allows the reformulation of constraint blocks so that each constraint block can be redefined using a convex combination of its corresponding extreme points. The main advantage of this reformulation approach is that the new formulation contains a significantly lower number of constraints compared to the original one. On the other hand, the number of variables significantly increases. It turns out that the latter issue can be often efficiently resolved through the gradual introduction of variables into the model by following the same logic used to introduce new columns into the basis in the simplex algorithm. The motivation for this strategy is that an optimal solution can often be found by adding only a small number of the variables into the model.

Formally, Dantzig-Wolfe decomposition works with the general linear programming problems of the following form.

$$\text{Min } cx \tag{6.1}$$

subject to:

$$Ax = b \tag{6.2}$$

$$Dx \leq d \tag{6.3}$$

$$x \in R_+^n \tag{6.4}$$

The main assumption in the Dantzig-Wolfe decomposition is that the constraint matrix  $D$  in the above problem has a block-angular structure. As mentioned, for such constraint matrices we can define a finite number of constraint blocks  $D^k$  that together define constraint matrix  $D$ . Once this is done, we can group variables  $x$  that are *active* for each of the constraint blocks  $D^k$  and define

them as the new sets of variables  $x^k$  each of the size  $n_k$  (i.e.,  $\sum_{k=1}^K n_k = n$ , and  $\cup_{k=1}^K x^k = x$ , where  $K$  represents the number of constraint blocks  $D^k$ ). Using the new sets of variables  $x^k$ , we can redefine constraint matrix  $A$ , and the vectors  $c$  and  $d$ , so that they are defined for each  $k$  ( $k = 1, \dots, K$ ). The general linear programming problem (6.1)-(6.4) can be then written as follows.

$$\text{Min} \sum_{k=1}^K c^k x^k \quad (6.5)$$

subject to:

$$\sum_{k=1}^K A^k x^k = b \quad (6.6)$$

$$D^k x^k \leq d^k \quad k = 1, \dots, K \quad (6.7)$$

$$x^k \in R_+^{n_k} \quad k = 1, \dots, K \quad (6.8)$$

In the Dantzig-Wolfe decomposition procedure the linear programming problems that can be decomposed into form (6.5)-(6.8) are reformulated by replacing all variables  $x^k \in X^k$  ( $X^k = \{x^k | D^k x^k \leq d^k, x^k \in R_+^{n_k}\}$ ) by convex combinations of the extreme points of the corresponding set  $X^k$  (for ease of exposition we will describe column generation for the case when the feasible regions  $X^k$  are bounded). In other words, if we let  $y_j^k$  represent the  $j^{th}$  extreme point of the set  $X^k$  ( $j = 1, \dots, J_k$ , where  $J_k$  represents the total number of extreme points for  $X^k$ ), then each set of variables  $x^k \in X^k$  can be defined as:

$$x^k = \sum_j^{J_k} \lambda_j^k y_j^k, \quad \sum_{j=1}^{J_k} \lambda_j^k = 1, \quad \lambda_j^k \geq 0, \quad j = 1, \dots, J_k.$$

The formulation (6.5) - (6.8) can now be written in the following from:

$$\text{Min} \sum_{k=1}^K \sum_{j=1}^{J_k} c^k \lambda_j^k y_j^k \quad (6.9)$$

subject to:

$$\sum_{k=1}^K \sum_{j=1}^{J_k} A^k \lambda_j^k y_j^k = b \quad (6.10)$$

$$\sum_{j=1}^{J_k} \lambda_j^k = 1 \quad \forall k = 1, \dots, K \quad (6.11)$$

$$\lambda_j^k \geq 0 \quad \forall j = 1, \dots, J_k, k = 1, \dots, K \quad (6.12)$$

In the context of Dantzig-Wolfe decomposition, formulation (6.9) - (6.12) is referred to as the **master model** or master formulation. This formulation contains a significantly fewer number of

constraints compared to the original formulation, but has a much larger number of variables (one for each extreme point of each set  $X^k$ ). However, since the optimal solution is the convex combination of only a small subset of all the extreme points of  $X^k$ , we could try to solve a **restricted** version of the problem (6.9) - (6.12) using only a subset of variables each representing an extreme point of a given set  $X^k$ . If the solution of the restricted problem is not optimal to the master problem, we need to add additional extreme points to the restricted master problem. This can be done in the same manner new columns are introduced into the basis in the simplex algorithm. That is, we look at the reduced cost of all variables (including those not in the restricted master problem) and try to find the one with the minimum reduced cost. If the minimum reduced cost turns out to be nonnegative, the current solution is optimal and we can stop the search. Otherwise, we need to introduce column(s) with negative reduced cost and resolve the restricted master problem. The problem of identifying new columns is referred to as **subproblem** or **pricing** problem, and with respect to the original set of variables  $x^k$  is formulated in the following way (for each  $k$ ,  $k = 1, \dots, K$ ):

$$\text{Min} \quad (c^k - \pi A^k)x^k - \mu^k \tag{6.13}$$

$$\text{s.t.} \quad D^k x^k \leq d^k \tag{6.14}$$

$$x^k \in R_+^{n_k} \tag{6.15}$$

where  $(\pi, \mu^k)$  are the optimal values of dual variables (corresponding to constraints (6.10) and (6.11) respectively) obtained by solving the restricted master problem. Notice that each of the  $K$  problems are linear programs, and thus the optimal solution of the above pricing problem is an extreme point of  $X^k$ . If for each  $k$ , expression (6.13) turns out to be nonnegative, it means that the reduced cost of all variables of the master problem is nonnegative, and therefore the current solution is optimal. Otherwise, the solution of the subproblem (in the form of a new variable representing an extreme point) is added to the restricted master model. The procedure is repeated until the optimal solution is found.

The effectiveness of Dantzig-Wolfe decomposition when combined with branch-and-bound to solve integer programs, depends on many implementation issues. Some of these issues in-

clude initialization, strength of the linear relaxation of the master problem, subproblem structure, branching strategies, and termination (see Vanderbeck and Wolsey [61] for a more comprehensive review of some other important issues). The initialization is crucial for two reasons. First, in order to implement column generation it is necessary to start with a feasible initial solution (even if it is a very bad one). Second, we need to be careful with respect to our choice of initial variables as a good set of initial variables could significantly speed up the search for the optimal solution. But, when column generation is used in combination with branch-and-bound (these procedures are referred to as **branch-and-price** algorithms), even the start with a good (or even optimal) set of initial columns, does not guarantee that the search will terminate early. This is directly related to the strength of the linear relaxation of the master problem, an issue that can completely halt progress of any branch-and-price algorithm. So, as in the standard integer programming techniques based on the branch-and-bound method, it is still very important to start with a strong linear relaxation of the master model. However, the number of inequalities that are needed to provide a strong linear relaxation for a given problem may be too large to include all of them in the model. For this reason, it is not uncommon to generate necessary inequalities only when needed as in the standard cutting plane algorithms. (Algorithms that combine column generation with dynamic cut generation for the solution of integer (or mixed integer) programs are called **branch-and-price-and-cut algorithms**.) And, just as branching combined with column generation is a non-trivial task, the addition of cutting planes may be much more complex when done together with column generation. Branch-and-price and branch-and-price-and-cut algorithms also require a feasible restricted master problem at each node of the branch-and-bound tree. As indicated by Vanderbeck [60, 61], this issue can be resolved through the use of high cost artificial variables(s). However, the choice of these artificial variables and the choice of their coefficients in the objective function and the constraint matrix, may influence the computational performance of the underlying branch-and-price algorithm.

The subproblem structure has a great importance in column generation, since, in some cases, the subproblem may turn out to be NP-hard problem itself, which then additionally complicates



the column generation procedure. For this reason, we should try to formulate the master problem in such a way that the calculation of the reduced cost or solving the subproblem is relatively easy.

In the case of branch-and-price algorithms, we also need to look for a convenient form of the subproblem, so that branching does not become too complicated. As mentioned, branching in a column generation framework is significantly more sensitive issue than in the standard branch-and-bound framework. That is, when used in combination with column generation, branching decisions must be such that the structure of the subproblem is not destroyed in subsequent iterations. We will explain these issues in more detail in our discussion of column generation and branch-and-price algorithms for the WDM optical network design problem.

The arc-based formulations for the WDM optical network design problem presented in the previous chapter have a general form where feasible solutions to the original problem are not defined explicitly (that is, each path in the physical, or logical layer is specified using multiple variables and constraints). We could, however, consider the use of a single variable for each specific path for logical layer, and a single variable for each specific flow path for each commodity. This leads to the idea of reformulating the arc-based formulations as path-based formulations, an approach that is commonly used for multicommodity flow problems.

In the next section we will present two path based formulations corresponding to the MIP-ARC-LOCAL and MIP-ARC-GLOBAL formulations presented in Chapter 5. We will then review and discuss relevant column generation and branch-and-price algorithms for the WDM optical network design problem.

### 6.3 Path Based Mathematical Formulations for the WDM Optical Network Design Problem

The notation used in the path-based formulations presented in this chapter is identical to notation used for the arc-based formulations presented in Chapter 5, except for the definition of variables, which will be specified separately for each formulation. Additionally, we will refer to the lightpaths with known paths in the physical topology as the **local lightpaths**. The lightpaths with unspecified paths in the physical layer (that is, where we only know origins and destinations

of the lightpaths) will be referred to as **global lightpaths**. We will also use the following notation for all possible lightpaths and flow paths that can be set up in a network.

$Z$  - Set of all possible lightpaths

$P^{(s,d)}$  - Set of all possible flow paths  $p$  for commodity  $(s, d)$

### 6.3.1 MIP-PATH-LOCAL Formulation

#### Variables

$X_z$  - local lightpath indicator variable; indicates whether lightpath  $z$  is used in a given solution.

$f_p^{(s,d)}$  - flow path indicator variable; indicates whether flow path  $p$  is used to carry traffic demand for commodity  $(s, d)$

$H^{(s,d)}$  - lost traffic indicator variable; indicates whether demand of commodity  $(s, d)$  is lost or satisfied.

$$\text{Min} \quad \sum_{(s,d) \in \Omega} T^{(s,d)} H^{(s,d)} \quad (6.16)$$

Subject to:

$$\sum_{z:O(z)=i} X_z \leq \Delta_t^i \quad \forall i \in V \quad (6.17)$$

$$\sum_{z:D(z)=j} X_z \leq \Delta_r^j \quad \forall j \in V \quad (6.18)$$

$$\sum_{z:(l,m) \in z} X_z \leq L_{lm} \quad \forall (l, m) \in A \quad (6.19)$$

$$X_z - \sum_{p:z \in p} f_p^{(s,d)} \geq 0 \quad \forall z \in Z, (s, d) \in \Omega \quad (6.20)$$

$$X_z - \sum_{(s,d) \in \Omega, p:z \in p} T^{(s,d)} f_p^{(s,d)} \geq 0 \quad \forall z \in Z \quad (6.21)$$

$$\sum_{p \in P^{(s,d)}} f_p^{(s,d)} + H^{(s,d)} = 1 \quad \forall (s, d) \in \Omega \quad (6.22)$$

$$f_p^{(s,d)} \in B^1 \quad \forall p \in P^{(s,d)}, (s, d) \in \Omega \quad (6.23)$$

$$H^{(s,d)} \in R^1 \quad \forall (s, d) \in \Omega \quad (6.24)$$

$$0 \leq X_z \leq 1 \quad \forall z \in Z \quad (6.25)$$

Constraints (6.17) and (6.18) limit out/in degree of any node not to be larger than the total number of transmitters/receivers. Constraint set (6.19) limits the number of lightpaths that can be carried

over a fiber. As before, we treat multiple fibers between nodes as a single fiber with the number of wavelengths available equal to the number of fibers multiplied by the number of wavelengths available at each fiber. The next constraint set, (6.20), ensures that the traffic of any commodity can be sent on lightpath  $z$  only if that lightpath exists. Constraint set (6.21) ensures that the total traffic over lightpath  $z$  cannot exceed the total capacity of that lightpath. Finally, constraint set (6.22) ensures that either all the demand for a given commodity is satisfied, or is entirely lost.

### 6.3.2 MIP-PATH-GLOBAL Formulation

#### Variables

$Y^{(i,j)}$  - global lightpath variable; indicates the number of lightpaths established between nodes  $i$  and  $j$ . Note that, as in the case of the MIP-ARC-GLOBAL formulation, these variables do not provide information on how the lightpaths of a given origin and destination are routed over the physical topology.

$X_z^{(i,j)}$  - local lightpath variable; indicates the number of lightpaths  $z$  used in a given solution.

$f_p^{(s,d)}$  - flow path indicator variable; indicates whether flow path  $p$  is used to carry traffic demand for commodity  $(s, d)$ . In this formulation, flow paths are defined over the global lightpaths (i.e., we do not determine the exact propagation path of a given commodity in the physical topology).

$H^{(s,d)}$  - lost traffic indicator variable; indicates whether demand of commodity  $(s, d)$  is lost or satisfied.

$$\text{Min} \sum_{(s,d) \in \Omega} T^{(s,d)} H^{(s,d)} \quad (6.26)$$

Subject to:

$$\sum_{j:(i,j) \in \Lambda} Y^{(i,j)} \leq \Delta_t^i \quad \forall i \in V \quad (6.27)$$

$$\sum_{i:(i,j) \in \Lambda} Y^{(i,j)} \leq \Delta_r^j \quad \forall j \in V \quad (6.28)$$

$$Y^{(i,j)} - \sum_{z:O(z)=i,D(z)=j} X_z^{(i,j)} = 0 \quad \forall (i,j) \in \Lambda \quad (6.29)$$

$$\sum_{(i,j) \in \Lambda, z:(l,m) \in z} X_z^{(i,j)} \leq L_{lm} \quad \forall (l,m) \in A \quad (6.30)$$

$$Y^{(i,j)} - \sum_{p:(i,j) \in p} f_p^{(s,d)} \geq 0 \quad \forall (i,j) \in \Lambda, (s,d) \in \Omega \quad (6.31)$$

$$Y^{(i,j)} - \sum_{(s,d) \in \Omega, p: (i,j) \in p} T^{(s,d)} f_p^{(s,d)} \geq 0 \quad \forall (i,j) \in \Lambda \quad (6.32)$$

$$\sum_{p \in P^{(s,d)}} f_p^{(s,d)} + H^{(s,d)} = 1 \quad \forall (s,d) \in \Omega \quad (6.33)$$

$$Y^{(i,j)} - B_h^{(i,j)} S_h^{(i,j)} \geq 0 \quad \forall (i,j) \in \Omega, h \in H \quad (6.34)$$

$$C_h - \sum_{(s,d) \in h, p: (i,j) \in p} f_p^{(s,d)} + S_h^{(i,j)} \geq 1 \quad \forall (i,j) \in \Lambda, h \in H \quad (6.35)$$

$$S_h^{(i,j)} - f_p^{(s,d)} \leq 0 \quad \forall (i,j) \in p, h \in H, (s,d) \in h \quad (6.36)$$

$$S_h^{(i,j)} \in B^1 \quad \forall (i,j) \in \Lambda, h \in H \quad (6.37)$$

$$f_p^{(s,d)} \in B^1 \quad \forall p \in P^{(s,d)}, (s,d) \in \Omega \quad (6.38)$$

$$Y^{(i,j)} \in R_+^1 \quad \forall (i,j) \in \Lambda \quad (6.39)$$

$$X_z^{(i,j)} \in Z_+^1 \quad \forall z \in Z, (i,j) \in \Lambda \quad (6.40)$$

Constraints in the MIP-PATH-GLOBAL formulation have a similar interpretation to those in the MIP-PATH-LOCAL formulation. Constraints (6.27) and (6.28) limit the out/in degree of any node to be not larger than the total number of transmitters/receivers. Constraint set (6.29) ensures that all global lightpaths are defined in the physical topology through adequate number of the local lightpaths in the physical topology. Constraint set (6.30) represents a limit on the number of lightpaths that can be established on any physical edge. Constraint set (6.31) ensures that the flow path of any given commodity  $(s, d)$  can use global lightpath  $(i, j)$  only if that lightpath is included in the logical topology. Constraints (6.32) are capacity constraints limiting total flow over all global lightpaths established between two nodes. Constraint set (6.33) ensures that either all demand for a given commodity is satisfied, or is entirely lost. Constraints (6.34)-(6.37) are packing constraints that ensure no bifurcation of flow among lightpaths with the same origin and destination. These constraints have the same interpretation as the *pack* constraints (5.25)-(5.28) defined in Chapter 5 for the MIP-ARC-GLOBAL formulation. The only difference is that the flow variables are now path-based.

## 6.4 Review of Column Generation and Branch-and-Price Algorithms for the WDM Optical Network Design Problem

As mentioned previously, the WDM optical network design problem is a more general case of the well known and well studied ODIMCF problem. (Recall that ODIMCF problem can be seen as optical network with  $F/F/N$  setting with at most one lightpath defined between pairs of nodes, and *cap* constraints.) This problem has a much simpler form than our path-based models for the WDM optical network design problem presented in the previous section, and it can be formulated in the following way.

$$\text{Min} \quad \sum_{(s,d) \in \Omega, p \in P^{(s,d)}} \sum_{(l,m) \in p} T^{(s,d)} c_{(l,m)}^{(s,d)} f_p^{(s,d)} \quad (6.41)$$

$$\sum_{(s,d) \in \Omega, p: (l,m) \in p} T^{(s,d)} f_p^{(s,d)} \leq L_{lm} \quad \forall (l,m) \in A \quad (6.42)$$

$$\sum_{p \in P^{(s,d)}} f_p^{(s,d)} = 1 \quad \forall (s,d) \in \Omega \quad (6.43)$$

$$f_p^{(s,d)} \in B^1 \quad \forall p \in P^{(s,d)}, (s,d) \in \Omega \quad (6.44)$$

The term  $L_{lm}$  represents capacity of arc  $(l, m)$ , and the term  $c_{(l,m)}^{(s,d)}$  represents cost associated with the use of the arc  $(l, m)$  by the commodity  $(s, d)$ . Barnhart et al. [7] developed an efficient branch-and-price algorithm for this problem that works as follows. The search is started with only a subset of flow paths  $f_p^{(s,d)}$ . Potential new columns are identified by solving the subproblem, which in this case is a polynomially solvable shortest path problem defined separately for each commodity. (For a given commodity  $(s, d)$  this problem is defined over a network with arc costs equal to  $c_{lm}^{(s,d)} - v_{lm}$ , for each arc  $(l, m)$ , where  $v_{lm}$  represents the dual variable corresponding to constraint (6.42)).

An important and new aspect of the algorithm developed by Barnhart et al. is the branching strategy that efficiently prunes the search space. Specifically, at each node of the branch-and-bound tree, two new nodes are added by first identifying the commodity with the greatest total demand that has a fractional flow. Starting from the origin of this commodity, the flow is traced to the first divergence node (note that since the flow is fractional, it follows that demand for this commodity is sent over two or more different paths and will diverge somewhere in the network). The two arcs diverging out of this node with greatest fractional flow of the commodity are identified. Two

subsets of arcs (of approximately same size) with origins at the located divergence node, and each containing exactly one of the two arcs with greatest fractional flow are then defined. At the first child node, a restriction that the given commodity cannot use any of the arcs in the first selected subset of arcs is imposed, and at the second child node, restriction that a given commodity cannot use any of the arcs in the second selected subset of arcs is imposed. A summary of this procedure is provided in Figure 6.1.

An important property of this strategy is that it does not destroy the structure of the subproblem in the subsequent iterations, that is the subproblem remains to be a shortest path problem. Just to get a better understanding of the importance of this property, consider what would happen if we were to perform standard branching based on the variable dichotomy. That is, we could identify a single fractional flow path variable, and on one branch we could decide that a given path cannot be used and on the other that it should always be used. The second case is easy to enforce in the pricing problem, as we would not need to consider any new paths for a given commodity (i.e., since  $f_p^{(s,d)} = 1$ , there is no further need for pricing for a given commodity  $(s, d)$ ). However, the first restriction would be very hard to implement in the pricing problem, as there is no guarantee that the same path would not be generated in the subsequent iterations of column generation. The only way to ensure that we are not going to introduce the forbidden path is to perform a check of the shortest path found for a given commodity. If we find that the generated path is not supposed to be introduced into model again, we would have to solve the second shortest path problem. (Of course, as the number of the restrictions of this type increases, the chance that we will have to solve second, third, ... shortest path algorithm increases as well.)

This issue is not uncommon for column generation, so it is often preferred to perform branching on variables from the corresponding more extensive formulation where solutions are implicitly defined by sets of constraints rather than being explicitly defined using a single variable for each feasible solution of the original problem. In the case of the ODIMCF problem this would correspond to the use of the arc-based flow variables, instead of path-based variables. So, if we were to use the arc-based variables for branching decisions, we could apply the following branching

rules. On one branch we could forbid a given commodity to use a specific arc, and on the second branch we would force the same commodity to always use that arc. In this case, ensuring that the given commodity does not use a given arc is easy to implement, as we can set a high cost on the use of that arc in the shortest path algorithm. However, the requirement to use an arc in a given solution leads to shortest path problem with a side constraint, which makes the subproblem much harder to solve.

The idea of defining two subsets of arcs presented by Barnhart et al. [7] has two important benefits. First, it is easy to implement in the subproblem (we simply set high costs on arcs that are forbidden), and it does not destroy its structure. And, second, eliminating groups of arcs vs. single arc in one branch prunes a larger portion of the solution space.

To relate these ideas to the WDM optical network design problem we can look at the special case of the WDM optical network design problem with  $U/U/N$  setting, infinite number of wavelengths, and up to one lightpath between any two nodes. Now, given the assumption that all nodes in the network are equipped with wavelength changers **and** that there is no restriction on the number of wavelengths available at each fiber, we can replace all logical paths by arcs and formulate the WDM optical network design problem as an integer multicommodity flow problem on a complete graph<sup>1</sup> with a constraint on node degrees. Notice that, due to additional constraints on node degrees, this problem represents a more complex, generalized version of the ODIMCF problem.

Using the notation from the previous section, the **path** based formulation for this problem can be stated in the following way.

$$\text{Min} \quad \sum_{(s,d) \in \Omega} T^{(s,d)} H^{(s,d)} \quad (6.45)$$

Subject to:

$$\sum_{j:(i,j) \in \Lambda} Y^{(i,j)} \leq \Delta_i^i \quad \forall i \in V \quad (6.46)$$

---

<sup>1</sup>Arcs in this graph are actually lightpaths, which (if we assume that the underlying physical topology is a connected graph) implies that lightpaths could be established between any two nodes in the network, and therefore the logical topology is a complete graph.

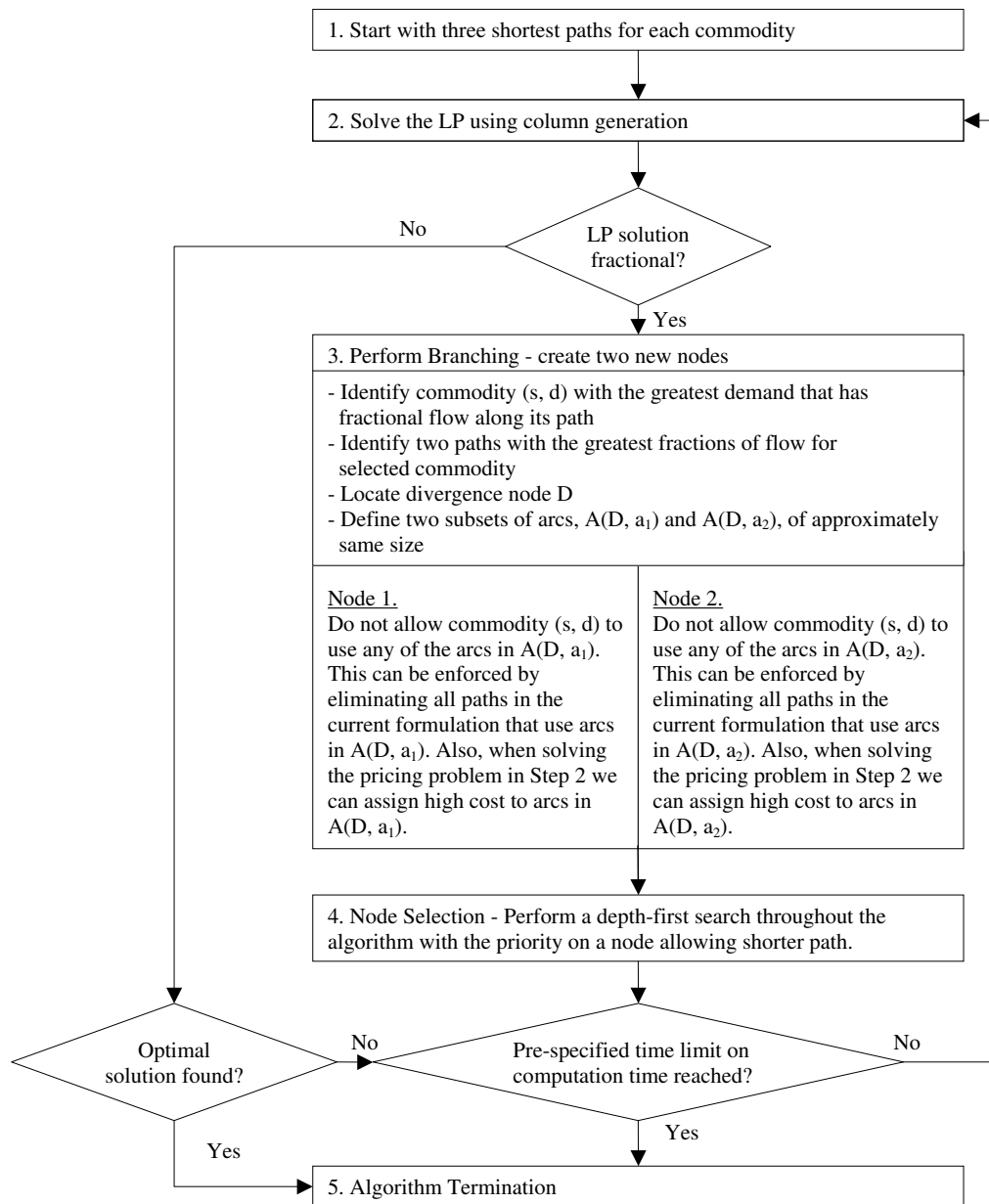


Figure 6.1: Branch-and-Price Algorithm for the ODIMCF problem (Barnhart et al., 2000)



$$\sum_{i:(i,j) \in \Lambda} Y^{(i,j)} \leq \Delta_r^j \quad \forall j \in V \quad (6.47)$$

$$Y^{(i,j)} - \sum_{p:(i,j) \in p} f_p^{(s,d)} \geq 0 \quad \forall (i,j) \in \Lambda, (s,d) \in \Omega \quad (6.48)$$

$$Y^{(i,j)} - \sum_{(s,d) \in \Omega, p:(i,j) \in p} T^{(s,d)} f_p^{(s,d)} \geq 0 \quad \forall (i,j) \in \Lambda \quad (6.49)$$

$$\sum_{p \in P^{(s,d)}} f_p^{(s,d)} + H^{(s,d)} = 1 \quad \forall (s,d) \in \Omega \quad (6.50)$$

$$Y^{(i,j)} \leq 1 \quad \forall (i,j) \in \Lambda \quad (6.51)$$

$$f_p^{(s,d)} \in B^1 \quad \forall p \in P^{(s,d)}, (s,d) \in \Omega \quad (6.52)$$

Note that only the  $f_p^{(s,d)}$  variables need to be defined as integer (binary) variables in this formulation. The  $Y^{(i,j)}$  variables are guaranteed to be integer by constraints (6.48) and (6.51), while integrality of the variables  $H^{(s,d)}$  follows from the constraint (6.50).

We will now explain how the ideas used in the branch-and-price algorithm developed by Barnhart et al. for the ODIMCF problem can be used to solve this special case of the WDM optical network design problem.

#### 6.4.1 Branch-and-Price Algorithm for the Special Case of the WDM Optical Network Design Problem with Infinite Number of Wavelengths

As in the case of the branch-and-price algorithm for the ODIMCF problem, we start the search with only a subset of paths for each commodity. We then solve the pricing problem at the root node of the branch-and-bound tree using the following procedure.

If we associate nonnegative dual variables  $r_{(i,j)}^{(s,d)}$  and  $v_{(i,j)}$  with constraints (6.48) and (6.49) respectively, and unrestricted dual variable  $w^{(s,d)}$  with constraint (6.50), then the reduced cost of any  $f_p^{(s,d)}$  variable is defined by:

$$\sum_{(i,j) \in p} (r_{(i,j)}^{(s,d)} + T^{(s,d)} v_{(i,j)}) - w^{(s,d)}$$

To identify the flow variables that have a negative reduced cost we would need to find the variables  $f_p^{(s,d)}$  such that  $\sum_{(i,j) \in p} r_{(i,j)}^{(s,d)} + T^{(s,d)} v_{(i,j)} < w^{(s,d)}$ . One way to do this is to solve the shortest path problem (for each commodity  $(s,d)$ ) over the network with arc costs defined by:  $r_{(i,j)}^{(s,d)} + T^{(s,d)} v_{(i,j)}$ , for all arcs  $(i,j)$ , and add only those variables that correspond to shortest paths that have negative reduced cost. So, in this case, the objective of our subproblem for each commodity can be defined

as:

$$\text{Min} \quad \sum_{(i,j) \in \Lambda} (r_{(i,j)}^{(s,d)} + T^{(s,d)} v_{(i,j)}),$$

subject to the constraint that the selected arcs constitute a path from from the origin to the destination of the given commodity. Again, if at any iteration, there is a commodity  $(s, d)$  such that the above term is less than  $w^{(s,d)}$ , it means that the variable corresponding to the shortest path found has a negative reduced cost, and, therefore, should be introduced to the restricted master problem.

The above column generation algorithm could be applied only to the root node of the branch-and-bound tree, and the resulting formulation could be then solved using a commercial MIP optimizer such as CPLEX. However, while this approach is relatively simple to implement, it also has an important drawback. That is, as we progress in the branch-and-bound tree, we impose more restrictions on the structure of the logical topology, which may require very different paths for traffic routing from the ones added at the root node. This implies that the application of the column generation algorithm only at the root node of the branch-and-bound tree is not guaranteed to provide an optimal solution. This issue can be resolved by developing a branch-and-price algorithm that would allow the application of the column generation algorithm at each node of the branch-and-bound tree. It turns out that, since this special case of the WDM optical network design problem is a generalization of the ODIMCF problem, where only flow variables need to be defined as integers, we can simply use the previously described branching strategy developed in [7] for the ODIMCF problem. The issue is that as we increase the complexity of the WDM optical network design problem, this approach loses its applicability, and the development of more complex pricing strategies and branching rules becomes necessary. To illustrate this problem, we will next discuss in a greater detail several existing exact algorithms that use column generation technique to solve different versions of the WDM optical network design problem.

#### 6.4.2 Existing Column Generation and Branch-and-Price Algorithms for the WDM Optical Network Design Problem

Notice that in the formulation (6.45) - (6.52) for the special case of WDM optical networks with infinite number of wavelengths propagation delay constraints were not included. It turns out that introduction of propagation delay constraints significantly complicates the problem. This increased complexity stems from the fact that propagation delay constraints require knowledge of propagation paths in the physical topology for both lightpaths and flow paths. This further implies that we cannot treat lightpaths as arcs any longer, but, instead, we have to deal with a two-layer network design, where both lightpaths and flow paths are defined as the paths in the network. A version of the WDM optical network design problem with the propagation delay constraints was studied by Haque et al. [33], who developed a column generation based algorithm for the WDM optical networks with  $F/U/A$  setting without wavelength conversion, and with at most one lightpath between any two nodes. (This study was designed for networks without wavelength conversion. However, the proposed procedures can be used in networks with wavelength conversion, by making an assumption that there are a sufficient number of wavelengths on each fiber, so that the wavelength assignment problem can be ignored.) The algorithm of Haque et al. solves the WDM design problem in two-stages. First, one of the heuristics proposed in the paper is used to define the logical topology, and then the optimal routing of traffic over the defined logical topology is performed. For the second stage (the optimal routing of traffic over the defined logical topology), a column generation algorithm is defined. Since the problem of optimal routing in this case is a ODMCF problem (note that bifurcation of flow is allowed) with a constraint on maximum allowable delay for each commodity, the subproblem is a constrained shortest path problem (each path must satisfy delay restrictions). In [33], Haque et al. solve the subproblem using a **bicriteria algorithm** that basically considers the delay constraint in the shortest path algorithm as a second objective (in addition to the original objective that minimizes the average length of all paths used). This bicriteria algorithm works as follows. First, the shortest path problem is solved. Then, a modified version of Dijkstra's algorithm is used to find a path that has the optimal length (determined in

the previous step) and minimum possible delay. If the length of the resulting path satisfies the delay constraint, the optimal solution is found. Otherwise, the shortest path problem is solved by using the delays as arc costs. If this leads to a total delay that is larger than the permissible delay given by the propagation delay constraint, the search is stopped, since this would indicate that the problem is infeasible. Otherwise, the solution found by the modified Dijkstra's algorithm and the solution of the shortest path problem performed in the first step are used to define two starting efficient extreme points of the efficient (Pareto) frontier. The search is continued (through generation of the new efficient extreme points) until the optimal solution to the routing problem is found. Computational experiments performed by Haque et al. indicate that this procedure efficiently solves problems with up to 50 nodes. It was also shown that depending on the quality of logical topology provided as input, the resulting network throughput may be improved by up to 15%.

The first branch-and-price algorithm that simultaneously considers traffic grooming and logical topology design was developed by Sung and Song [58]. The optical network studied in [58] had the  $U/X/N$  setting. That is, there are no restrictions at the nodes in terms of equipment available. Additionally, capacity of lightpaths was not considered to be constraint. It was also assumed that at most one logical path can be established between any two nodes in the logical layer. The last assumption is very important, as all global lightpath variables can be added at the beginning of the algorithm, and the branch-and-price procedure is used only to determine the actual propagation of logical paths in the physical layer, and flow paths over the global lightpaths. This assumption also warrants that only global lightpath variables need to be defined as integers (see [58] for proof), which then significantly simplifies the branching procedure, as branching decisions have no impact on the subproblem solved in column generation. We recognize that this procedure could be extended so that it is applicable for instances where we have multiple edges in the physical layer, and possible multiple lightpaths between pairs of nodes. But, in order to use the model developed in [58], we would have to include all global lightpath variables in advance, as that procedure does not offer a direct way of adding new global lightpath variables. This

may become impractical in instances where we do not know the actual number of transmitters and receivers, and we would have to provide an upper bound on these numbers, so that we can ensure that all the global lightpath variables are added at the beginning of the algorithm. In computational experiments with networks with up to 20 nodes and 64 commodities, the proposed solution procedure found optimal solution in almost all instances in less than 10 minutes of CPU time.

Recently, Belotti and Malucelli (2005) have proposed a column generation algorithm for the two-layer telecommunications networks [8]. The network design problem studied in [8] has two important aspects that differ from the WDM optical network design problem that is studied in this dissertation. First, in [8], there is no limit on the number of lower layer connections that can be supported by a single physical link. This is similar to WDM optical networks with unlimited number of lightpaths. However, the problem studied in [8] does take into account the physical topology through a requirement that all physical links must have sufficient capacity to support the total traffic flow carried over these links. This restriction is much simpler to implement in the column generation framework compared to our requirement regarding the number of lightpaths supported by each link. The reason is that the physical topology constraint in [8] does not include lower layer path variables, so the dual variables corresponding to this constraint can be taken into account in a straightforward manner when calculating the reduced cost of the higher level (flow) path variables (the issue related to the calculation of the reduced cost when the lower layer path variables are included in the physical topology constraints will become more clear in the next section after we present our branch-and-price algorithm for the MIP-PATH-LOCAL formulation). We note here that the pricing algorithm proposed by Belotti and Malucelli is not entirely correct, as they underestimate the cost of use of the existing lower level paths. However, this does not invalidate the optimality of the solution found by their column generation algorithm, but it may cause the addition of a large number of unnecessary variables. The second simplifying aspect of the problem studied in [8] is that it does not take into account node degree constraints.

In order to find feasible integer solutions, Belotti and Malucelli defined a heuristic rounding

procedure that they apply to solutions found by their column generation algorithm. In order to try to improve the resulting upper bound, their column generation is then repeated using modified reduced costs that favor use of physical edges not close to their capacity, which is then followed by another iteration of the heuristic rounding procedure.

In the next section we will present two new branch-and-price algorithms for the WDM optical network design problem with  $U/U/N$  setting and  $deg/cap$  constraints. We will show that depending on the choice of constraints in the MIP-PATH-LOCAL formulation, our branch-and-price strategy can be used either as an exact solution procedure, or as an efficient upper bound procedure for the WDM optical network design problem.

## 6.5 Branch-and-Price Framework for the WDM Optical Network Design Problem

In this section we discuss our branch-and-price algorithms for the formulations MIP-PATH-LOCAL and MIP-PATH-GLOBAL.

### 6.5.1 Branch-and-Price Algorithm for the MIP-PATH-LOCAL formulation

An important difference between the MIP-PATH-LOCAL formulation for the WDM optical network design and the path based formulation for the ODIMCF problem is that the MIP-PATH-LOCAL formulation contains *two types* of variables, whose number grows exponentially with the network size. These are local lightpath variables  $X_z$  each representing a different lightpath in the physical topology, and flow path variables  $f_p^{(s,d)}$ , each representing a different path for the flow of commodity  $(s, d)$  in the logical topology. Since it is not practical to include all the paths for either of these variables, we can try to dynamically generate both types variables. Having two types of variables in the column generation algorithm is not necessarily a problem, as long as the restricted master problem is row complete (i.e., there are no missing constraints, which are not included in the model simply because we haven't introduced variables that define these constraints). For the row complete formulation, this has already been done in the framework of WDM optical networks (see Sung and Song [58]), and is not much different from performing column generation when we have a single type of variable. However, the omission of lightpath

variables  $X_z$  from the restricted master problem of the MIP-PATH-LOCAL formulation makes the restricted master problem row incomplete (some of the constraints of type (6.20) and (6.21) are not included initially) which significantly complicates column generation. The basic problem is that the standard pricing scheme used in column generation may be of little help when the master problem is row incomplete. The issue can be best explained by looking at the calculation of reduced cost for the lightpath and flow path variables in the MIP-PATH-LOCAL formulation. First, let the following dual variables correspond to the constraints of the MIP-PATH-LOCAL formulation:

- nonpositive  $a_i$  and  $b_j$  variables for constraints (6.17) and (6.18) respectively
- nonpositive  $d_{(l,m)}$  variables for constraint (6.19)
- nonnegative  $r_z^{(s,d)}$  variables for constraint (6.20)
- nonnegative  $v_z$  variables for constraint (6.21)
- unrestricted in sign  $w^{(s,d)}$  variables for constraint (6.22)
- nonpositive  $u_z$  variables for constraint (6.25).

Also, let  $Z^E$  denote the set of all lightpaths that are included in the restricted master problem, and let  $Z^N$  denote the set of all lightpaths that are not included in the restricted master problem. The reduced cost of any lightpath variable  $X_z$  with origin at node  $i$  and destination at node  $j$  is then:

$$RC_z = -a_i - b_j - \sum_{(l,m) \in z} d_{(l,m)} - \sum_{(s,d) \in \Omega} r_z^{(s,d)} - v_z - u_z.$$

Now, if lightpath  $z$  and the corresponding variable  $X_z$  are not already part of the restricted master model, dual variables  $r_z^{(s,d)}$ ,  $v_z$ , and  $u_z$  can be set to zero, as the corresponding constraints are not in the restricted master model. So, the reduced cost  $RC_z$  becomes:

$$RC_z = -a_i - b_j - \sum_{(l,m) \in z} d_{(l,m)}.$$

Given that dual variables  $a_i$ ,  $b_j$ , and  $d_{(l,m)}$  are nonpositive, it follows that the reduced cost of any new lightpath can *never* be negative, and, therefore, it would never be beneficial to add a new

lightpath. While the last statement is obviously not correct (we may very well need to include many new lightpaths to the model), from a computational perspective this is not surprising, since the introduction of a new lightpath without the presence of a flow path that actually uses it has no effect on the objective function. In order to determine whether to introduce a given lightpath in the restricted master model, we should find the **combined reduced cost**<sup>2</sup> for two or more non-basic variables - a non-trivial task that usually does not have an efficient solution, other than resolving the problem with the new variables included in the restricted master model. One approach that we may try for computing of this combined reduced cost is to simply sum up the reduced cost of the new lightpath and the new flow path variable that would use the lightpath(s). To investigate the validity of this idea for the MIP-PATH-LOCAL formulation, we can look at the indirect pricing that takes into account both lightpath and flow path variables using flow path variables only. The idea is as follows.

First, observe that the addition of a **single** new lightpath variable  $X_z$  used by the new flow path variable  $f_p^{(s,d)}$  has the same impact on the linear relaxation of the restricted master model, as the introduction of flow path variable  $f_p^{(s,d)}$  only, where all occurrences of variable  $X_z$  are substituted by variable  $f_p^{(s,d)}$ . This is true since constraint (6.20) forces  $X_z$  to be at least as large as  $f_p^{(s,d)}$ . (To simplify the explanation, but without loss of generality, we can assume that the new  $X_z$  variable only uses one physical edge, and the flow path variable  $f_p^{(s,d)}$  only uses lightpath  $z$ .) After replacing all occurrences of  $X_z$  by  $f_p^{(s,d)}$ , the reduced cost of variable  $f_p^{(s,d)}$  in this case becomes:

$$RC_p^{(s,d)} = -a_s - b_d - d_{(s,d)} - w^{(s,d)}$$

Note that we did not take into account constraints (6.20) and (6.21) here, which is not a problem since these constraints are not yet defined for the new  $X_z$  variable (and therefore the corresponding dual variables can be set to zero). So, in order to take into account possible new lightpaths we never look at the reduced cost of lightpath variables (although the above described procedure is equivalent to summing reduced costs of the new lightpath variables and the new flow

---

<sup>2</sup>We use the term ‘combined reduced cost’ to refer to the change in the objective value caused by introduction of two or more non-basic variables into the basis.



path variables).

In addition to calculating the reduced cost of a flow path variable using new lightpath(s), we have to find a way to determine whether existing or new lightpaths or a combination of both are going to be used for a given flow path. Taking into account the fact that a given flow path may contain both existing and new lightpaths, the reduced cost of any flow path variable becomes:

$$RC_p^{(s,d)} = \sum_{z \in p, z \in Z^E} (r_z^{(s,d)} + T^{(s,d)}v_z) + \sum_{z \in p, z \in Z^N} (-a_{O(z)} - b_{D(z)} - \sum_{(l,m) \in z} d_{(l,m)}) - w^{(s,d)}.$$

Note that for the existing lightpath variables used by the flow path variable  $f_p^{(s,d)}$  we need to take into account only dual variables corresponding to constraints (6.20) and (6.21) (i.e., in this case we simply look at the reduced cost of the variable  $f_p^{(s,d)}$  calculated from the MIP-PATH-LOCAL formulation). In a sense, the term associated with lightpaths that are already included in the restricted master model can be thought of as the price of using existing lightpaths, and, likewise, the term associated with the new lightpaths can be thought of as the price of using the new lightpaths that are not part of the restricted master model yet. So, we can rewrite the reduced cost of the path  $p$  for commodity  $(s, d)$  as:

$$RC_p^{(s,d)} = \sum_{z \in p, z \in Z^E} P_{z;(s,d)}^E + \sum_{z \in p, z \in Z^N} P_{z;(s,d)}^N - w^{(s,d)}$$

where

$$P_{z;(s,d)}^E = r_z^{(s,d)} + T^{(s,d)}v_z, \text{ and}$$

$$P_{z;(s,d)}^N = -a_{O(z)} - b_{D(z)} - \sum_{(l,m) \in z} d_{(l,m)}$$

represent the price of an existing and a new lightpaths (respectively) used by the flow path  $f_p^{(s,d)}$ .

The problem with this approach is that we looked only at the case where the introduction of a new lightpath is justified by its use by a single flow path. It is, however, possible that a new lightpath should be introduced only if multiple flow paths (corresponding to multiple commodities) use that lightpath. To better understand this issue, consider the following formulation derived from the linear relaxation of MIP-PATH-LOCAL by replacing all occurrences of the new new lightpath variables in the linear relaxation of the MIP-PATH-LOCAL formulation by new flow paths  $p'$  that are not already included in the model.

$$\text{Min} \sum_{(s,d) \in \Omega} T^{(s,d)} H^{(s,d)} \quad (6.53)$$

Subject to:

$$\sum_{z:O(z)=i} X_z + \sum_{(s,d) \in \Omega, p': \exists z \in p': O(z)=i, z \in Z^N} f_{p'}^{(s,d)} \leq \Delta_i^i \quad \forall i \in V \quad (6.54)$$

$$\sum_{z:D(z)=j} X_z + \sum_{(s,d) \in \Omega, p': \exists z \in p': D(z)=j, z \in Z^N} f_{p'}^{(s,d)} \leq \Delta_r^j \quad \forall j \in V \quad (6.55)$$

$$\sum_{z:(l,m) \in z} X_z + \sum_{(s,d) \in \Omega, p': \exists z \in p': (l,m) \in z, z \in Z^N} f_{p'}^{(s,d)} \leq L_{lm} \quad \forall (l,m) \in A \quad (6.56)$$

$$X_z - \sum_{p:z \in p} f_p^{(s,d)} \geq 0 \quad \forall z \in Z, (s,d) \in \Omega \quad (6.57)$$

$$X_z - \sum_{(s,d) \in \Omega, p:z \in p} T^{(s,d)} f_p^{(s,d)} \geq 0 \quad \forall z \in Z \quad (6.58)$$

$$\sum_{p \in P^{(s,d)}} f_p^{(s,d)} + H^{(s,d)} + f_{p'}^{(s,d)} = 1 \quad \forall (s,d) \in \Omega \quad (6.59)$$

$$f_p^{(s,d)} \geq 0 \quad \forall p \in P^{(s,d)}, (s,d) \in \Omega \quad (6.60)$$

$$H^{(s,d)} \in R^1 \quad \forall (s,d) \in \Omega \quad (6.61)$$

$$0 \leq X_z \leq 1 \quad \forall z \in Z \quad (6.62)$$

If we were to consider a single commodity  $(s, d)$ , the formulation (6.53) - (6.62), would be equivalent to the linear relaxation of the MIP-PATH-LOCAL formulation, and the previously described approach for calculation of the combined reduced cost is correct. However, consider what happens when the introduction of a single lightpath is beneficial only if multiple flow paths use that lightpath. The above formulation is no longer equivalent to the linear relaxation of the MIP-PATH-LOCAL, since it is possible that the term  $\sum_{(s,d) \in \Omega, p': \exists z \in p': O(z)=i, z \in Z^N} f_{p'}^{(s,d)}$  overestimates the value of a given lightpath variable  $X_z$ . This in turn means that it is possible that our combined reduced cost is overestimated, and thus some useful columns may not be introduced in the restricted master model. Notice that this issue may arise only in situations where multiple new flow paths use the same new lightpath, in which case any  $X_z$  variable takes the following value.

$$X_z = \max\left\{ \max_{(s,d) \in \Omega} \sum_{p:z \in p} f_p^{(s,d)}, \sum_{(s,d) \in \Omega, p:z \in p} T^{(s,d)} f_p^{(s,d)} \right\} \quad (6.63)$$

Therefore, pricing that uses our suggested calculation of the combined reduced cost is not guaranteed to find the optimal solution for the MIP-PATH-LOCAL formulation. However, if we were to use this approach in combination with a valid branching strategy, we would obtain a valid *upper bound* for the original problem. We discuss one such strategy later in this chapter.

To develop a column generation approach that would solve our original problem to optimality we need to find a way to correctly model formulation (6.53) - (6.62), so that this formulation is equivalent to the linear relaxation of the MIP-PATH-LOCAL. We have already mentioned that in the linear relaxation of the MIP-PATH-LOCAL the  $X_z$  variables used by a single commodity take the values defined by constraint (6.20). Since constraint (6.20) uniquely defines values of  $X_z$  variables used by a single commodity, we were able to derive an alternative formulation (6.53)-(6.62) that allows for a simple calculation of reduced costs. In the case where multiple commodities use the same lightpath  $z$ , value of the corresponding lightpath variable  $X_z$  is defined by non-linear equation (6.63). Since equation (6.63) is nonlinear, we cannot simply use it in the MIP-PATH-LOCAL formulation and derive a new, equivalent formulation as we did before. However, if we were to drop constraint (6.20) from MIP-PATH-LOCAL (we will refer to the new formulation as MIP-PATH-LOCALw), then the value of any  $X_z$  variable in the linear relaxation of the MIP-PATH-LOCALw formulation is:

$$X_z = \sum_{(s,d) \in \Omega, p: z \in p} T^{(s,d)} f_p^{(s,d)}.$$

We can now substitute the new expression for  $X_z$  in the MIP-PATH-LOCALw formulation, and use the previously described approach (of determining the combined reduced cost) to develop an exact column generation approach for the WDM optical network design problem. The linear relaxation of the formulation MIP-PATH-LOCALw, now becomes:

$$\text{Min} \sum_{(s,d) \in \Omega} T^{(s,d)} H^{(s,d)} \tag{6.64}$$

Subject to:

$$\sum_{z: O(z)=i} X_z + \sum_{(s,d) \in \Omega, p': \exists z \in p': O(z)=i, z \in Z^N} T^{(s,d)} f_{p'}^{(s,d)} \leq \Delta_i^i \quad \forall i \in V \tag{6.65}$$

$$\sum_{z:D(z)=j} X_z + \sum_{(s,d)\in\Omega, p':\exists z\in p':D(z)=j, z\in Z^N} T^{(s,d)} f_{p'}^{(s,d)} \leq \Delta_r^j \quad \forall j \in V \quad (6.66)$$

$$\sum_{z:(l,m)\in z} X_z + \sum_{(s,d)\in\Omega, p':\exists z\in p':(l,m)\in z, z\in Z^N} T^{(s,d)} f_{p'}^{(s,d)} \leq L_{lm} \quad \forall (l,m) \in A \quad (6.67)$$

$$X_z - \sum_{(s,d)\in\Omega, p:z\in p} T^{(s,d)} f_p^{(s,d)} \geq 0 \quad \forall z \in Z \quad (6.68)$$

$$\sum_{p\in P^{(s,d)}} f_p^{(s,d)} + H^{(s,d)} + f_{p'}^{(s,d)} = 1 \quad \forall (s,d) \in \Omega \quad (6.69)$$

$$f_p^{(s,d)} \geq 0 \quad \forall p \in P^{(s,d)}, (s,d) \in \Omega \quad (6.70)$$

$$H^{(s,d)} \in R^1 \quad \forall (s,d) \in \Omega \quad (6.71)$$

$$0 \leq X_z \leq 1 \quad \forall z \in Z \quad (6.72)$$

Of course, the calculation of the combined reduced cost changes slightly due to coefficient  $T^{(s,d)}$  that is now present in constraints (6.65)-(6.67), and constraint (6.20) is no longer used. In other words, the combined reduced cost for introduction of the new flow path variables in the linear relaxation of the formulation MIP-PATH-LOCALw is as follows.

$$RC_p^{(s,d)} = \sum_{z\in p, z\in Z^E} P_{z;(s,d)}^E + \sum_{z\in p, z\in Z^N} P_{z;(s,d)}^N - w^{(s,d)}$$

where

$$P_{z;(s,d)}^E = T^{(s,d)} v_z, \text{ and}$$

$$P_{z;(s,d)}^N = -T^{(s,d)} a_{O(z)} - T^{(s,d)} b_{D(z)} - T^{(s,d)} \sum_{(l,m)\in z} d_{(l,m)}.$$

While we can use this approach to get the optimal solution for the linear relaxation of the MIP-PATH-LOCALw formulation, we also face a significant drawback from the fact that the linear relaxation of the MIP-PATH-LOCALw formulation is weaker than the linear relaxation of the MIP-PATH-LOCAL formulation. Additionally, since we do not have constraint (6.20) in the MIP-PATH-LOCALw formulation, we need to define  $X_z$  variables as integer, which adds to the complexity of the solution procedures for this formulation. We will next describe details of column generation approach for both MIP-PATH-LOCAL and MIP-PATH-LOCALw.

## Column Generation Algorithm for the MIP-PATH-LOCALw and MIP-PATH-LOCAL Formulations

In each iteration of the column generation algorithm for the MIP-PATH-LOCALw formulation we need to answer the following two questions regarding the flow path variables with minimum negative reduced cost. *Should the new flow path solely use existing lightpaths, new lightpaths, or a mixture of the two?* If we do need to add (use) new lightpath(s) for the new flow path, *what is the best propagation path for the new lightpath(s)?*

As we show next, these questions can be answered using two simple steps. In the first step, the best possible propagation paths (over the physical layer) of potential new lightpaths between all pairs of nodes in the original network are determined by using the all-pairs shortest path algorithm. Then, in the second step, we choose between the lowest (reduced) cost existing lightpath and new lightpath between all pairs of nodes. Once the lightpaths are selected, a new auxiliary graph is constructed, and the shortest path problem on this graph is solved. (In this new graph, the selected lightpaths represent arcs, and the prices  $P_{z;(s,d)}^E$  and  $P_{z;(s,d)}^N$  represent costs associated with these arcs.) As the length of the shortest path between nodes  $s$  and  $d$  reduced by the value of dual variable  $w^{(s,d)}$  represents the minimum reduced cost of all  $f_p^{(s,d)}$  for a given commodity  $(s, d)$ , we know that if this value is negative, then the new flow path(s) needs to be added to the model. Otherwise, if for all commodities, it holds that the minimum reduced cost of all flow variables is non-negative, then the solution of the LP at a given node of the branch-and-bound tree is optimal. We will prove this formally shortly.

To understand the first step of our column generation algorithm recall that the **price** for using a new lightpath  $z$  with origin at node  $i$  and destination at node  $j$  is:

$$P_{z;(s,d)}^N = -T^{(s,d)}a_i - T^{(s,d)}b_j - T^{(s,d)} \sum_{(l,m) \in z} d_{(l,m)} \quad (6.73)$$

Now, in order to find the new lightpaths that could reduce the current objective value, we need to look for lightpaths that minimize the term  $P_{z;(s,d)}^N$ . That is, for each pair of nodes  $(i, j)$ , we need to find a path that minimizes:

$$-a_i - b_j - \sum_{(l,m) \in z} d_{(l,m)}, \quad \text{or} \quad - \sum_{(l,m) \in z} d_{(l,m)}.$$

But, this latter problem can be easily solved as a shortest path problem over an auxiliary graph defined on a physical network, where the cost of using any fiber between nodes  $l$  and  $m$  is given by the value  $-d_{(l,m)}$  (which is positive). Once we apply this procedure, we obtain exactly one new lightpath  $z$  for each pair of nodes  $(l, m)$  in the physical network. (This shortest path problem needs to be solved for each commodity and each pair of nodes separately due to our branching strategy, which requires modification of  $d_{(l,m)}$  costs, but for the clarity of exposition, details of that modification are given later along with the description of the branching strategy applied.)

To determine the solution of the subproblem we now need to make a choice between existing and new lightpaths. As mentioned earlier, this can be done by finding the shortest path between the origin and the destination nodes of a given commodity, where the arcs of the underlying auxiliary graph are represented by (existing and/or new) lightpaths. Note that there may exist more than one lightpath between two nodes, and the auxiliary graph may have many parallel arcs of the same or different cost. Given that in the shortest path, only the cheapest arc between two pairs of nodes will be selected, we can reduce the auxiliary graph to include only one arc for a given pair of nodes by selecting the lightpath with a minimum cost. Observe that this could be either an existing or a new lightpath.

If for all commodities  $(s, d)$ , the shortest path cost turns out to be greater than or equal to the value of the corresponding dual variable  $w^{(s,d)}$ , we are done, as the reduced cost of all flow variables not in the current restricted model is nonnegative, and we cannot reduce the objective value any further. However, if for a given commodity  $(s, d)$ , the reduced cost turns out to be negative, we first need to check if any new lightpaths need to be added for the entering flow variable, and if so, we add these lightpaths, and the corresponding constraints (6.21) and (6.25) to the model. Once this is done, the new flow path variable is added to the restricted master model. The procedure is repeated for all commodities, so in one iteration of the column generation algorithm we may add one new flow variable for each commodity.

The column generation algorithm described above for the MIP-PATH-LOCALw formulation can be directly applied to the MIP-PATH-LOCAL formulation with three minor modifications.

First, the **price** for using an existing lightpath  $z$  is:

$$P_{z;(s,d)}^E = r_z^{(s,d)} + T^{(s,d)}v_z.$$

Second, the **price** for using a new lightpath  $z$  with origin at node  $i$  and destination at node  $j$  is:

$$P_{z;(s,d)}^N = -a_i - b_j - \sum_{(l,m) \in z} d_{(l,m)}$$

And, finally, if for a given commodity  $(s, d)$ , the shortest path cost turns out to be negative, and a new lightpath needs to be introduced, then we add the variable corresponding to that lightpath, and we also add the corresponding constraints (6.20), (6.21), and (6.25) to the model. (Recall that in the MIP-PATH-LOCALw formulation we never use constraint (6.20).) As we explained previously, this approach provides an upper bound for the linear relaxation of the MIP-PATH-LOCAL formulation. And, when combined with a valid branching strategy, this column generation algorithm can be used to derive valid upper bounds for the WDM optical network design problem.

The summary of the main steps of our pricing procedure for the formulations MIP-PATH-LOCALw and MIP-PATH-LOCAL are outlined in Figure 6.2.

We now prove the correctness of the proposed column generation algorithm for the MIP-PATH-LOCALw formulation.

**Proposition 6.1** *If at any given iteration of the proposed column generation algorithm the term  $\sum_{z \in p: z \in Z^E} P_{z;(s,d)}^E + \sum_{z \in p: z \in Z^N} P_{z;(s,d)}^N - w^{(s,d)}$  is nonnegative for all flow paths  $p$  and all commodities  $(s, d)$ , then the current solution of the linear relaxation of the MIP-PATH-LOCALw is optimal.*

**Proof:** The correctness of the proposed column generation algorithm can be proved by considering three different types of new flow paths: those that only use lightpaths already included in the restricted master model, those that only use new lightpaths that are not included in the restricted master model, and those that use both lightpaths that are included and lightpaths that are not included in the restricted master model.

**Case 1. Flow paths that only use lightpaths already included in the restricted master model**

The reduced cost of any flow path  $p$  using only lightpaths already included in the restricted master model can be computed directly from the MIP-PATH-LOCALw formulation as:

$$RC_p^{(s,d)} = \sum_{z \in p, z \in Z^E} T^{(s,d)}v_z - w^{(s,d)}$$

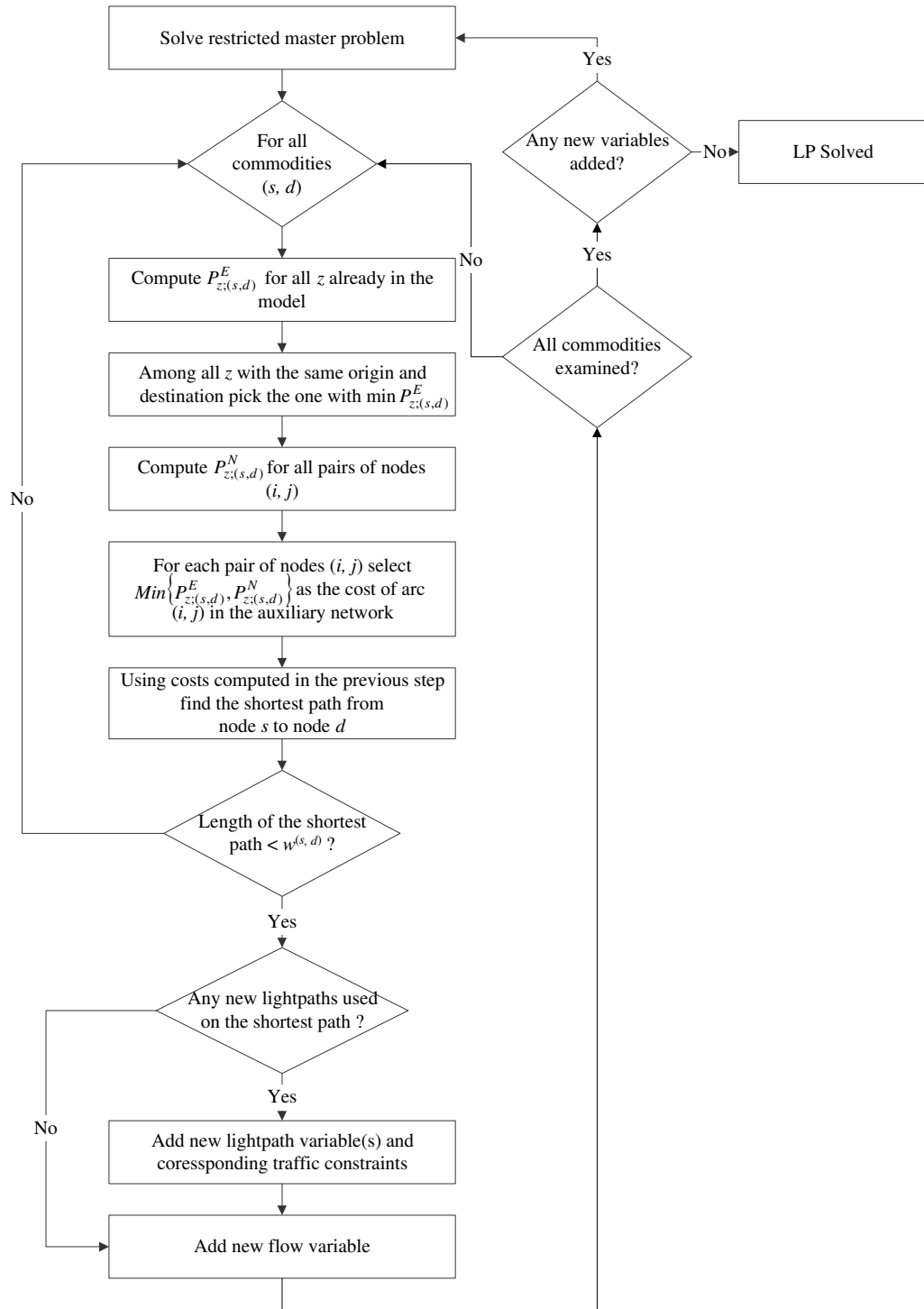


Figure 6.2: Steps of column generation algorithm for MIP-PATH-LOCALw and MIP-PATH-LOCAL formulations



If the above reduced cost is nonnegative for all  $p$  and  $(s, d)$ , it means that the addition of any of the flow path variables not included in the restricted master model would not change objective value, and since the addition of the corresponding constraint to the dual problem would not violate its feasibility, it follows that the current solution cannot be improved through the addition of new flow paths that would only use existing lightpaths.

**Case 2. Flow paths that only use lightpaths that are not included in the restricted master model**

We have already established that in terms of the sequential steps of column generation, the introduction of a new lightpath variable is equivalent to the introduction of a single or multiple new flow path variables that would replace all occurrences of the new lightpath variables in the linear relaxation of the MIP-PATH-LOCALw formulation. In that case the reduced cost of any flow path  $p'$  using only new lightpaths that are not included in the restricted master model can be computed directly from formulation (6.64) - (6.72) as:

$$RC_p^{(s,d)} = \sum_{z \in p, z \in Z^N} T^{(s,d)}(-a_{O(z)} - b_{D(z)} - \sum_{(m,n) \in z} d_{(m,n)}) - w^{(s,d)}.$$

As in Case 1, if the above reduced cost is nonnegative for all  $p'$  and  $(s, d)$ , it means that no new flow path or lightpath variables need to be added to the restricted master model.

**Case 3. Flow paths that use both lightpaths that are included in the restricted master model and lightpaths that are not included in the restricted master model**

Using formulation (6.64) - (6.72), the reduced cost of any new flow path variable using both lightpath variables that included in the restricted master problem and the lightpath variables that are not included in the restricted master problem is:

$$RC_p^{(s,d)} = \sum_{z \in p, z \in Z^E} T^{(s,d)}v_z + \sum_{z \in p, z \in Z^N} T^{(s,d)}(-a_{O(z)} - b_{D(z)} - \sum_{(l,m) \in z} d_{(l,m)}) - w^{(s,d)}$$

where the first term  $T^{(s,d)}v_z$  is computed from (6.68) for existing lightpaths used by  $f_p^{(s,d)}$ . The second term,  $T^{(s,d)}(-a_{O(z)} - b_{D(z)} - \sum_{(l,m) \in z} d_{(l,m)})$  is computed from (6.65), (6.66), and (6.67) for the new lightpaths used by  $f_p^{(s,d)}$ .

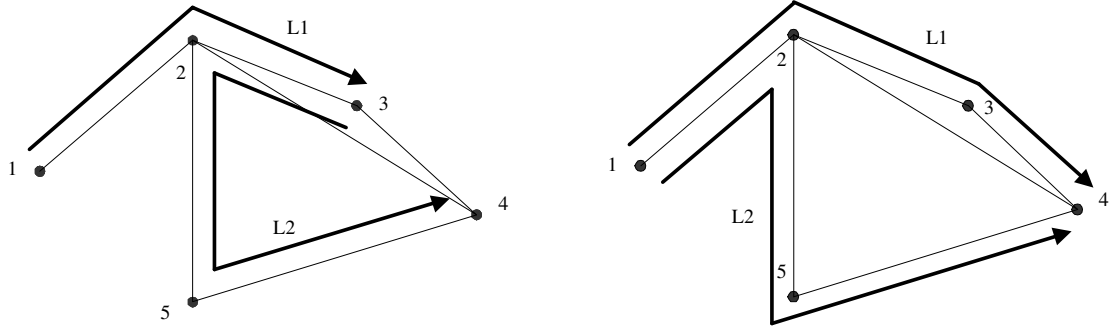
So, if the above reduced cost turns out to be nonnegative for all flow paths, we do not need to add any new variables as the current solution is optimal. □

## Branching Strategy

The selection of a branching strategy used in any branch-and-price algorithm is often more complicated than in standard MIPs where all columns are already included in the model. The reason is that when we start with only a subset of columns, and solve the LP at the root node using column generation, it is still possible that additional columns need to be added to the model in order to find the optimal MIP solution. In other words, as we perform branching, we still need to use column generation algorithm at all subsequent nodes of the search tree. Now, in order to maintain the original structure of the pricing problem used to solve LPs in the search tree, we have to make sure that the branching rules used in the tree can be implemented without causing significant changes to the pricing problem.

As mentioned and discussed earlier in Section 6.4, the branching strategy for the standard ODIMCF problem defined in Barnhart et al. [7] maintains the structure of the pricing problem and works well for that problem. However, this branching strategy would not work well with our WDM optical network design problem if applied directly. The reason is that the branching strategy proposed in [7] is based on the assumption that a given flow path visits each node at most once, and, therefore, in the final solution there is at most one arc originating from a given node in the network for a given commodity. In the WDM optical network design problem, on the other hand, it is possible that single flow path visits the same node more than once in the physical network.

The example shown in Figure 6.3 illustrates one such situation. Here, the commodity originating at node 1, and terminating at node 4, uses two lightpaths. The lightpath L1 originates at node 1, and terminates at node 3, while the lightpath L2 originates at node 3 and terminates at node 4. The actual propagation paths of L1 and L2 are  $(1, 2, 3)$  and  $(3, 2, 5, 4)$ , which means that the propagation path of the flow of this commodity in the physical layer is  $(1, 2, 3, 2, 5, 4)$ . Now, let's assume that this flow path  $(1, 2, 3, 2, 5, 4)$  is the path that should be used in the final optimal solution and that it is not yet introduced in our model. If at the time of branching we identify two fractional flow paths  $(1, 2, 5, 4)$  and  $(1, 2, 3, 4)$  (see Figure 6.3(b)), the branching strategy defined in [7] would create two new nodes. In the first one, arc  $(2, 5)$ , in the physical layer, would be



(a) Optimal routing solution for the flow of commodity (1, 4)

(b) Fractional routing solution for the flow of commodity (1, 4)

Figure 6.3: Example of the branching issue for the flow path visiting the same node more than once in the physical topology

forbidden and in the second node arc (2, 3), in the physical layer, would be forbidden. Given these rules we would never be able to identify the optimal path (1, 2, 3, 2, 5, 4) as that path requires use of both arcs (2, 3) and (2, 5).

We will show next that when applied hierarchically by moving from the higher to the lower layer, we can still apply a similar branching strategy to our problem. The idea is based on the fact that *in the logical topology*, a flow path never visits the same node twice, and *in the physical topology*, a lightpath never visits the same node twice. So, we can design a branching strategy, where we first make sure that the flow path for any given commodity has a unique path in the logical topology (for example, flow path (1, 4) is using lightpaths (1, 3) and (3, 4), but we do not guarantee that the flow of a given commodity is not split (bifurcated) between multiple lightpaths with the same origin and destination. Once we make sure that flow paths for all commodities satisfy this condition, we move on to branching in the physical layer (assuming that we still have fractional solution). Now, if a given commodity has a fractional solution, we check if any two flow paths of this commodity differ in the physical topology. If we find two such paths, we first identify origin and destination of the lightpaths that have different propagation paths, and then we apply the same branching strategy as in [7]. That is, we identify the divergence node for the two lightpaths that have different propagation paths and define two sets of arcs, of approximately the same size, each containing one of the arcs used by the fractional paths. We then branch into

two problems, so that on one branch we prohibit this commodity to use the lightpaths that are using any of the arcs in the first set, while on the second branch we prohibit this commodity to use the lightpaths that are using any of the arcs in the second set.

Finally, if the fractional flow path variables use identical paths between origin and destination of the corresponding commodity both in the logical and the physical topology, but are split among different lightpaths, we apply the following branching strategy. We first select one commodity with a fractional flow. Then we identify two lightpaths with the same origin  $i$  and destination  $j$  that are used by this commodity. Next, we define two (exhaustive and mutually exclusive) sets of lightpaths originating and terminating at nodes  $i$  and  $j$  respectively. On the first branch we prohibit this commodity to use any of the lightpaths in the first set. On the second branch we prohibit this commodity to use any of the lightpaths in the second set. Notice that this last branching rule may cause the generation of many identical lightpaths with the same origin and destination (i.e., once we prohibit the use of a certain set of lightpaths, there is nothing to stop the generation of a new, identical lightpath in the next iteration of our column generation algorithm). But, although we may end up creating unnecessary lightpaths, we guarantee termination of the algorithm by imposing an additional rule. That is, *for each commodity*, we keep track of the number of forbidden lightpaths that have the same origin and destination, and which use the same arc in the physical topology. Once this number reaches the maximum possible number of lightpaths that can be established over given arc in the physical topology (this is either the number of transmitters and receivers, or the maximum number of lightpaths that can be established on any physical link), we do not allow generation of any additional new lightpaths with the same origin and destination that require use of the same arc. (This rule is used separately for each commodity, so we may not allow use of certain lightpaths for one commodity, but we may allow use of the same lightpaths for other commodities.)

More specifically, we carry out the following branching strategy whenever the LP solution at a given node of the branch-and-bound tree is fractional.

**Step 1.** Check if there are any commodities with fractional lost traffic. If there is no such com-

modity go to Step 4.

**Step 2.** Identify commodity with the greatest demand that has fractional lost traffic.

**Step 3.** Create 2 new nodes:

Node 1. Set  $H^{(s,d)} = 0$

Node 2. Set  $H^{(s,d)} = 1$ .

Exit.

**Step 4.** Check if there are any commodities with fractional traffic that use paths differing in the *logical* layer. If there are no such commodities, go to Step 7.

**Step 5.** Identify one commodity with fractional traffic that uses paths differing in the logical layer. Locate the first divergence node in the logical layer, and define two subsets of lightpaths originating at the divergence node such that one set contains one of the fractional paths of the given commodity, and the second set contains the other fractional path of the same commodity. (In other words, use the branching strategy developed in [7], but only in the logical layer.)

**Step 6.** Create 2 new nodes:

Node 1. Do not allow lightpaths from the first set defined in Step 5 to be used by the selected commodity.

Node 2. Do not allow lightpaths from the second set defined in Step 5 to be used by the selected commodity.

Exit.

**Step 7.** Check if there are any commodities with fractional traffic that use identical paths in terms of the lightpath origins and destinations, but the lightpaths have different propagation path in the physical layer. If there are no such commodities, go to Step 10.

**Step 8.** Select one commodity with fractional traffic that uses identical paths in terms of the lightpaths origins and destinations, but that has different propagation paths in the physical topology. Also, select two lightpaths carrying fractional traffic of this commodity that have identical origin and destination, but have different propagation paths in the physical topology. Locate the divergence node of these two lightpaths in the physical layer and define 2 sets of arcs such that

one set contains the arc originating at divergence node belonging to the first fractional flow path, and the second one contains the arc originating at the divergence node belonging to the second fractional flow path.

**Step 9** Create 2 new nodes:

Node 1. Do not allow lightpaths for a selected origin and destination, and commodity to use any of the arcs from the first set defined in Step 8.

Node 2. Do not allow lightpaths for a selected origin and destination, and commodity to use any of the arcs from the second set defined in Step 8.

Exit.

**Step 10.** Select one commodity with a fractional flow. Identify two lightpaths with the same origin  $i$  and destination  $j$  that are used by this commodity (notice that these lightpaths will have the same propagation path in the physical layer). Next, define two (exhaustive and mutually exclusive) sets of lightpaths originating and terminating at nodes  $i$  and  $j$  respectively.

**Step 11.** Create 2 new nodes:

Node 1. Prohibit the selected commodity from using any of the lightpaths in the first set.

Node 2. Prohibit the selected commodity from using any of the lightpaths in the second set.

Exit.

In order to ensure proper implementation of the rules defined in Step 6, and Step 9, we set the cost  $d_{(l,m)}$  to high values in the pricing part of the column generation algorithm for those arcs that are prohibited by the proposed branching rule in Step 9. Similarly, we never generate lightpaths for origin - destination pairs that are forbidden (in Step 6) for a given commodity.

Since the  $X_z$  variables in the MIP-PATH-LOCAL formulation do not have to be defined as integers, the described branching strategy provides an integer solution for this formulation, and therefore provides a valid upper bound for the original WDM optical network design problem.

In the case of the MIP-PATH-LOCALw formulation, we need to perform additional branching in order to ensure that the  $X_z$  variables are integer. It turns out that this is a non-trivial task, and we will next explain the complexity behind branching on the lightpath variables, and propose

one possible branching strategy.

In order to understand the problem related to branching on the lightpath variables, consider what would happen if we performed branching based on variable dichotomy. Basically, for a given fractional variable  $X_z$ , and its current fractional value  $X_z^*$ , we would create 2 new nodes by adding restriction  $X_z \leq \lfloor X_z^* \rfloor$  on one branch, and restriction  $X_z \geq \lceil X_z^* \rceil$  on the other branch. Enforcing the second restriction is not a problem, however, the first restriction requires that we do not generate any new lightpaths that would have the same propagation path as the lightpath  $z$  corresponding to the variable  $X_z$ . As in the case of the flow path variables, this is hard to enforce, so we propose a different branching strategy that indirectly forces all lightpath variables to be binary.

This strategy is based on observation that if the **sum** of all lightpath variables with the same origin and destination is integer over every arc used in the physical layer, then either:

- a) all lightpath variables are integer, or
- b) all lightpath variables do not have integer values, but the solution can be interpreted as integer.

The first case is obviously correct, since if all lightpath variables are integer, it directly follows that **sum** of all lightpath variables with the same origin and destination over all arcs in the physical network is integer as well. The second case, however, is not so obvious. First, note that if the sum of all lightpath variables is integer over all arcs in the physical network, it means that the network can support an integer number of lightpaths over all arcs used (the number of lightpaths of a given origin and destination that the network can support on any arc is equal to the sum of all the lightpath variables with the same origin and destination that use that arc). Now, if the network can support an integer number of lightpaths between two nodes, we can reassign arcs for the fractional lightpath variables so that we obtain integer values for each lightpath variable. This reassignment of values of the fractional lightpaths variables can be formally solved as a single commodity flow problem, where we need to route  $\sum_{z:O(z)=i,D(z)=j} X_z$  units of flow between given origin  $i$  and destination  $j$  over the network with arc capacities defined by the total number of lightpaths with origin node  $i$  and destination node  $j$  that were supported on each arc (i.e., capacity

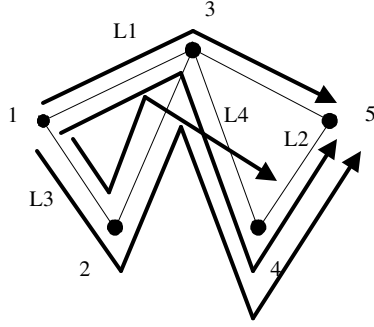


Figure 6.4: Example of the branching issue for the MIP-PATH-LOCALw formulation

of each arc  $(l, m)$  is set to  $\sum_{z:(l,m) \in z, O(z)=i, D(z)=j} X_z^*$ , and the cost of carrying a unit of flow on each arc is set to 1.

An example of situation where the sum of all lightpath variables with the same origin and destination is integer over every arc used in the physical layer, but there are fractional lightpath variables is shown in Figure 6.4. Lightpath  $L_1$  uses path  $(1, 3, 5)$ , lightpath  $L_2$  uses path  $(1, 3, 4, 5)$ , lightpath  $L_3$  uses path  $(1, 2, 3, 4, 5)$ , and lightpath  $L_4$  uses path  $(1, 2, 3, 5)$ . Values of these variables in a current solution are 0.7 for the lightpaths  $L_1$  and  $L_3$ , and 0.3 for the lightpaths  $L_2$  and  $L_4$ . Since the flow balance constraints guarantee that the sum of all ‘flows’ into any given transient node must equal sum of all ‘flows’ going out of that node, it follows that in any fractional solution that satisfies requirement of integer sum of lightpath variables over all arcs we can eliminate some of the fractional lightpaths so that an integer solution is obtained, without any impact on other aspects of the current solution. In the example shown in Figure 6.4, we could, for example, eliminate lightpaths  $L_2$  and  $L_4$ , and round up values of variables associated with variables  $L_1$  and  $L_3$ , without making any other changes to the current solution. This finding allows us to branch on the **sum** of all lightpath variables with the same origin and destination that are using the same arc in the physical layer, instead of directly branching on individual lightpath variables.

The specific branching step that we apply to ensure integrality of  $X_z$  variables are as follows.

**Step 0.** Check if there is an arc with a fractional sum of all lightpath variables with the same origin and destination using that arc. If there is no such arc, go to Step 1, otherwise create 2 new nodes:



Node 1. Add restriction  $\sum_{z:(m,n) \in z} X_z \leq \lfloor \sum_{z:(m,n) \in z} X_z^* \rfloor$

Node 2. Add restriction  $\sum_{z:(m,n) \in z} X_z \geq \lceil \sum_{z:(m,n) \in z} X_z^* \rceil$

where  $(m, n)$  and  $(i, j)$  are an arc and the lightpath origin-destination pair identified in the previous check, and  $X_z^*$  are the actual values of lightpath variables in the current solution.

Exit.

In our implementation of the branching strategy for the MIP-PATH-LOCAL<sub>w</sub>, we first perform Step 0, and then the previously described eleven branching steps. In other words, we first branch on the sum of lightpath variables, then we branch on the lost traffic variables, and, finally, we branch on the flow variables.

Finally, in order to guarantee feasibility of the restricted master model at each node of the branch-and-bound tree, we perform the following modifications of our branch-and-price procedures. In the case of the MIP-PATH-LOCAL formulation, the only situation when our branching strategy can cause infeasibility in one of the child nodes is when we impose restriction that all the traffic must be served (that is, when on one branch we set  $H^{(s,d)} = 0$ ). In this case, we may not have a sufficient number of lightpath and/or flow path variables to satisfy additional demand for a given commodity  $(s, d)$  (recall that we performed branching on the lost traffic variables only if the lost traffic variable has a fractional value in a given solution). To ensure that the node is not pruned in such situations, we add an artificial variable to the constraint (6.22) in the restricted master model, which accounts for all traffic that cannot be served. We also set a high cost for this artificial variable in the objective function so that this variable does not assume a positive value when a feasible solution to the original problem exists. If at the end of the column generation stage for a given node in the branch-and-bound tree this variable still has a positive value, we fathom the node, since this means that, given the restrictions in the branch-and-bound tree, a feasible solution for the original problem does not exist (note that if there was a feasible solution, the artificial variable would have to be equal to zero, given its high cost in the objective function).

In the case of the MIP-PATH-LOCAL<sub>w</sub> formulation, we use the same logic, except that the artificial variable is also added to all constraints of the type  $X_z \geq \lceil X_z^* \rceil$ . The reason that we need

to add the artificial variable to the constraint above is that at a given node of the branch-and-bound tree the lightpaths that are present in the restricted master model may be such that there is no sufficient capacity in the network to accommodate the new requirement  $X_z \geq \lceil X_z^* \rceil$ .

### 6.5.2 Applying the Branch-and-Price Algorithms for MIP-PATH-LOCALw and MIP-PATH-LOCAL to WDM Optical Network Design with Alternative Design Objectives

The algorithm proposed in the previous section solves the WDM optical network design problem so that the total lost traffic is minimized. It is possible, however, that different design objectives need to be considered when solving this problem. We address several objectives commonly used in the literature, and show that our algorithm can be easily modified to solve the WDM optical network design for different objective functions. In all cases that we will discuss next, the lost traffic variables can be dropped out from our path formulation without any effect on the pricing procedure. The branching strategy changes only in that we don't use lost traffic variables for branching any longer. Instead, in the case of the MIP-PATH-LOCAL formulation we branch on the sum of fractional flow variables only, while in the case of the MIP-PATH-LOCALw formulation we branch on the sum of fractional lightpath and flow path variables.

#### Minimizing the network-wide average packet delay

Given the propagation delay  $\delta_{(l,m)}$  for each physical arc  $(l,m)$ , this objective function becomes:

$$\text{Min} \quad \sum_{(s,d) \in \Omega, p \in P^{(s,d)}} \sum_{(l,m) \in p} \delta_{(l,m)} f_p^{(s,d)}$$

Notice that if we were to set  $\delta_{(i,j)} = 1$ , for all arcs  $(i,j)$ , the above objective function would minimize the average hop distance measured by the number of physical edges traversed by a flow of a given commodity.

As for our branch-and-price algorithm, the use of this objective function would only change the expression for the reduced cost used in the pricing part as follows:

MIP-PATH-LOCAL:

$$RC_p^{(s,d)} = \sum_{(l,m) \in p} \delta_{(l,m)} + \sum_{z \in p, z \in Z^E} (r_z^{(s,d)} + T^{(s,d)} v_z) - w^{(s,d)} + \sum_{z \in p, z \in Z^N} (-a_{O(z)} - b_{D(z)} - \sum_{(l,m) \in z} d_{(l,m)})$$

MIP-PATH-LOCALw:

$$RC_p^{(s,d)} = \sum_{(l,m) \in p} \delta_{(l,m)} + \sum_{z \in p, z \in Z^E} T^{(s,d)} v_z - w^{(s,d)} + \sum_{z \in p, z \in Z^N} T^{(s,d)} (-a_{O(z)} - b_{D(z)} - \sum_{(l,m) \in z} d_{(l,m)})$$

which can be easily used in the calculation of the cost of using existing and/or new lightpaths.

The remaining part of our branch-and-price algorithm remains the same.

### Minimizing the congestion

In this case the objective function becomes:

$$\text{Min } \lambda_{max}$$

where  $\lambda_{max}$  is a variable that represents the maximum load on any given lightpath. In order to use this objective function, it is necessary to add a new set of constraints to the MIP-PATH-LOCAL and MIP-PATH-LOCALw formulations:

$$\lambda_{max} - \sum_{(s,d) \in \Omega, p: z \in p} T^{(s,d)} f_p^{(s,d)} \geq 0 \quad \forall z$$

This also requires modification of the expression for the reduced cost used in the pricing part of our branch-and-price algorithm in a following way:

MIP-PATH-LOCAL:

$$RC_p^{(s,d)} = \sum_{z \in p, z \in Z^E} (r_z^{(s,d)} + T^{(s,d)} v_z + T^{(s,d)} t_z) + \sum_{z \in p, z \in Z^N} (-a_{O(z)} - b_{D(z)} - \sum_{(l,m) \in z} d_{(l,m)}) - w^{(s,d)}$$

MIP-PATH-LOCALw:

$$RC_p^{(s,d)} = \sum_{z \in p, z \in Z^E} (T^{(s,d)} v_z + T^{(s,d)} t_z) + \sum_{z \in p, z \in Z^N} T^{(s,d)} (-a_{O(z)} - b_{D(z)} - \sum_{(l,m) \in z} d_{(l,m)}) - w^{(s,d)}$$

where  $t_z$  is a nonnegative dual variable associated with the new set of constraints. Again as in the case of minimization of propagation delay, the new objective function requires only a minor change in computation of the reduced cost, and the remaining part of our branch-and-price algorithm remains the same.

### Minimizing the number of transponders

In this case the objective function becomes:

$$\text{Min } \sum_i (\Delta_t^i + \Delta_r^i)$$

In our branch-and-price algorithm, the use of this objective function would only mean that  $\Delta_t^i$  and  $\Delta_r^i$  would be variables, not given constants. This has **no** effect on our branch-and-price algorithm, since lightpath variables  $X_z$  are integer, which immediately implies that  $\Delta_t^i$  and  $\Delta_r^i$  will be integer as well, i.e., there is no need to branch on these variables in order to get integer values.

### 6.5.3 Branch-and-Price-and-Cut-Algorithm for the MIP-PATH-GLOBAL formulation

The advantage of the MIP-PATH-GLOBAL formulation is that all global lightpath variables are included in the model at the beginning of the search, and as flow variables are related only to the global lightpath variables, the pricing part of the column generation algorithm becomes straightforward and local lightpath variables and flow path variables can be priced out separately. The downside of this formulation is that now we need to add packing constraints to the model. Additionally, this formulation cannot be used for problems where it is necessary to know the actual propagation path of a given commodity in the physical layer (such as accounting for the network-wide average delay). Since it is practically impossible to add all the packing constraints (6.34)-(6.37), and the dynamic addition of these constraints requires solution of the bin-packing problem, we will present the branch-and-price algorithm for the relaxed version of the MIP-PATH-GLOBAL formulation that allows bifurcation of traffic between lightpaths with the *same* origin and destination (i.e., we will not consider constraints (6.34)-(6.37)).

Let the following dual variables correspond to the constraints of the MIP-PATH-GLOBAL formulation:

- nonpositive  $a_i$  and  $b_j$  variables for constraints (6.27) and (6.28) respectively
- unrestricted in sign  $g^{(i,j)}$  variables for constraint (6.29)
- nonpositive  $d_{(l,m)}$  variables for constraint (6.30)
- nonpositive  $r_{(i,j)}^{(s,d)}$  variables for constraint (6.31)
- nonnegative  $v_{(i,j)}$  variables for constraint (6.32)
- unrestricted in sign  $w^{(s,d)}$  variables for constraint (6.33)

The reduced cost of any  $X_z^{(i,j)}$  variable is:

$$RC_z^{(i,j)} = g^{(i,j)} - \sum_{(l,m) \in z} d_{(l,m)}.$$

The reduced cost of any  $f_p^{(s,d)}$  variable is:

$$RC_p^{(s,d)} = \sum_{(i,j) \in p} (r_{(i,j)}^{(s,d)} + T^{(s,d)}v_{(i,j)}) - w^{(s,d)}.$$

The new lightpaths that need to be added to the restricted master problem are identified by finding paths for each  $(i, j)$  that minimize the following expression:

$$g^{(i,j)} - \sum_{(l,m) \in z} d_{(l,m)}$$

or

$$- \sum_{(l,m) \in z} d_{(l,m)}.$$

This can be solved as a shortest path problem on an auxiliary network where the cost of an arc  $(l, m)$  is  $-d_{(l,m)}$ . If the cost of this path is greater than  $-g^{(i,j)}$  the lightpath is added. Otherwise, no new lightpaths with origin at node  $i$  and destination at node  $j$  are added.

The new flow paths that need to be added to the restricted master problem are identified by finding the path for each commodity  $(s, d)$  with the minimum value of:

$$\sum_{(i,j) \in \Lambda} (r_{(i,j)}^{(s,d)} + T^{(s,d)}v_{(i,j)}) - w^{(s,d)}$$

or

$$\sum_{(i,j) \in \Lambda} (r_{(i,j)}^{(s,d)} + T^{(s,d)}v_{(i,j)}).$$

This can be solved as the shortest path problem on an auxiliary network where the cost of an arc  $(i, j)$  is defined by the expression  $r_{(i,j)}^{(s,d)} + T^{(s,d)}v_{(i,j)}$ . If the cost of the shortest path is less than  $w^{(s,d)}$  the flow path is added. Otherwise, no new flow paths for commodity  $(s, d)$  are added.

## Branching Strategy

Although only local lightpath variables  $X_z^{(i,j)}$  and flow path variables  $f_p^{(s,d)}$  need to be defined as integers in the MIP-PATH-GLOBAL formulation, the branching strategy proposed here uses global lightpath variables too. Similar to the branching strategy defined for the MIP-PATH-LOCAL formulation, branching is performed hierarchically from the higher to the lower level. Therefore, we first branch on the global lightpath variables, then on the local lightpath variables, followed by branching on the lost traffic variables, and, finally, we branch on the flow path variables. For the global lightpath variables and the lost traffic variables, variable dichotomy is used, while branching decisions for the local lightpath variables and the flow path variables are more complicated.

The problem related to branching on the local lightpath variables in the MIP-PATH-GLOBAL formulation is identical to the one that we previously discussed for the MIP-PATH-LOCALw formulation. In other words, if we were to use variable dichotomy to branch on these variables, then for a given fractional variable  $X_z^{(i,j)}$ , and its current fractional value  $X_z^{(i,j)*}$ , we would create 2 new nodes by adding restriction  $X_z^{(i,j)} \leq \lfloor X_z^{(i,j)*} \rfloor$  on one branch, and restriction  $X_z^{(i,j)} \geq \lceil X_z^{(i,j)*} \rceil$  on the other branch. As before, enforcing the second restriction is not a problem, however, the first restriction requires that we do not generate any new lightpaths that would have the same propagation path as the lightpath  $z$  corresponding to the variable  $X_z^{(i,j)}$ . To resolve this issue, we use the same branching strategy that we defined for the  $X_z$  variables in the MIP-PATH-LOCALw formulation.

Branching on the fractional flow variables can be performed using the branching strategy defined in Barnhart et al. [7] by looking at the divergence node of the fractional flow variables in the logical topology.

In summary, whenever the solution at a given node of the branch-and-bound tree is fractional, we perform the following steps.

**Step 1.** Check if there are any fractional global lightpath variables. If there are no such variables, go to Step 2, otherwise, select one fractional global variable, and create 2 new nodes using variable dichotomy.

**Step 2.** Check if there is an arc with a fractional sum of all local lightpath variables with the same origin and destination using that arc. If there is no such arc, go to Step 3, otherwise create 2 new nodes:

Node 1. Add restriction  $\sum_{z:(m,n) \in z} X_z^{(i,j)} \leq \lfloor \sum_{z:(m,n) \in z} X_z^{(i,j)*} \rfloor$

Node 2. Add restriction  $\sum_{z:(m,n) \in z} X_z^{(i,j)} \geq \lceil \sum_{z:(m,n) \in z} X_z^{(i,j)*} \rceil$

where  $(m, n)$  and  $(i, j)$  are an arc and lightpath origin-destination pair identified in the previous check, and  $X_z^{(i,j)*}$  are the actual values of local lightpath variables in the current solution.

Exit.

**Step 3.** Check if there are any commodities with fractional lost traffic. If there is no such com-

modity go to Step 4. Otherwise, create 2 new nodes:

Node 1. Set  $H^{(s,d)} = 0$

Node 2. Set  $H^{(s,d)} = 1$ .

Exit.

**Step 4.** Identify a commodity with fractional flow path variables, and locate the divergence node.

Then, define two subsets of global lightpaths originating at the divergence node such that one set contains one of the fractional paths of the given commodity, and the second set contains the other fractional path of the same commodity. (In other words, use branching strategy developed in [7], but in the logical layer.) Create 2 new nodes:

Node 1. Do not allow global lightpaths from the first set defined in this step to be used by the selected commodity.

Node 2. Do not allow global lightpaths from the second set defined in this step to be used by the selected commodity.

Exit.

Finally, in order to ensure feasibility of the restricted master model at the beginning of the column generation procedure at each node of the branch-and-bound tree, we modify the MIP-PATH-GLOBAL formulation by using an approach similar to the one that we described for the MIP-PATH-LOCAL and the MIP-PATH-LOCALw formulations. Specifically, we add a high cost, artificial variable to the objective function, as well as in the constraint (6.33), which ensures that either all demand for a given commodity is satisfied, or is entirely lost. We also add an artificial variable when we add constraints of the type  $Y^{(i,j)} \geq \lceil Y^{(i,j)*} \rceil$  and  $\sum_{z:(m,n) \in z} X_z^{(i,j)} \geq \lceil \sum_{z:(m,n) \in z} X_z^{(i,j)*} \rceil$  in our branching rules.

#### 6.5.4 Applying the Branch-and-Price Algorithm for MIP-PATH-GLOBAL to WDM Optical Network Design with Alternative Design Objectives

The proposed branch-and-price algorithm for the MIP-PATH-GLOBAL formulation can be used with minor modifications for certain alternative network design objectives. However, as we discussed in the previous chapters, when *packing* constraints are not used with the MIP-PATH-

GLOBAL formulation, this formulation cannot be used in situations where we need information regarding propagation paths of the flow path variables in the physical topology. (Recall that when the packing constraints are not included in the MIP-PATH-GLOBAL formulation, bifurcation of flow is not allowed only in the logical layer. But, it is possible that the traffic of a given commodity is split among multiple lightpaths with the same origin and destination.) For this reason, MIP-PATH-GLOBAL cannot be used for objectives of minimizing the network-wide average packet delay or minimizing the congestion. The problem with the objective of minimizing the network-wide average packet delay is that computation of packet delay requires knowledge of the exact propagation path for each commodity in the physical layer. In the case of objective of minimizing the congestion we need to know the exact amount of flow carried on each *local* lightpath, and, since MIP-PATH-GLOBAL does not provide this information, we cannot use the corresponding branch-and-price algorithm for this network design objective.

In the situations where we don't need information on propagation of the flow paths in the physical layer, we can easily modify our branch-and-price algorithm for MIP-PATH-GLOBAL for the same objective designs as we did our branch-and-price algorithm for MIP-PATH-LOCAL. For example, in the case where we need to minimize the number of transponders in the network, the objective function is:

$$\text{Min} \quad \sum_i (\Delta_t^i + \Delta_r^i)$$

where  $\Delta_t^i$  and  $\Delta_r^i$  represent variables corresponding to the number of transmitters and receivers used at each node in the network. Just like in the case of MIP-PATH-LOCALw and MIP-PATH-LOCAL, the use of this objective function would only mean that  $\Delta_t^i$  and  $\Delta_r^i$  would be variables, not given constants. This has **no** effect on our branch-and-price algorithm, since integrality of  $Y^{(i,j)}$  variables immediately implies that  $\Delta_t^i$  and  $\Delta_r^i$  will be integer as well, i.e., there is no need to branch on these variables in order to get integer values.

## 6.6 Valid Inequalities for WDM Optical Network Design Formulations

In this section, we discuss possible ways to develop stronger formulations for WDM optical network design problem. We will first explain applicability of the lifted cover inequalities, and



then describe a simple class of valid inequalities that can be used for WDM optical networks that guarantee service for all traffic requests.

### 6.6.1 Lifted Cover Inequalities for the WDM optical network design problem

#### Background

Lifted cover inequalities represent a class of inequalities that can be useful for strengthening knapsack type of constraints. Specifically, if the set of variables  $x_j$ ,  $j \in C \subset N$  represents a minimal cover<sup>3</sup> for a knapsack constraint:

$$\sum_{j=1}^n a_j x_j \leq b \quad (6.74)$$

$$x_j \in B^1 \quad \forall j \in N \quad (6.75)$$

then constraint

$$\sum_{j \in C} x_j \leq |C| - 1 \quad (6.76)$$

represents a valid inequality for (6.74) - (6.75).

One way to strengthen inequality (6.74) is to try to include other variables  $x_j$  ( $j \in N \setminus C$ ), and derive a new inequality

$$\sum_{j \in C} x_j + \sum_{j \in N \setminus C} \alpha_j x_j \leq |C| - 1 \quad (6.77)$$

Inequality of the type (6.77) is known as a *simple lifted cover inequality* [31], where the coefficients  $\alpha_i$  for each variable  $x_i$  ( $i \in N \setminus C$ ) are determined using the following relationship:

$$\alpha_i = |C| - 1 - t_i$$

where  $t_i$  is obtained by solving the following problem.

$$t_i = \text{Max} \sum_{j \in C} x_j + \sum_{j \in L_u} \alpha_j x_j \quad (6.78)$$

$$\text{s.t.} \quad \sum_{j \in C \cup L_u} a_j x_j \leq b - a_i \quad (6.79)$$

---

<sup>3</sup>Recall that a set  $C \subset N$  is a cover for a given knapsack constraint if  $\sum_{j \in C} a_j > b$ . We say that the cover is minimal if (for any  $j \in C$ ) the set  $C' = C \setminus \{j\}$  is not a cover.

The set  $L_u$  in the problem (6.78)-(6.79) includes all variables  $x_j$  ( $j \in N \setminus C$ ) that are already included in the lifted cover inequality (6.80).

$$\sum_{j \in C} x_j + \sum_{j \in L_u} \alpha_j x_j \leq |C| - 1 \quad (6.80)$$

The procedure of finding coefficients  $\alpha_j$  for  $j \in N \setminus C$  is known as *up-lifting* due to the fact that we are trying to find coefficients that would make inequality (6.80) valid when variables  $x_j \in N \setminus C$  are equal to one.

Identification of the initial cover can be done using a simple algorithm of Gu et al. [31], which works in the following way. A binary variable  $z_j$  is defined for each variable  $x_j$ , and it is set to zero for all  $x_j = 0$ , while the remaining  $x_j$  variables are sorted in non-increasing order of their current value. The corresponding  $z_j$  variables are then assigned a value of 1 starting from the beginning of the list, until the following condition is satisfied.

$$\sum a_j z_j > b.$$

The set of  $x_j$  variables for which  $z_j = 1$  then defines an initial cover  $C$ .

### **Lifted Cover Inequalities for MIP-PATH-LOCALw and MIP-PATH-LOCAL Formulation**

A good candidates for cover inequalities are the knapsack like constraints. In the MIP-PATH-LOCAL and MIP-PATH-LOCALw formulations, we have one knapsack type constraint that defines the total amount of traffic that can be sent over any given lightpath:

$$X_z - \sum_{(s,d) \in \Omega, p: z \in p} T^{(s,d)} f_p^{(s,d)} \geq 0.$$

In this constraint, we are trying to pack as many commodities as possible on the lightpath  $z$ , so we can treat this constraint as a knapsack constraint where the right hand side is equal to the capacity of the lightpath  $z$ . A cover inequality for this constraint can be developed by identifying lightpaths with a non-zero flow (similar to the idea of Barnhart et al. [7], where one LCI is identified for each saturated arc in the LP solution). To identify the initial cover, we can follow the procedure of Gu et al. [31], and define a single  $\gamma^{(s,d)}$  binary variable for each commodity  $(s, d)$ , and perform the following steps.

For each lightpath  $z$  with a non-zero flow:

Step 1. Set  $\gamma^{(s,d)} = 0$  for all commodities with  $\sum_{z \in p} f_p^{(s,d)} = 0$

Step 2. Sort all  $\gamma^{(s,d)} > 0$  in non-increasing order of their value  $\sum_{z \in p} f_p^{(s,d)}$

Step 3. Set  $\gamma^{(s,d)} = 1$  starting from the beginning of the list, until weight  $\sum_{(s,d) \in \Omega} T^{(s,d)} \gamma^{(s,d)}$  exceeds 1.

Set  $\gamma^{(s,d)} = 0$  for the remaining commodities

Step 4. Set  $C = \{(s,d) : \gamma^{(s,d)} = 1\}$ . If  $C = \emptyset$ , STOP.

Once the initial cover is identified, we can obtain a lifted cover inequality

$$\sum_{(s,d) \in \Omega, p: (s,d) \in C, z \in p} f_p^{(s,d)} + \sum_{(s,d) \in \Omega, p: (s,d) \notin C, z \in p} \alpha^{(s,d)} f_p^{(s,d)} \leq |C| - 1. \quad (6.81)$$

Here, the coefficients  $\alpha^{(s,d)}$  are determined by solving the following problem for each  $(s,d) \notin C$ .

$$\alpha^{(s,d)} = |C| - 1 - t^{(s,d)}$$

where

$$t^{(s,d)} = \text{Max} \sum_{(s,d) \in C} f_p^{(s,d)} + \sum_{(s,d) \in L_u} \alpha^{(s,d)} f_p^{(s,d)} \quad (6.82)$$

$$\text{s.t.} \quad \sum_{(s,d) \in C \cup L_u} T^{(s,d)} f_p^{(s,d)} \leq 1 - T^{(s,d)} \quad (6.83)$$

As before, the set  $L_u$  in the problem (6.82) - (6.83), includes all commodities  $(s,d) \notin C$  that are already included in the lifted cover inequality.

An apparent drawback of the LCIs defined this way is that these LCIs are lightpath specific, and after adding an LCI for a given lightpath  $z$ , we may generate an identical lightpath  $z$ , which would be less expensive than the original one (this is due to the fact that the lightpaths that are already in the model, and have one or more LCIs defined, are more restrictive than any new lightpath that does not have an LCI defined on it, and therefore have higher cost in the pricing part of the column generation algorithm).

We performed a set of computational experiments to test the impact of the addition of LCIs to both MIP-PATH-LOCAL and MIP-PATH-LOCALw. We found that, as expected, the addition of the lifted cover inequalities does not improve lower bounds and that there are two reasons why this happens. First, the number of lightpaths with a non-zero flow at any given iteration of the

column generation algorithm is high, and the identification of LCIs requires long computational times. Second, there is no guarantee that multiple identical lightpaths are not generated due to the LCIs added for the lightpaths that are already included in the restricted master model.

### 6.6.2 Valid Inequalities for WDM Optical Networks that Guarantee Service for all Traffic Requests

When we know that all the traffic in the network *must be* served, we can use this fact to derive new valid inequalities that may strengthen our formulations for the WDM optical network design problem. A simple class of such inequalities defines the minimum number of lightpaths that need to originate and terminate at any given node in the network. We describe these inequalities next.

#### Node Degree Inequalities

If all the traffic originating from a given node  $i$  must be served, then we can add the following type of valid inequality to each of our mathematical formulations for the WDM optical network design problem. For MIP-PATH-LOCALw and MIP-PATH-LOCAL we add:

$$\sum_{z:O(z)=i} X_z \geq \Gamma_i^{(s)} \quad \forall i \in V : \exists (s, d) \in \Omega : s = i$$

where  $\Gamma_i^{(s)}$  is the minimum number of lightpaths needed to serve all commodities originating at node  $i$ .

Similarly, we can add to MIP-PATH-LOCALw and MIP-PATH-LOCAL the following valid inequality:

$$\sum_{z:D(z)=j} X_z \geq \Gamma_j^{(d)} \quad \forall i \in V : \exists (s, d) \in \Omega : d = j$$

where  $\Gamma_j^{(d)}$  is the minimum number of lightpaths needed to serve all commodities with a destination at node  $j$ .

In the case of the MIP-PATH-GLOBAL formulation, we use the global lightpath variables to define these inequalities. In other words, we add the following constraints for each node in the network.

$$\begin{aligned} \sum_{i:(i,j) \in \Lambda} Y^{(i,j)} &\geq \Gamma_i^{(s)} & \forall i \in V : \exists (s, d) \in \Omega : s = i \\ \sum_{j:(i,j) \in \Lambda} Y^{(i,j)} &\geq \Gamma_j^{(d)} & \forall i \in V : \exists (s, d) \in \Omega : d = j. \end{aligned}$$

Our computational experiments indicate that node degree inequalities, generally, improve the lower bound, but often, result in worse upper bounds. This indicates that the application of heuristic algorithms may be helpful for this class of WDM optical network problems.

## 6.7 Computational Experiments

The procedures presented in this chapter were coded using Microsoft Visual C++, ILOG *Maestro* [1] library, and CPLEX 9.0. All computations were performed on a workstation with 2.66 GHz Xeon processor and 2GB RAM.

We have used four different sets of problems in our computational tests. The first two sets represent randomly defined problems, while the other two represent the existing NSFNET optical network and a set of problems defined and used in the study performed by Prathombutr et al. [50].

The first set of random instances is defined over a complete graph representing the physical layer, while the second set is defined over an incomplete graph representing the physical layer. For each set we have generated networks with 5, 7, 10, and 20 nodes with four different traffic patterns. Two traffic patterns are defined so that there is demand between all pairs of nodes, and the other two are defined so that there is demand only between 50% of nodes in the network. For each “density” of traffic demand matrix we defined two demand levels - high and low. High demand indicates that demand between pairs of nodes was uniformly generated in the interval  $[0.1, 1]$ , while low demand indicates that demand between pairs of nodes was uniformly generated in the interval  $[0.1, 0.5]$ . The second set of instances is generated in the same manner as the first set, only, in this case we have also randomly generated the physical network. (Physical network in this case is generated using uniform distribution to select the pairs of nodes that will be connected by a pre-specified number of arcs. Similarly, we have used uniform distribution to determine commodities with a positive demand.) We have performed computational tests for two different design objectives in these instances. The first design objective minimizes the total lost traffic in the network, while the second design objective minimizes the total number of transmitters and receivers used.

The third and the fourth set of test instances are from the study of Parthombutr et al. [50].

	1	2	3	4	5	6
1	0.00	0.73	0.46	0.54	0.56	0.44
2	0.75	0.00	0.92	0.73	0.77	1.13
3	0.35	0.69	0.00	0.69	0.50	0.77
4	0.90	0.67	1.00	0.00	0.65	0.48
5	0.83	0.79	0.38	0.56	0.00	0.56
6	0.60	0.52	0.92	0.94	0.77	0.00

Table 6.1: Demand matrix for 6-node network example in Figure 6.5

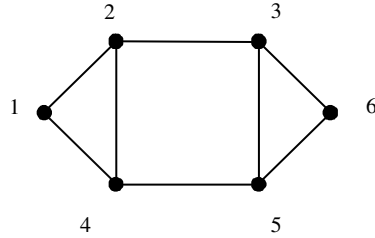


Figure 6.5: Physical topology of the 6-node optical network

These include a simple 6-node network and NSFNET network. Since instances defined in [50] use “granular” traffic in terms of the number of OC-1, OC-3, and OC-12 demands between pairs of nodes, we have modified these instances so that the total demand between two nodes is expressed as a fraction of capacity of a lightpath, which was set to OC-48 in the 6-node network, and OC-192 in the NSFNET network. In one instance (commodity (5, 1)), we have reduced original demand from 54 OC units to 48 OC units to accommodate for our assumption that demand between pairs of nodes does not exceed capacity of a lightpath. As in [50], 7 instances of 6 nodes network, and 6 instances of NSFNET network are tested for different number of wavelengths that can be supported by a single fiber and a different maximum number of transmitters and receivers that can be used at each node in the network. The physical topology of both networks is shown in Figures 6.5 and 6.6 respectively, and the corresponding modified demand matrices are shown in Tables 6.1 and 6.2. For these instances, we have performed tests using objective design that minimizes the total lost traffic in the network.

In all our test instances, the column generation is started with a single lightpath defined for each pair of nodes (determined using the shortest path algorithm, with the length of the path defined by the number of physical hops used by the lightpath), and single flow path for

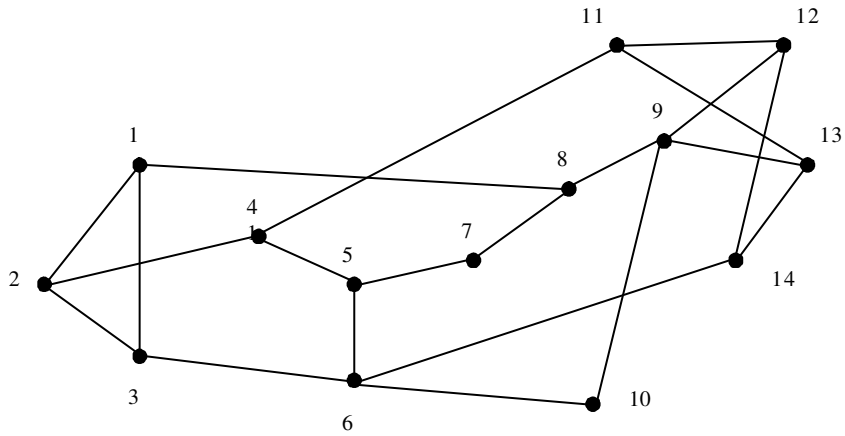


Figure 6.6: Physical topology of the NSFNET optical network

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0.00	0.11	0.03	0.15	0.08	0.17	0.14	0.13	0.18	0.09	0.13	0.19	0.26	0.17
2	0.22	0.00	0.20	0.26	0.06	0.18	0.09	0.22	0.09	0.14	0.10	0.04	0.14	0.10
3	0.15	0.13	0.00	0.21	0.18	0.13	0.02	0.29	0.16	0.17	0.27	0.17	0.06	0.18
4	0.05	0.08	0.13	0.00	0.28	0.13	0.11	0.24	0.16	0.22	0.14	0.27	0.20	0.17
5	0.28	0.06	0.26	0.23	0.00	0.12	0.21	0.27	0.22	0.17	0.16	0.21	0.14	0.01
6	0.29	0.14	0.32	0.07	0.25	0.00	0.20	0.15	0.23	0.11	0.06	0.16	0.08	0.28
7	0.18	0.09	0.14	0.13	0.18	0.24	0.00	0.17	0.17	0.07	0.16	0.18	0.19	0.19
8	0.10	0.12	0.07	0.19	0.11	0.32	0.20	0.00	0.13	0.08	0.22	0.29	0.04	0.10
9	0.24	0.15	0.23	0.26	0.11	0.15	0.28	0.08	0.00	0.26	0.11	0.13	0.16	0.26
10	0.22	0.07	0.06	0.17	0.17	0.15	0.17	0.06	0.03	0.00	0.14	0.28	0.27	0.27
11	0.23	0.16	0.27	0.24	0.06	0.13	0.24	0.11	0.15	0.14	0.00	0.20	0.21	0.25
12	0.33	0.13	0.21	0.19	0.16	0.17	0.07	0.21	0.14	0.13	0.06	0.00	0.05	0.15
13	0.26	0.20	0.21	0.21	0.16	0.15	0.03	0.17	0.17	0.16	0.21	0.11	0.00	0.24
14	0.25	0.20	0.11	0.09	0.19	0.16	0.08	0.09	0.18	0.19	0.15	0.08	0.27	0.00

Table 6.2: Demand matrix for the NSFNET network shown in Figure 6.6

each commodity. Additionally, our initial computational experiments indicated that use of the minimum parent lower bound in the node selection phase of the branching provides better results than the depth first rule. We have, therefore, used the minimum parent lower bound for selection of branching nodes in all our computational experiments.

In order to improve upper bounds obtained by our branch-and-price algorithms, we have included the use of the local MIP optimizer at the root node of the branch-and-bound tree with a 600 second CPU time limit. Also, in the branch-and-price algorithm for the MIP-PATH-GLOBAL formulation, we have included the use of the local MIP optimizer at all nodes of the branch-and-bound tree at which both global and local lightpath variables are integer, but there are fractional flow variables.

Tables 6.3 and 6.4 provide results for the randomly defined networks that use complete graph in the physical layer. These results indicate that, as expected, the branch-and-price algorithm on the MIP-PATH-LOCALw formulation requires longer computational times to solve the problems. Actually, this procedure solves to optimality only one test instance within 3600 second CPU time limit. The other two branch-and-price algorithms (for the formulations MIP-PATH-LOCAL and MIP-PATH-GLOBAL), on the other hand, solved 6 out of 8 instances in the networks with 5 and 7 nodes for the first objective function. For the second objective function, branch-and-price algorithm for MIP-PATH-LOCAL solved 1 problem and the branch-and-price algorithm for MIP-PATH-GLOBAL solved 2 problems. We can see that in all solved instances bound provided by the branch-and-price algorithms for MIP-PATH-LOCAL and MIP-PATH-GLOBAL are the same, indicating that the solutions found by these procedures are optimal. (Recall that solutions provided by the branch-and-price algorithm for MIP-PATH-LOCAL represent upper bounds for the original problem, while the solutions provided by the branch-and-price algorithm for MIP-PATH-GLOBAL represent lower bounds for the original problem. This means that we have an optimal solution if the bounds provided by these two procedures are equal.) We can also see that in terms of upper bounds branch-and-price algorithm for the MIP-PATH-LOCAL formulation outperforms the branch-and-price algorithm for the MIP-PATH-GLOBAL formulation. This may appear surprising



considering the fact that the (final) solutions provided by the branch-and-price algorithm for the MIP-PATH-GLOBAL formulation represent a lower bound for the original problem. However, these results may indicate that a better set of lightpaths was generated at the root node of the algorithm using the MIP-PATH-LOCAL formulation. This is likely to be due to the fact that the local lightpath variables in the MIP-PATH-LOCAL formulation are defined as binary variables, while in the MIP-PATH-GLOBAL formulation these variables are defined as general integers. Consequently, we can expect to have larger number of lightpaths (that differ from the lightpaths in the initial set) generated at the root node of the branch-and-price algorithm for the MIP-PATH-LOCAL formulation. As a result, in the case of MIP-PATH-LOCAL formulation, the search space corresponding to the problem solved by the local optimizer has a higher chance to include larger number of good candidate lightpaths.

Tables 6.5 and 6.6 provide results for the randomly defined networks that use incomplete graph in the physical layer. These results indicate the same pattern in the performance of our branch-and-price algorithms as in the case of the first set of instances defined over the complete physical networks.

The results for test instances defined in the study of Pathombutr et al. [50] are shown in Tables 6.7 and 6.8. We can see that within a specified CPU time limit, our branch-and-price algorithms were not able to find optimal solutions for any of these instances. In this case, the upper bounds provided by the branch-and-price algorithm for the MIP-PATH-GLOBAL formulation were, in most instances, better than the upper bounds provided by the branch-and-price algorithm for the MIP-PATH-LOCAL formulation. Most likely, the reason for this pattern, is that in most of these instances the branch-and-price algorithm for the MIP-PATH-LOCAL formulation did not complete column generation at the root node, and the algorithm did not get a chance to have any benefits from the local optimizer otherwise applied to the root node.

**Use of Valid Inequalities.** In the second stage of our computational experiments we have tested the impact of two classes of valid inequalities on the performance of the proposed branch-and-price algorithms. We have performed this test on the selected set of 8 instances defined over

the complete physical network and 8 instances over the incomplete physical network.

The results of our computational experiments with the lifted cover inequalities (LCIs) applied within the branch-and-price framework for MIP-PATH-LOCAL<sub>w</sub> and MIP-PATH-LOCAL are shown in Tables 6.9 and 6.10. We can see that the use of LCIs for the MIP-PATH-LOCAL<sub>w</sub> formulation never improves the lower bounds, and results in the worse upper bounds in two test instances. Use of LCIs for the MIP-PATH-LOCAL formulation provides similar results. That is, in two test instances, lower bounds became worse, while an upper bound was improved and deteriorated on one instance each. As we explained earlier, these results are not surprising given the fact that the LCIs are applied to the large number of lightpaths and that there is no guarantee that multiple identical lightpaths will not be added to the restricted master model, thus cancelling the effect of the added LCIs.

Our second test with the use of valid inequalities involved use of node degree inequalities in the branch-and-price framework for the MIP-PATH-LOCAL and MIP-PATH-GLOBAL formulations. (We did not test use of node degree inequalities for the MIP-PATH-LOCAL<sub>w</sub> formulation, since the corresponding branch-and-price algorithm has shown consistently poor performance in all previous experiments.) The results of these test are shown in Tables 6.11 and 6.12. We can see that the use of node degree inequalities resulted in the improved lower bounds in 14 out of 16 instances both for MIP-PATH-LOCAL and MIP-PATH-GLOBAL. On the other hand, use of these inequalities caused deterioration of the quality of the upper bounds in 6 out of 16 instances both for MIP-PATH-LOCAL and MIP-PATH-GLOBAL. These results suggest that the node degree inequalities can provide better results for the WDM optical network design problem, but that, also, we may also benefit from use of custom heuristic procedures that could improve upper bounds.

**Summary of Computational Results.** The summary of all computational results presented in this chapter is provided in Tables 6.13 - 6.16. These tables provide average percentage gaps and CPU times for each type of network used in our test instances. The percentage gaps for the first objective function were calculated as  $\frac{UB-LB}{totaldemand-LB} * 100\%$ . In other words, we provide the percentage gaps with the respect to the total traffic served, instead of the total traffic lost.

(We do this only because the straightforward computation of the gap in the form  $\frac{UB-LB}{LB}$  is not practical for these problems, since the LB is equal to zero in a large number of instances.) The percentage gaps for the second objective function were calculated as  $\frac{UB-LB}{LB}$ .

The results in Table 6.13 provide a summary of the results for the objective of minimizing the total lost traffic in the network. These results clearly indicate superior performance of the branch-and-price algorithms for MIP-PATH-LOCAL and MIP-PATH-GLOBAL. Further, these results indicate that out of the three branch-and-price procedures proposed in this dissertation, the branch-and-price algorithm for the MIP-PATH-GLOBAL formulation is the best choice for obtaining good bounds for the WDM optical network design problem. The branch-and-price algorithm for the MIP-PATH-LOCAL formulation is second best, and is a better choice than the branch-and-price algorithm for the MIP-PATH-LOCALw formulation when we are looking for good upper bounds.

Table 6.14 provides results for the objective of minimizing the number of transmitters and receivers in the network, and indicates similar findings as in the case of the first objective function.

The results in Table 6.15 provide a summary of the use of LCIs in combination with our branch-and-price algorithms for the MIP-PATH-LOCALw and MIP-PATH-LOCAL formulations. We see that the average percentage gap slightly deteriorates with the use of LCIs, which means that these inequalities should not be used in the branch-and-price framework for the WDM optical network design problem.

Finally, Table 6.16 provides a summary of results for use of node degree inequalities in combination with our branch-and-price algorithms for the MIP-PATH-LOCAL and MIP-PATH-GLOBAL formulations. We can see that the use of these inequalities typically provides lower percentage gaps, except in the case of instances defined over complete physical networks and solved using the branch-and-price algorithm for MIP-PATH-GLOBAL. However, as we mentioned earlier, this increase in the percentage gap is only due to the inferior upper bounds. Overall, the use of these inequalities improves lower bounds, which indicates that we should use these inequalities when solving WDM optical network design problems in networks that guarantee service of the

V	C	D	B&P (LOCALw)			B&P (LOCAL)			B&P (GLOBAL)		
			LB	UB	CPU (sec)	LB	UB	CPU (sec)	LB	UB	CPU (sec)
5	20	H	0.11	0.88	3602.16	0.62	0.62	2.31	0.62	0.62	2283.58
5	20	L	0.00	0.89	3602.58	0.00	0.00	0.34	0.00	0.00	0.13
5	10	H	0.00	0.16	3600.58	0.00	0.00	0.17	0.00	0.00	0.06
5	10	L	0.00	0.00	3.22	0.00	0.00	0.19	0.00	0.00	0.05
7	42	H	2.28	6.97	3600.58	4.32	5.32	3601.36	4.29	5.29	3602.26
7	42	L	0.00	4.25	3600.33	0.00	0.15	3600.88	0.00	0.00	312.05
7	21	H	0.19	1.90	3600.73	0.70	0.70	11.23	0.70	0.70	3617.31
7	21	L	0.00	1.23	3600.39	0.00	0.00	2.06	0.00	0.00	0.67
10	90	H	17.65	24.50	3600.20	21.14	22.89	3602.56	21.15	23.14	3601.14
10	90	L	0.21	14.18	3600.17	5.18	7.14	3602.33	5.20	6.92	3879.33
10	45	H	1.47	5.88	3600.23	2.41	4.33	3600.66	2.55	4.17	3603.84
10	45	L	0.00	3.81	3600.20	0.00	0.43	3601.83	0.00	0.14	3602.39
20	380	H	149.39	156.67	3607.74	153.68	155.44	3601.36	152.44	155.04	4137.05
20	380	L	54.13	86.84	3610.52	74.87	86.84	3601.99	70.49	74.50	3653.72
20	190	H	48.07	57.09	3702.78	54.11	57.29	3601.80	52.76	57.01	3626.23
20	190	L	7.19	31.17	3602.48	18.97	32.80	3600.38	16.60	22.26	3672.38

Table 6.3: Minimizing the lost traffic in the network. Complete physical network with 2 fibers (fiber capacity: 2 lightpaths) between all pairs of nodes.

V	C	D	B&P (LOCALw)			B&P (LOCAL)			B&P (GLOBAL)		
			LB	UB	CPU (sec)	LB	UB	CPU (sec)	LB	UB	CPU (sec)
5	20	H	22.50	40.00	3602.53	24.94	28.00	3609.70	26.00	28.00	3603.45
5	20	L	13.78	40.00	3602.25	17.70	20.00	3607.13	18.11	20.00	3607.08
5	10	H	14.50	20.00	3606.13	16.00	16.00	11.64	16.00	16.00	1.91
5	10	L	9.18	20.00	3610.53	12.21	14.00	3615.76	14.00	14.00	62.14
7	42	H	44.64	84.00	3600.22	51.57	58.00	3601.03	52.60	60.00	3608.95
7	42	L	25.66	84.00	3600.22	34.83	42.00	3600.41	35.23	42.00	3604.11
7	21	H	22.66	42.00	3600.41	27.82	30.00	3601.13	28.38	30.00	3619.83
7	21	L	13.76	40.00	3601.01	19.85	22.00	3601.17	20.38	22.00	3611.19
10	90	H	96.24	180.00	3600.14	112.13	126.00	3603.13	112.54	130.00	3603.99
10	90	L	53.18	180.00	3600.11	74.90	90.00	3603.09	75.00	86.00	3601.58
10	45	H	46.20	84.00	3600.26	56.34	68.00	3600.86	57.10	66.00	3602.17
10	45	L	26.10	70.00	3600.17	38.80	48.00	3603.38	39.39	48.00	3604.61
20	380	H	419.78	760.00	3600.91	533.53	760.00	3601.34	488.23	562.00	3668.94
20	380	L	228.26	760.00	3601.03	386.98	760.00	3600.36	322.34	374.00	6517.19
20	190	H	209.32	376.00	3714.16	259.97	380.00	3601.91	247.19	294.00	3896.58
20	190	L	114.00	-	3601.50	174.81	380.00	3600.86	165.23	208.00	4359.09

Table 6.4: Minimizing the total number of transmitters and receivers in the network. Complete physical network with 2 fibers (fiber capacity: 2 lightpaths) between all pairs of nodes.

entire traffic demand.

## 6.8 Conclusion

In this chapter, we have presented two path-based formulations for the WDM optical network design problem. We have discussed the complexity behind the design of branch-and-price algorithms for this problem, and then developed two branch-and-price procedures for the proposed formulations MIP-PATH-LOCAL and MIP-PATH-GLOBAL. We have also shown that, with a few modifications, the branch-and-price algorithm for MIP-PATH-LOCAL can be used to obtain

V	A	C	D	B&P (LOCALw)			B&P (LOCAL)			B&P (GLOBAL)		
				LB	UB	CPU (sec)	LB	UB	CPU (sec)	LB	UB	CPU (sec)
5	10	15	H	0.38	0.89	3604.70	0.69	0.69	1.06	0.69	0.69	1953.45
5	10	15	L	0.00	0.00	53.97	0.00	0.00	0.22	0.00	0.00	0.09
7	15	30	H	1.71	4.61	3601.27	2.40	3.44	3601.81	2.04	3.10	3607.97
7	15	30	L	0.00	2.60	3600.49	0.00	0.16	3600.72	0.00	0.00	9.39
10	20	40	H	1.71	4.25	3600.51	2.41	2.95	3601.33	1.86	3.05	3605.39
10	20	40	L	0.00	2.61	3600.28	0.00	0.16	3601.72	0.00	0.19	3601.51
20	30	60	H	4.68	6.55	3600.33	4.68	6.19	3605.47	4.68	6.42	3609.92
20	30	60	L	0.00	4.18	3600.33	0.00	1.20	3619.41	0.00	1.32	3628.48

Table 6.5: Minimizing the lost traffic in the network. Incomplete physical network with 2 fibers (fiber capacity: 20 lightpaths) on all arcs.

V	A	C	D	B&P (LOCALw)			B&P (LOCAL)			B&P (GLOBAL)		
				LB	UB	CPU (sec)	LB	UB	CPU (sec)	LB	UB	CPU (sec)
5	10	15	H	20.66	30.00	3608.97	20.40	22.00	3621.63	22.00	22.00	57.48
5	10	15	L	14.36	30.00	3607.59	14.79	16.00	3617.55	16.00	16.00	92.22
7	15	30	H	37.48	60.00	3601.30	41.69	48.00	3602.05	42.55	48.00	3611.48
7	15	30	L	22.28	56.00	3601.06	27.95	34.00	3601.33	28.45	32.00	3607.03
10	20	40	H	50.16	76.00	3600.41	56.29	66.00	3600.86	57.26	66.00	3603.06
10	20	40	L	28.18	66.00	3600.36	39.22	50.00	3602.59	39.57	50.00	3601.76
20	30	60	H	70.30	120.00	3600.36	87.11	112.00	3642.34	87.30	106.00	3648.08
20	30	60	L	38.02	120.00	3600.39	61.93	84.00	3765.64	62.19	84.00	3795.56

Table 6.6: Minimizing the total number of transmitters and receivers in the network. Incomplete physical network with 2 fibers (fiber capacity: 20 lightpaths) on all arcs.

T/R Nbr	Wav. Nbr	B&P (LOCALw)			B&P (LOCAL)			B&P (GLOBAL)		
		LB	UB	CPU (sec)	LB	UB	CPU (sec)	LB	UB	CPU (sec)
3	3	3.47	6.46	3602.08	5.10	6.02	3606.42	5.09	6.02	3614.95
4	3	0.28	3.14	3601.84	0.77	2.78	3601.02	1.12	2.68	3605.81
5	3	0.28	2.81	3601.64	0.44	2.43	3600.94	0.28	1.29	3616.30
7	3	0.28	2.81	3601.78	0.44	2.45	3600.75	0.28	0.92	3618.39
3	4	3.60	6.46	3602.02	5.10	6.02	3605.81	5.03	6.02	3615.16
4	4	0.17	3.02	3601.80	0.75	2.63	3602.53	0.95	2.48	3608.02
5	4	0.00	1.31	3601.08	0.00	0.35	3600.69	0.00	0.91	3611.61

Table 6.7: Minimizing the lost traffic in the network. The 6-node network with a single fiber on each arc.

T/R Nbr	Wav. Nbr	B&P (LOCALw)			B&P (LOCAL)			B&P (GLOBAL)		
		LB	UB	CPU (sec)	LB	UB	CPU (sec)	LB	UB	CPU (sec)
3	3	0.00	19.26	3600.35	4.13	8.89	3606.41	3.99	8.99	3655.20
4	3	0.00	16.82	3600.32	0.00	8.98	3626.69	0.00	6.71	3641.36
5	3	0.00	15.39	3600.35	0.00	5.32	3641.06	0.00	2.62	3612.26
4	4	0.00	17.04	3600.32	0.00	8.46	3614.05	0.00	9.30	3618.47
5	4	0.00	14.91	3601.21	0.00	4.93	3609.97	0.00	2.39	3614.84
6	4	0.00	13.51	3600.32	0.00	3.26	3620.81	0.00	0.21	3917.52

Table 6.8: Minimizing the lost traffic in the network. The NSFNET network with a single fiber on each arc.

V	C	D	B&P&C (LOCALw)			B&P&C (LOCAL)		
			LB	UB	CPU (sec)	LB	UB	CPU (sec)
5	20	L	0.00	0.89	3600.19	0.00	0.00	0.34
5	10	H	0.00	0.16	3600.14	0.00	0.00	0.19
7	42	L	0.00	4.25	3600.22	0.00	0.10	3603.67
7	21	H	0.19	1.90	3600.38	0.70	0.70	19.78
10	90	L	0.21	14.18	3600.39	5.18	6.80	3610.69
10	45	H	1.47	5.88	3600.16	2.38	4.03	3610.19
20	380	L	54.13	86.84	3616.20	74.87	86.84	3601.03
20	190	H	48.07	57.18	3716.98	54.11	57.29	3600.66

Table 6.9: Minimizing the lost traffic in the network (use of lifted cover inequalities). Selected instances for a complete physical network with 2 fibers (fiber capacity: 2 lightpaths) between all pairs of nodes.

V	A	C	D	B&P&C (LOCALw)			B&P&C (LOCAL)		
				LB	UB	CPU (sec)	LB	UB	CPU (sec)
5	10	15	H	0.38	0.89	3601.69	0.69	0.69	3.89
5	10	15	L	0.00	0.00	51.50	0.00	0.00	0.22
7	15	30	H	1.71	4.61	3601.36	2.40	3.44	3603.19
7	15	30	L	0.00	2.60	3600.20	0.00	0.16	3615.42
10	20	40	H	1.71	4.25	3600.33	2.36	3.23	3616.08
10	20	40	L	0.00	2.61	3600.28	0.00	0.16	3620.20
20	30	60	H	4.68	6.61	3600.39	4.68	6.48	3610.41
20	30	60	L	0.00	4.18	3600.27	0.00	4.18	3605.97

Table 6.10: Minimizing the lost traffic in the network (use of lifted cover inequalities). Incomplete physical network with 2 fibers (fiber capacity: 20 lightpaths) on all arcs.

V	C	D	B&P (LOCAL)			B&P (GLOBAL)		
			LB	UB	CPU (sec)	LB	UB	CPU (sec)
5	20	L	18.28	20.00	3604.41	20.00	20.00	1256.78
5	10	H	16.00	16.00	0.75	16.00	16.00	0.30
7	42	L	35.08	42.00	3601.09	35.39	42.00	3604.34
7	21	H	30.00	32.00	3601.13	30.00	30.00	1205.95
10	90	L	74.94	90.00	3602.86	75.10	88.00	3602.31
10	45	H	62.00	70.00	3601.14	62.00	72.00	3602.88
20	380	L	384.12	760.00	3601.41	322.38	760.00	3665.31
20	190	H	261.50	380.00	3601.39	252.56	290.00	3657.64

Table 6.11: Minimizing the total number of transmitters and receivers in the network (use of node degree inequalities). Selected instances for a complete physical network with 2 fibers (fiber capacity: 2 lightpaths) between all pairs of nodes.

V	A	C	D	B&P (LOCAL)			B&P (GLOBAL)		
				LB	UB	CPU (sec)	LB	UB	CPU (sec)
5	10	15	H	22.00	22.00	69.52	22.00	22.00	2.27
5	10	15	L	15.35	16.00	3612.69	16.00	16.00	21.09
7	15	30	H	48.00	52.00	3601.16	48.00	52.00	3604.64
7	15	30	L	28.23	32.00	3601.31	28.84	32.00	3608.08
10	20	40	H	64.00	68.00	3601.67	64.00	68.00	3603.47
10	20	40	L	39.75	48.00	3601.89	40.14	50.00	3604.45
20	30	60	H	92.19	120.00	3601.06	91.95	106.00	3618.76
20	30	60	L	62.83	90.00	3670.02	63.12	90.00	3642.05

Table 6.12: Minimizing the total number of transmitters and receivers in the network (use of node degree inequalities). Incomplete physical network with 2 fibers (fiber capacity: 20 lightpaths) on all arcs.

Net. Type	B&P (LOCALw)		B&P (LOCAL)		B&P (GLOBAL)	
	Avg. Gap	Avg. CPU (sec)	Avg. Gap	Avg. CPU (sec)	Avg. Gap	Avg. CPU (sec)
Complete	24.04 %	3383.43	7.02 %	2251.97	4.17 %	2474.51
Incomplete	14.49 %	3157.73	3.06 %	2703.97	3.41 %	2502.03
6 nodes	13.46 %	3601.75	7.62 %	3602.59	5.85 %	3612.89
NSFNET	54.03 %	3600.48	20.33 %	3619.83	15.05 %	3676.61
Overall	24.84 %	3411.13	8.44 %	2827.04	6.09 %	2890.76

Table 6.13: Summary: Minimizing the lost traffic in the network.

Net. Type	B&P (LOCALw)		B&P (LOCAL)		B&P (GLOBAL)	
	Avg. Gap	Avg. CPU (sec)	Avg. Gap	Avg. CPU (sec)	Avg. Gap	Avg. CPU (sec)
Complete	138.62 %	3608.85	29.43 %	3378.93	13.04 %	3410.8
Incomplete	104.7 %	3602.56	20.22 %	3631.75	15.43 %	2752.08
Overall	127.31 %	3606.75	26.36 %	3463.20	13.84 %	3191.23

Table 6.14: Summary: Minimizing the total number of transmitters and receivers in the network.(We have assumed an upper bound of 380 for one instance of LOCALw where we did not have an integer solution.)

either optimal solutions or approximate upper bound solutions.

Our computational experiments with the two variations of the branch-and-price algorithm for the MIP-PATH-LOCAL formulation, however, indicated that the approximate upper bound branch-and-price algorithm provides better results in shorter CPU times. Moreover, in all instances where the optimal solution is known, the approximate branch-and-price algorithm found an optimal solution indicating that this procedure can be used to obtain a high quality upper bounds for the WDM optical network design problem.

For the second proposed formulation, MIP-PATH-GLOBAL, we have proposed an approximate lower-bound branch-and-price algorithm that provides the best quality of solutions out of three branch-and-price procedures proposed in this dissertation. Again, as in the case of an approximate branch-and-price algorithm for the MIP-PATH-LOCAL formulation, the approximate algorithm for the MIP-PATH-GLOBAL formulation found optimal solution for all instances where the optimal solution is known.

Given the fact that our approximate branch-and-price algorithms for the MIP-PATH-LOCAL and MIP-PATH-GLOBAL formulations provide valid upper and lower bounds for the WDM optical network design problem, we suggest the combined use of these procedures to find provably optimal solutions.

Finally, we have proposed and computationally tested two classes of valid inequalities that

Net. Type	B&P&C (LOCALw)		B&P&C (LOCAL)	
	Avg. Gap	Avg. CPU (sec)	Avg. Gap	Avg. CPU (sec)
Complete	27.17 % (27.15%)	3616.83 (3614.74)	6.69 % (7.12%)	2255.82 (2252.43)
Incomplete	14.52 % (14.49%)	3157 (3157.73)	5.39 % (3.06%)	2709.42 (2703.97)
Overall	20.85 % (20.82%)	3386.92 (3386.24)	6.04 % (5.04%)	2482.62 (2478.20)

Table 6.15: Summary: Minimizing the total lost traffic in the network. Lifted Cover Inequalities. (The numbers in parenthesis indicate results for the corresponding branch-and-price algorithms when lifted cover inequalities are not used.)

Net. Type	B&P (LOCAL)		B&P (GLOBAL)	
	Avg. Gap	Avg. CPU (sec)	Avg. Gap	Avg. CPU (sec)
Complete	26.5 % (28.10%)	3151.77 (3153.32)	25.32 % (12.57%)	2574.44 (3556.31)
Incomplete	15.79 % (20.22%)	3169.91 (3631.75)	13.5 % (15.43%)	2713.1 (2752.08)
Overall	21.15 % (24.16%)	3160.84 (3392.54)	19.41 % (14.00%)	2643.77 (3154.2)

Table 6.16: Summary: Minimizing the total number of transmitters and receivers in the network. Node Degree Inequalities. (The numbers in parenthesis indicate results for the corresponding branch-and-price algorithms when node degree inequalities are not used.)

can be used in combination with our branch-and-price algorithms.

The first class of valid inequalities represents lifted cover inequalities that can be applied to our exact and approximate branch-and-price procedures for the MIP-PATH-LOCAL formulation. Our computational results indicate that addition of the lifted cover inequalities does not improve lower bounds. We believe that the reason for this result is a fact that these inequalities are applied to individual lightpaths. Consequently, a large percentage of the computational time is spent on generation of these inequalities. Additionally, once an LCI is added for a specific lightpath, there is no guarantee that an identical new lightpath will not be generated in the next column generation step, thus cancelling the effect of the LCIs already added to the restricted master model.

The second class of valid inequalities tested in this chapter represents node degree inequalities that can be used in situations where all the demand must be served in a given optical network. Our computational tests indicate that these inequalities improve the lower bounds provided by our branch-and-price procedures, but also have a tendency to make the problem harder in terms of finding good upper bounds. We suspect that in combination with fast custom upper bound heuristics for the WDM optical network design, use of node degree inequalities can lead to higher quality solutions for this problem.

In the next chapter, we will discuss the application of our approximate branch-and-price



algorithms for the MIP-PATH-LOCAL and MIP-PATH-GLOBAL formulations for WDM optical networks with different network settings.

## Chapter 7

### Extensions to WDM Optical Networks with Alternative Design Requirements

#### 7.1 Introduction

In this chapter we discuss the application of our approximate branch-and-price procedures for the design of WDM optical networks where design requirements differ from our original problem defined and studied in Chapters 5 and 6. The requirements that we focus on include networks where flow bifurcation is allowed and networks where there is ample fiber capacity, so there is no constraint on the number of lightpaths that can be established on any physical link. We also look at WDM optical networks where we can make both assumptions. That is, we allow bifurcation of flow *and* assume that an infinite number of wavelengths is available in the network. These three special cases of WDM optical networks are of special interest for practical purposes. The first case, which allows bifurcation of traffic is directly applicable to all situations where the network is used for the ATM traffic. Notice that for this special network setting, our implementation of the MIP-PATH-GLOBAL formulation and the corresponding branch-and-price algorithm provides optimal solutions to the original problem. The second case is a common assumption used for planning purposes, and our procedures can be used to provide lower and upper bounds. The third case can be used in networks where it is reasonable to assume that bifurcation of flow is acceptable *and* that there is ample fiber capacity. In this case, again, the branch-and-price algorithm for the MIP-PATH-GLOBAL formulation is an exact solution procedure, and the solutions obtained using this algorithm are optimal for the original WDM optical network design problem.

#### 7.2 WDM Optical Networks with Traffic Bifurcation

The MIP-PATH-LOCAL and MIP-PATH-GLOBAL mathematical formulations that we presented for the networks with the U/U/N setting are directly applicable to networks where bifurca-

tion of traffic is allowed. However, we do need to make several modifications. In both formulations we need to relax integrality constraints on the flow path variables. However, relaxation of the integrality constraint on the flow variables in the MIP-PATH-LOCAL formulation means that we must now impose an integrality constraint on the lightpath variables  $X_z$  (recall that the constraint  $X_z - \sum_{p:z \in p} f_p^{(s,d)} \geq 0 \quad \forall z \in Z, (s,d) \in \Omega$  was ensuring integrality of the variables  $X_z$  as long as the flow path variables  $f_p^{(s,d)}$  had integer values). While this modification of the MIP-PATH-LOCAL formulation may appear to be minor, it actually significantly affects the branch-and-price algorithm that we proposed for this formulation. That is, since the flow path variables are not integer any longer, we need to adopt a new branching strategy that will ensure integrality of the  $X_z$  variables. For this purpose, we use the same branching strategy that we used for the  $X_z$  variables in the branch-and-price algorithm for the MIP-PATH-LOCALw formulation.

In order to apply our MIP-PATH-GLOBAL formulation to the WDM optical networks with the U/U/A setting, we just need to relax the integrality constraints on the flow path variables, and exclude the “packing” constraints that we used to ensure no flow bifurcation. As for our branch-and-price algorithm that corresponds to this formulation, no further modifications are necessary.

The new formulations for the case of bifurcation allowed are as follows.

#### MIP-PATH-LOCAL<sup>(A)</sup> Formulation

$$\text{Min} \quad \sum_{(s,d) \in \Omega} T^{(s,d)} H^{(s,d)} \quad (7.1)$$

Subject to:

$$\sum_{z:O(z)=i} X_z \leq \Delta_t^i \quad \forall i \in V \quad (7.2)$$

$$\sum_{z:D(z)=j} X_z \leq \Delta_r^j \quad \forall j \in V \quad (7.3)$$

$$\sum_{z:(l,m) \in z} X_z \leq L_{lm} \quad \forall (l,m) \in A \quad (7.4)$$

$$X_z - \sum_{p:z \in p} f_p^{(s,d)} \geq 0 \quad \forall z \in Z, (s,d) \in \Omega \quad (7.5)$$

$$X_z - \sum_{(s,d) \in \Omega, p:z \in p} T^{(s,d)} f_p^{(s,d)} \geq 0 \quad \forall z \in Z \quad (7.6)$$

$$\sum_{p \in P^{(s,d)}} f_p^{(s,d)} + H^{(s,d)} = 1 \quad \forall (s,d) \in \Omega \quad (7.7)$$

$$f_p^{(s,d)} \in R_+^1 \quad \forall p \in P^{(s,d)}, (s,d) \in \Omega \quad (7.8)$$

$$H^{(s,d)} \in R_+^1 \quad \forall (s,d) \in P^{(s,d)} \quad (7.9)$$

$$X_z \in B^1 \quad \forall z \in Z \quad (7.10)$$

### MIP-PATH-GLOBAL<sup>(A)</sup> Formulation

$$\text{Min} \sum_{(s,d) \in \Omega} T^{(s,d)} H^{(s,d)} \quad (7.11)$$

Subject to:

$$\sum_{j:(i,j) \in \Lambda} Y^{(i,j)} \leq \Delta_t^i \quad \forall i \in V \quad (7.12)$$

$$\sum_{i:(i,j) \in \Lambda} Y^{(i,j)} \leq \Delta_r^j \quad \forall j \in V \quad (7.13)$$

$$Y^{(i,j)} - \sum_{z:O(z)=i,D(z)=j} X_z^{(i,j)} = 0 \quad \forall (i,j) \in \Lambda \quad (7.14)$$

$$\sum_{(i,j) \in \Lambda, z:(l,m) \in z} X_z^{(i,j)} \leq L_{lm} \quad \forall (l,m) \in A \quad (7.15)$$

$$Y^{(i,j)} - \sum_{p:(i,j) \in p} f_p^{(s,d)} \geq 0 \quad \forall (i,j) \in \Lambda, (s,d) \in \Omega \quad (7.16)$$

$$Y^{(i,j)} - \sum_{(s,d) \in \Omega, p:(i,j) \in p} T^{(s,d)} f_p^{(s,d)} \geq 0 \quad \forall (i,j) \in \Lambda \quad (7.17)$$

$$\sum_{p \in P^{(s,d)}} f_p^{(s,d)} + H^{(s,d)} = 1 \quad \forall (s,d) \in \Omega \quad (7.18)$$

$$f_p^{(s,d)} \in R_+^1 \quad \forall p \in P^{(s,d)}, (s,d) \in \Omega \quad (7.19)$$

$$H^{(s,d)} \in R_+^1 \quad \forall (s,d) \in \Omega \quad (7.20)$$

$$Y^{(i,j)} \in R_+^1 \quad \forall (i,j) \in \Lambda \quad (7.21)$$

$$X_z^{(i,j)} \in Z_+^1 \quad \forall z \in Z, (i,j) \in \Lambda \quad (7.22)$$

Our computational results with the branch-and-price algorithms for MIP-PATH-LOCAL and MIP-PATH-GLOBAL are shown in Tables 7.1 - 7.4. We can see that the number of instances solved by these procedures is higher than the number of instances solved in the networks where bifurcation of flow is not allowed (presented in the previous chapter). The summary of the average percentage gaps and the average CPU times for the two objective functions used are shown in Tables 7.5 and 7.6. We can see that these averages are lower than in the case of WDM optical

networks without bifurcation of flow. Specifically, the average percentage gap for MIP-PATH-LOCAL decreased from 20.82% to 3.48% for the objective of minimizing the lost traffic in the network, and 21.15% to 19.59% for the objective of minimizing the total number of transmitters and receivers in the network. The average percentage gap for MIP-PATH-GLOBAL decreased from 5.04% to 0.77% for the objective of minimizing the total lost traffic in the network. For the second objective, minimizing the total number of transmitters and receivers, the percentage gap decreased from 19.41% to 19.39%. As before, these results indicate superior performance of the branch-and-price procedure for the MIP-PATH-GLOBAL formulation.

V	C	D	B&P (LOCAL)			B&P (GLOBAL)		
			LB	UB	CPU (sec)	LB	UB	CPU (sec)
5	20	L	0.00	0.00	0.30	0.00	0.00	0.125
5	10	H	0.00	0.00	0.17	0.00	0.00	0.062
7	42	L	0.00	0.00	4.61	0.00	0.00	4.141
7	21	H	0.19	0.19	5.09	0.19	0.19	47.844
10	90	L	5.18	5.20	3603.11	5.20	5.20	1053.55
10	45	H	2.35	2.83	3601.05	2.54	2.79	3604.03
20	380	L	71.90	86.84	3674.48	70.49	70.49	3656.03
20	190	H	53.20	57.29	3627.16	52.76	53.43	3624.42

Table 7.1: Minimizing the lost traffic in the network. Bifurcation of flow allowed. Selected instances for a complete physical network with 2 fibers (fiber capacity: 2 lightpaths) between all pairs of nodes.

V	A	C	D	B&P (LOCAL)			B&P (GLOBAL)		
				LB	UB	CPU (sec)	LB	UB	CPU (sec)
5	10	15	H	0.38	0.38	0.44	0.38	0.38	0.204
5	10	15	L	0.00	0.00	0.22	0.00	0.00	0.094
7	15	30	H	1.87	2.04	3601.91	2.04	2.04	382.922
7	15	30	L	0.00	0.00	2.20	0.00	0.00	0.688
10	20	40	H	1.80	2.12	3601.28	1.85	2.11	3602.72
10	20	40	L	0.00	0.16	3600.92	0.00	0.00	337.265
20	30	60	H	4.68	5.26	3605.19	4.68	5.63	3609.92
20	30	60	L	0.00	0.74	3604.47	0.00	0.97	3632.92

Table 7.2: Minimizing the lost traffic in the network. Bifurcation of flow allowed. Incomplete physical network with 2 fibers (fiber capacity: 20 lightpaths) on all arcs.

### 7.3 WDM Optical Networks with Infinite Number of Wavelengths

In Section 6.4.1, we have discussed one way of modelling WDM optical networks with infinite number of wavelengths. Another option is to try to work with the MIP-PATH-GLOBAL formulation with the local lightpaths variables  $X_z^{(i,j)}$  eliminated. This new formulation is equivalent

V	C	D	B&P (LOCAL)			B&P (GLOBAL)		
			LB	UB	CPU (sec)	LB	UB	CPU (sec)
5	20	L	17.38	20.00	3603.98	18.11	20.00	3607.22
5	10	H	15.28	16.00	3602.72	16.00	16.00	1.83
7	42	L	35.03	38.00	3600.81	35.23	38.00	3604.20
7	21	H	26.96	30.00	3601.74	28.38	30.00	3619.50
10	90	L	74.93	80.00	3603.64	75.00	80.00	3604.41
10	45	H	56.10	62.00	3601.22	57.10	62.00	3602.20
20	380	L	371.54	760.00	3651.31	322.34	760.00	3600.00
20	190	H	255.72	380.00	3664.13	247.19	380.00	3888.88

Table 7.3: Minimizing the total number of transmitters and receivers in the network. Bifurcation of flow allowed. Selected instances for a complete physical network with 2 fibers (fiber capacity: 2 lightpaths) between all pairs of nodes.

V	A	C	D	B&P (LOCAL)			B&P (GLOBAL)		
				LB	UB	CPU (sec)	LB	UB	CPU (sec)
5	10	15	H	20.28	22.00	3606.95	22.00	22.00	55.937
5	10	15	L	14.79	16.00	3601.22	16.00	16.00	91.078
7	15	30	H	41.27	44.00	3602.69	42.49	44.00	3610.97
7	15	30	L	28.18	30.00	3602.09	28.47	30.00	3607.66
10	20	40	H	56.49	62.00	3603.05	57.26	64.00	3603.86
10	20	40	L	39.37	46.00	3602.17	39.57	44.00	3604.73
20	30	60	H	87.11	100.00	3626.61	87.30	100.00	3646.2
20	30	60	L	61.97	82.00	3807.52	62.19	84.00	3787.38

Table 7.4: Minimizing the total number of transmitters and receivers in the network. Bifurcation of flow allowed. Incomplete physical network with 2 fibers (fiber capacity: 20 lightpaths) on all arcs.

Net. Type	B&P (LOCAL)		B&P (GLOBAL)	
	Avg. Gap	Avg. CPU (sec)	Avg. Gap	Avg. CPU (sec)
Complete	5.72 %	1814.5	0.32 %	1498.78
Incomplete	1.23 %	2252.08	1.21 %	1445.84
Overall	3.48 %	2033.29	0.77 %	1472.31

Table 7.5: Summary: Minimizing the lost traffic in the network. Bifurcation of flow allowed.

Net. Type	B&P (LOCAL)		B&P (GLOBAL)	
	Avg. Gap	Avg. CPU (sec)	Avg. Gap	Avg. CPU (sec)
Complete	26.25 %	3616.19	28.59 %	3191.03
Incomplete	12.93 %	3631.54	10.19 %	2750.98
Overall	19.59 %	3623.87	19.39 %	2971.01

Table 7.6: Summary: Minimizing the total number of transmitters and receivers in the network. Bifurcation of flow allowed.

to the one defined in Chapter 6 (equations (6.45) - (6.52)), except that we would also need “packing” constraints to ensure no bifurcation of flow. (Note that, as in Chapter 6, we ignore these packing constraints. Thus, MIP-PATH-GLOBAL gives a valid lower bound.)

Alternatively, we could work with the MIP-PATH-LOCAL formulation. In this case, we would just need to eliminate constraint on the number of lightpaths on physical links. This modification of the MIP-PATH-LOCAL formulation simplifies our branch-and-price algorithm in

the following way. Recall that the reduced cost of any flow path variables in the MIP-PATH-LOCAL formulation was calculated as:

$$RC_p^{(s,d)} = \sum_{z \in p, z \in Z^E} (r_z^{(s,d)} + T^{(s,d)}v_z) + \sum_{z \in p, z \in Z^N} (-a_i - b_j - \sum_{(l,m) \in z} d_{(l,m)}) - w^{(s,d)}.$$

Elimination of the constraint on the number of lightpaths that can be established over any given physical link implies that we do not have term  $\sum_{(l,m) \in z} d_{(l,m)}$  in the expression above, that is, we get:

$$RC_p^{(s,d)} = \sum_{z \in p, z \in Z^E} (r_z^{(s,d)} + T^{(s,d)}v_z) + \sum_{z \in p, z \in Z^N} (-a_i - b_j) - w^{(s,d)}.$$

This means that the ‘‘cost’’  $P_{z;(s,d)}^N$  of any new lightpath becomes:

$$P_{z;(s,d)}^N = -a_i - b_j$$

which can be directly calculated without having to solve the shortest path problem. Other than this, our branch-and-price algorithm remains the same as before.

### MIP-PATH-LOCAL<sup>(I)</sup> Formulation

$$\text{Min} \quad \sum_{(s,d) \in \Omega} T^{(s,d)} H^{(s,d)} \quad (7.23)$$

Subject to:

$$\sum_{z:O(z)=i} X_z \leq \Delta_t^i \quad \forall i \in V \quad (7.24)$$

$$\sum_{z:D(z)=j} X_z \leq \Delta_r^j \quad \forall j \in V \quad (7.25)$$

$$X_z - \sum_{p:z \in p} f_p^{(s,d)} \geq 0 \quad \forall z \in Z, (s,d) \in \Omega \quad (7.26)$$

$$X_z - \sum_{(s,d) \in \Omega, p:z \in p} T^{(s,d)} f_p^{(s,d)} \geq 0 \quad \forall z \in Z \quad (7.27)$$

$$\sum_{p \in P^{(s,d)}} f_p^{(s,d)} + H^{(s,d)} = 1 \quad \forall (s,d) \in \Omega \quad (7.28)$$

$$f_p^{(s,d)} \in B^1 \quad \forall p \in P^{(s,d)}, (s,d) \in \Omega \quad (7.29)$$

$$H^{(s,d)} \in R^1 \quad \forall (s,d) \in \Omega \quad (7.30)$$

$$0 \leq X_z \leq 1 \quad \forall z \in Z \quad (7.31)$$

### MIP-PATH-GLOBAL<sup>(I)</sup> Formulation

$$\text{Min} \quad \sum_{(s,d) \in \Omega} T^{(s,d)} H^{(s,d)} \quad (7.32)$$

Subject to:

$$\sum_{j:(i,j) \in \Lambda} Y^{(i,j)} \leq \Delta_t^i \quad \forall i \in V \quad (7.33)$$

$$\sum_{i:(i,j) \in \Lambda} Y^{(i,j)} \leq \Delta_r^j \quad \forall j \in V \quad (7.34)$$

$$Y^{(i,j)} - \sum_{p:(i,j) \in p} f_p^{(s,d)} \geq 0 \quad \forall (i,j) \in \Lambda, (s,d) \in \Omega \quad (7.35)$$

$$Y^{(i,j)} - \sum_{(s,d) \in \Omega, p:(i,j) \in p} T^{(s,d)} f_p^{(s,d)} \geq 0 \quad \forall (i,j) \in \Lambda \quad (7.36)$$

$$\sum_{p \in P^{(s,d)}} f_p^{(s,d)} + H^{(s,d)} = 1 \quad \forall (s,d) \in \Omega \quad (7.37)$$

$$Y^{(i,j)} - B_h^{(i,j)} S_h^{(i,j)} \geq 0 \quad \forall (i,j) \in \Lambda, h \in H \quad (7.38)$$

$$C_h - \sum_{(s,d) \in h, p:(i,j) \in p} f_p^{(s,d)} + S_h^{(i,j)} \geq 1 \quad \forall (i,j) \in \Lambda, h \in H \quad (7.39)$$

$$S_h^{(i,j)} - f_p^{(s,d)} \leq 0 \quad \forall (i,j) \in p, h \in H, (s,d) \in h \quad (7.40)$$

$$S_h^{(i,j)} \in B^1 \quad \forall (i,j) \in \Lambda, h \in H \quad (7.41)$$

$$f_p^{(s,d)} \in B^1 \quad \forall p \in P^{(s,d)}, (s,d) \in \Omega \quad (7.42)$$

$$H^{(s,d)} \in R_+^1 \quad \forall (s,d) \in \Omega \quad (7.43)$$

$$Y^{(i,j)} \in R_+^1 \quad \forall (i,j) \in \Lambda \quad (7.44)$$

Our computational results with the price-and-branch algorithms for MIP-PATH-LOCAL and MIP-PATH-GLOBAL in networks where we can assume infinite number of wavelengths are shown in Tables 7.7 - 7.10. We can see that, again, the number of instances solved by these procedures is higher than the number of instances solved in the networks where bifurcation of flow is not allowed (presented in the previous chapter). The summary of the average percentage gaps and the average CPU times for the two objective functions used are shown in Tables 7.11 and 7.12. We can see that these averages are lower than in the case of WDM optical networks without bifurcation of flow. Specifically, the average percentage gap for MIP-PATH-LOCAL decreased from 20.82% to 3.92% for the objective of minimizing the lost traffic in the network, and 21.15% to 13.79% for the objective of minimizing the total number of transmitters and receivers in the network. The average percentage gap for MIP-PATH-GLOBAL decreased from 5.04% to 3.89% for the objective of minimizing the total lost traffic in the network. For the second objective,



minimizing the total number of transmitters and receivers, the gap decreased from 19.41% to 10.88%. We see that, again, these results indicate superior performance of the branch-and-price procedure for the MIP-PATH-GLOBAL formulation. (Note: For this special case of the WDM optical networks, we have used the node degree inequalities to strengthen the lower bounds.)

V	C	D	B&P (LOCAL)			B&P (GLOBAL)		
			LB	UB	CPU (sec)	LB	UB	CPU (sec)
5	20	L	0.00	0.00	0.16	0.00	0.00	0.09
5	10	H	0.00	0.00	0.08	0.00	0.00	0.05
7	42	L	0.00	0.12	3605.58	0.00	0.00	236.38
7	21	H	0.70	0.70	3.58	0.19	0.70	3619.67
10	90	L	5.20	7.11	3603.13	5.20	6.69	3860.67
10	45	H	2.49	4.34	3605.33	2.55	3.96	3603.42
20	380	L	70.49	73.93	3694.36	70.49	73.76	3649.81
20	190	H	52.72	57.29	3638.88	52.77	56.56	3623.06

Table 7.7: Minimizing the lost traffic in the network. Infinite number of wavelengths. Selected instances for a complete physical network with 2 fibers (fiber capacity: 2 lightpaths) between all pairs of nodes.

V	A	C	D	B&P (LOCAL)			B&P (GLOBAL)		
				LB	UB	CPU (sec)	LB	UB	CPU (sec)
5	10	15	H	0.69	0.69	0.41	0.69	0.69	974.84
5	10	15	L	0.00	0.00	0.11	0.00	0.00	0.09
7	15	30	H	2.40	3.44	3608.22	2.10	3.10	3610.86
7	15	30	L	0.00	0.16	3607.70	0.00	0.00	4.14
10	20	40	H	2.58	3.10	3603.33	1.89	2.98	3603.84
10	20	40	L	0.00	0.16	3603.19	0.00	0.19	3601.73
20	30	60	H	4.68	6.58	3607.61	4.68	6.97	3613.19
20	30	60	L	0.00	1.37	3628.16	0.00	1.30	3615.36

Table 7.8: Minimizing the lost traffic in the network. Infinite number of wavelengths. Incomplete physical network with 2 fibers (fiber capacity: 20 lightpaths) on all arcs.

V	C	D	B&P (LOCAL)			B&P (GLOBAL)		
			LB	UB	CPU (sec)	LB	UB	CPU (sec)
5	20	L	18.27	20.00	3667.67	20.00	20.00	1120.31
5	10	H	16.00	16.00	0.41	16.00	16.00	0.22
7	42	L	35.27	40.00	3604.22	35.41	40.00	3604.31
7	21	H	30.00	32.00	3604.69	30.00	30.00	1093.16
10	90	L	75.04	88.00	3605.02	75.11	86.00	3602.48
10	45	H	62.00	70.00	3601.80	62.00	70.00	3603.00
20	380	L	322.37	396.00	3750.09	322.38	376.00	3659.58
20	190	H	252.43	306.00	3737.81	252.59	294.00	3655.01

Table 7.9: Minimizing the total number of transmitters and receivers in the network. Infinite number of wavelengths. Selected instances for a complete physical network with 2 fibers (fiber capacity: 2 lightpaths) between all pairs of nodes.

V	A	C	D	B&P (LOCAL)			B&P (GLOBAL)		
				LB	UB	CPU (sec)	LB	UB	CPU (sec)
5	10	15	H	22.00	22.00	16.91	22.00	22.00	10.16
5	10	15	L	15.30	16.00	3705.39	16.00	16.00	17.47
7	15	30	H	48.00	52.00	3603.05	48.00	54.00	3986.88
7	15	30	L	28.46	32.00	3607.47	28.87	32.00	3606.23
10	20	40	H	64.00	68.00	3601.42	64.00	68.00	3603.22
10	20	40	L	39.93	48.00	3603.47	40.15	48.00	3603.09
20	30	60	H	91.76	112.00	3666.36	91.98	106.00	3629.70
20	30	60	L	62.96	90.00	3688.89	63.12	86.00	3621.98

Table 7.10: Minimizing the total number of transmitters and receivers in the network. Infinite number of wavelengths. Incomplete physical network with 2 fibers (fiber capacity: 20 lightpaths) on all arcs.

Net. Type	B&P (LOCAL)		B&P (GLOBAL)	
	Avg. Gap	Avg. CPU (sec)	Avg. Gap	Avg. CPU (sec)
Complete	4.51 %	2268.89	4.24 %	2324.14
Incomplete	3.33 %	2707.34	3.53 %	2378.01
Overall	3.92 %	2488.12	3.89 %	2351.08

Table 7.11: Summary: Minimizing the lost traffic in the network. Infinite number of wavelengths.

Net. Type	B&P (LOCAL)		B&P (GLOBAL)	
	Avg. Gap	Avg. CPU (sec)	Avg. Gap	Avg. CPU (sec)
Complete	12.97 %	3196.46	9.17 %	2542.26
Incomplete	14.6 %	3186.62	12.58 %	2759.84
Overall	13.79 %	3191.54	10.88 %	2651.05

Table 7.12: Summary: Minimizing the total number of transmitters and receivers in the network. Node Degree Inequalities. Infinite number of wavelengths.

## 7.4 WDM Optical Networks with Traffic Bifurcation and Infinite Number of Wavelengths

The necessary modifications for the MIP-PATH-LOCAL and the MIP-PATH-GLOBAL formulations in the WDM optical networks with traffic bifurcation and infinite number of wavelengths represent a combination of modifications that we described in the previous two sections. That is, in the MIP-PATH-LOCAL formulation we need to relax integrality constraints on the flow path variables, define lightpath variables  $X_z$  as binary, and eliminate constraint on the number of lightpaths that can be established over physical links. As for our branch-and-price algorithm that corresponds to this formulation, we also need to modify the calculation of the “cost” of the new lightpaths in the same manner we did in the previous section *and* perform branching on the sum of lightpath variables.

In the case of the MIP-PATH-GLOBAL formulation, we also need to relax the integrality

constraints on the flow path variables, eliminate local lightpath variables  $X_z$ , and eliminate “packing” constraints and constraints on the number of lightpaths that can be established over physical links. Note that both the upper bounds and lower bounds generated by MIP-PATH-GLOBAL are now valid.

### MIP-PATH-LOCAL<sup>(A,I)</sup> Formulation

$$\text{Min} \quad \sum_{(s,d) \in \Omega} T^{(s,d)} H^{(s,d)} \quad (7.45)$$

Subject to:

$$\sum_{z:O(z)=i} X_z \leq \Delta_t^i \quad \forall i \in V \quad (7.46)$$

$$\sum_{z:D(z)=j} X_z \leq \Delta_r^j \quad \forall j \in V \quad (7.47)$$

$$X_z - \sum_{p:z \in p} f_p^{(s,d)} \geq 0 \quad \forall z \in Z, (s,d) \in \Omega \quad (7.48)$$

$$X_z - \sum_{(s,d) \in \Omega, p:z \in p} T^{(s,d)} f_p^{(s,d)} \geq 0 \quad \forall z \in Z \quad (7.49)$$

$$\sum_{p \in P^{(s,d)}} f_p^{(s,d)} + H^{(s,d)} = 1 \quad \forall (s,d) \in \Omega \quad (7.50)$$

$$f_p^{(s,d)} \in R_+^1 \quad \forall p \in P^{(s,d)}, (s,d) \in \Omega \quad (7.51)$$

$$H^{(s,d)} \in R_+^1 \quad \forall (s,d) \in \Omega \quad (7.52)$$

$$X_z \in B^1 \quad \forall z \in Z \quad (7.53)$$

### MIP-PATH-GLOBAL<sup>(A,I)</sup> Formulation

$$\text{Min} \quad \sum_{(s,d) \in \Omega} T^{(s,d)} H^{(s,d)} \quad (7.54)$$

Subject to:

$$\sum_{j:(i,j) \in \Lambda} Y^{(i,j)} \leq \Delta_t^i \quad \forall i \in V \quad (7.55)$$

$$\sum_{i:(i,j) \in \Lambda} Y^{(i,j)} \leq \Delta_r^j \quad \forall j \in V \quad (7.56)$$

$$Y^{(i,j)} - \sum_{p:(i,j) \in p} f_p^{(s,d)} \geq 0 \quad \forall (i,j) \in \Lambda, (s,d) \in \Omega \quad (7.57)$$

$$Y^{(i,j)} - \sum_{(s,d) \in \Omega, p:(i,j) \in p} T^{(s,d)} f_p^{(s,d)} \geq 0 \quad \forall (i,j) \in \Lambda \quad (7.58)$$

$$\sum_{p \in P^{(s,d)}} f_p^{(s,d)} + H^{(s,d)} = 1 \quad \forall (s,d) \in \Omega \quad (7.59)$$

$$f_p^{(s,d)} \in R_+^1 \quad \forall p \in P^{(s,d)}, (s,d) \in \Omega \quad (7.60)$$

$$H^{(s,d)} \in R_+^1 \quad \forall (s,d) \in \Omega \quad (7.61)$$

$$Y^{(i,j)} \in Z_+^1 \quad \forall (i,j) \in \Lambda \quad (7.62)$$

Our computational results with the price-and-branch algorithms for MIP-PATH-LOCAL and MIP-PATH-GLOBAL in the networks that allow bifurcation of flow and have ample fiber capacity are shown in Tables 7.13 - 7.16. We can see that the number of instances solved by these procedures is higher than the number of instances solved in the networks where bifurcation of flow is not allowed (presented in the previous chapter). The summary of the average percentage gaps and the average CPU times for the two objective functions used are shown in Tables 7.17 and 7.18. We can see that these averages are lower than in the case of WDM optical networks without bifurcation of flow. Specifically, the average percentage gap for MIP-PATH-LOCAL decreased from 20.82% to 0.74% for the objective of minimizing the lost traffic in the network, and 21.15% to 20.09% for the objective of minimizing the total number of transmitters and receivers in the network. The average percentage gap for MIP-PATH-GLOBAL decreased from 5.04% to 0.88% for the objective of minimizing the total lost traffic in the network. For the second objective, minimizing the total number of transmitters and receivers, the percentage gap decreased from 19.41% to 10.67%. As before, these results indicate superior performance of the branch-and-price procedure for the MIP-PATH-GLOBAL formulation.

V	C	D	B&P (LOCAL)			B&P (GLOBAL)		
			LB	UB	CPU (sec)	LB	UB	CPU (sec)
5	20	L	0.00	0.00	0.23	0.00	0.00	0.09
5	10	H	0.00	0.00	0.08	0.00	0.00	0.06
7	42	L	0.00	0.00	2.20	0.00	0.00	5.91
7	21	H	0.19	0.19	1.16	0.19	0.19	12.50
10	90	L	5.20	5.20	799.55	5.20	5.20	901.67
10	45	H	2.53	2.79	3604.30	2.54	2.79	3602.81
20	380	L	70.49	70.49	2137.66	70.49	70.49	3649.83
20	190	H	52.78	53.44	3625.45	52.77	53.46	3617.34

Table 7.13: Minimizing the lost traffic in the network. Bifurcation of flow allowed and infinite number of wavelengths. Selected instances for a complete physical network with 2 fibers (fiber capacity: 2 lightpaths) between all pairs of nodes.

V	A	C	D	B&P (LOCAL)			B&P (GLOBAL)		
				LB	UB	CPU (sec)	LB	UB	CPU (sec)
5	10	15	H	0.38	0.38	0.20	0.38	0.38	0.11
5	10	15	L	0.00	0.00	0.11	0.00	0.00	0.08
7	15	30	H	2.04	2.04	1208.16	2.04	2.04	292.34
7	15	30	L	0.00	0.00	1.08	0.00	0.00	0.58
10	20	40	H	1.84	2.12	3602.58	1.86	2.12	3603.41
10	20	40	L	0.00	0.16	3602.14	0.00	0.00	227.92
20	30	60	H	4.68	5.37	3605.19	4.68	6.02	3611.86
20	30	60	L	0.00	0.81	3620.89	0.00	1.04	3614.13

Table 7.14: Minimizing the lost traffic in the network. Bifurcation of flow allowed and infinite number of wavelengths. Incomplete physical network with 2 fibers (fiber capacity: 20 lightpaths) on all arcs.

V	C	D	B&P (LOCAL)			B&P (GLOBAL)		
			LB	UB	CPU (sec)	LB	UB	CPU (sec)
5	20	L	18.11	20.00	3611.52	18.16	20.00	3604.01
5	10	H	16.00	16.00	3.53	16.00	16.00	1.41
7	42	L	34.80	38.00	3601.53	35.25	38.00	3603.66
7	21	H	28.52	30.00	3611.99	28.50	30.00	3621.89
10	90	L	74.87	80.00	3608.13	75.01	80.00	3606.59
10	45	H	56.77	62.00	3606.55	57.12	60.00	3603.20
20	380	L	322.34	760.00	3600.00	322.34	330.00	6377.45
20	190	H	247.19	380.00	3668.30	247.05	380.00	3804.00

Table 7.15: Minimizing the total number of transmitters and receivers in the network. Bifurcation of flow allowed and infinite number of wavelengths. Selected instances for a complete physical network with 2 fibers (fiber capacity: 2 lightpaths) between all pairs of nodes.

V	A	C	D	B&P (LOCAL)			B&P (GLOBAL)		
				LB	UB	CPU (sec)	LB	UB	CPU (sec)
5	10	15	H	22.00	22.00	159.20	22.00	22.00	45.63
5	10	15	L	16.00	16.00	158.81	16.00	16.00	74.13
7	15	30	H	42.28	44.00	3607.42	42.51	44.00	3615.05
7	15	30	L	27.97	30.00	3602.28	28.50	30.00	3608.11
10	20	40	H	56.92	62.00	3604.36	57.23	64.00	3602.13
10	20	40	L	39.15	46.00	3606.78	39.60	46.00	3603.30
20	30	60	H	87.34	106.00	3860.09	87.34	102.00	3627.73
20	30	60	L	62.19	82.00	3765.66	62.19	82.00	3753.22

Table 7.16: Minimizing the total number of transmitters and receivers in the network. Bifurcation of flow allowed and infinite number of wavelengths. Incomplete physical network with 2 fibers (fiber capacity: 20 lightpaths) on all arcs.

Net. Type	B&P (LOCAL)		B&P (GLOBAL)	
	Avg. Gap	Avg. CPU (sec)	Avg. Gap	Avg. CPU (sec)
Complete	0.31 %	1271.33	0.32 %	1473.78
Incomplete	1.16 %	1955.04	1.43 %	1418.8
Overall	0.74 %	1613.19	0.88 %	1446.29

Table 7.17: Summary: Minimizing the lost traffic in the network. Bifurcation of flow allowed and infinite number of wavelengths.

Net. Type	B&P (LOCAL)		B&P (GLOBAL)	
	Avg. Gap	Avg. CPU (sec)	Avg. Gap	Avg. CPU (sec)
Complete	28.80 %	3163.94	11.39 %	3527.78
Incomplete	11.37 %	2795.58	10.67 %	2741.16
Overall	20.09 %	2979.76	11.03 %	3134.47

Table 7.18: Summary: Minimizing the total number of transmitters and receivers in the network. Bifurcation of flow allowed and infinite number of wavelengths.

The summary of our computational results for different types of network restrictions and the two objective functions used in our computational experiments are shown in Tables 7.19 and 7.20. We can see that the gaps significantly decrease in the less restricted WDM optical network settings. The only exception seem to be the networks where bifurcation of flow is allowed and an infinite number of wavelengths is available when the objective is minimizing the total number of transmitters and receivers.

Network Setting	B&P (LOCAL)		B&P (GLOBAL)	
	Avg. Gap	Avg CPU	Avg. Gap	Avg CPU
BNA	20.82%	3386.24	5.04%	2478.2
BA	3.48%	2033.29	0.77%	1472.31
IW	3.92%	2488.12	3.89%	2351.08
BA & IW	0.74%	1613.19	0.88%	1446.29

Table 7.19: Overall Results. Minimizing the total lost traffic in the network.

Network Setting	B&P (LOCAL)		B&P (GLOBAL)	
	Avg. Gap	Avg CPU	Avg. Gap	Avg CPU
BNA	21.15%	3160.84	19.41%	2643.77
BA	19.59%	3623.87	19.39%	2971.01
IW	13.79%	3191.54	10.88%	2651.05
BA & IW	20.09%	2979.76	11.03%	3134.47

Table 7.20: Overall Results. Minimizing the total number of transmitters and receivers.

## 7.5 Conclusion

Our computational experiments for the WDM optical networks with three different types of network restrictions (bifurcation of flow allowed, infinite number of wavelengths, and the com-

combination of the two) indicate that the performance of our algorithms significantly improves when used in the less restrictive networks. These improvements are particularly high for the objective of minimizing the total lost traffic in the network where the average percentage gaps decreased to less than 5%. The gaps for the second objective function also improve as we reduce restrictions in the network. However, this improvement is not as significant as in the case of the first objective function. We suspect that better upper bounds could provide better results for this objective function.

## Chapter 8

### Summary and Conclusions

In this dissertation we presented solution procedures for two NP-hard network design problems that arise in telecommunications: the prize-collecting generalized minimum spanning tree (PCGMST) problem, and the WDM optical network design problems.

In the PCGMST problem, a set of local area networks needs to be connected by a minimum cost tree structure, and for that purpose exactly one gateway site needs to be selected out of a set of candidate sites from each region. The cost of this tree structure is defined as the difference between the cost of the transmission facilities laid out between gateways of different local area networks and the compensation or “prize” offered by the selected gateways.

For this problem, we first provided a comprehensive review of the mathematical formulations developed for this problem so far. Then, we discussed some important properties of one of the formulations for the PCGMST problem and explained how these properties can be used to develop simple exact procedures for this problem.

The solution procedures proposed in this dissertation include several heuristic and metaheuristic algorithms, and two exact solution procedures for the PCGMST problem. Our heuristic procedures include a simple lower bound for the PCGMST problems and three upper bound adaptations of the well-known minimum spanning tree algorithms (Kruskal’s, Prim’s, and Sollin’s). We also discussed a polynomial search paradigm called the Pilot Method that significantly improves these adaptations. We then presented two metaheuristic procedures - local search and genetic algorithm.

Our computational experiments showed that, as expected, the straightforward adaptations of the MST heuristics do not provide good bounds for the PCGMST problem. However, when the Pilot Method is applied to these heuristics they provide substantially better results. The results of our computational experiments for the two metaheuristic algorithms presented in this chapter



were particularly compelling. They indicate that both LS and GA can be used to rapidly obtain high quality solutions for the PCGMST problem in relatively large networks.

The two exact procedures for the prize-collecting generalized minimum spanning tree problem proposed in this dissertation include a new implementation of the Rooting Procedure originally proposed by Pop [46], and a novel branch-and-cut algorithm for the PCGMST problem, that works with integer incumbent nodes of the branch-and-bound tree. Our computational experiments suggest that these two procedures are a good choice for networks of small and moderate size. For larger networks, the metaheuristic procedures, such as genetic algorithm or local search are a better alternative.

The second problem that we study in this dissertation is the design of optical networks with wavelength division multiplexing (WDM). This problem has received a lot of research attention in the last 10 years due to enormous bandwidth provided by optical fibers and the expensive optical and/or electronic equipment needed to process traffic requests in these networks. But, due to notoriously hard nature of this problem, most researchers have developed only heuristic procedures. The few studies that presented exact solution procedures for the WDM optical network design problem mostly work with the relaxed versions of this problem. To the best of our knowledge, the branch-and-price algorithms proposed in this dissertation represent the first attempt to develop an exact solution procedure for a more general WDM optical network design setting that combines lightpath routing and traffic grooming.

In this dissertation, we have mainly focused on WDM optical networks with wavelength conversion and without bifurcation of traffic. In this specific optical network setting, we are given a physical network with a set of restrictions on its nodes and links. The goal is to simultaneously design a logical topology consisting of lightpaths routed over the physical network, and determine the routing of the traffic requests over the logical topology, so that a pre-specified measure of network performance is optimized.

We have presented two path-based formulations for the WDM optical network design problem, and then discussed the complexity behind the design of branch-and-price algorithms for this

problem. We have then developed two branch-and-price procedures for the proposed formulations MIP-PATH-LOCAL and MIP-PATH-GLOBAL. We also showed that, with a few modifications, the branch-and-price algorithm for MIP-PATH-LOCAL can be used to obtain either optimal solutions or approximate upper bound solutions.

For the second proposed formulation, MIP-PATH-GLOBAL, we proposed an approximate lower-bound branch-and-price algorithm that provided the best quality solutions out of the three branch-and-price procedures proposed in this dissertation.

In our computational experiments, both approximate branch-and-price algorithms for the MIP-PATH-LOCAL formulation and the MIP-PATH-GLOBAL formulation found optimal solution for all instances where the optimal solution is known. Given the fact that our approximate branch-and-price algorithms for the MIP-PATH-LOCAL and MIP-PATH-GLOBAL formulations provide valid upper and lower bounds for the WDM optical network design problem, we suggest the combined use of these procedures to find provably optimal solutions.

We also proposed and computationally tested two classes of valid inequalities (lifted cover inequalities and node inequalities) that can be used in combination with our branch-and-price algorithms. Our computational results indicate that the addition of the first class of valid inequalities, the lifted cover inequalities, does not improve lower bounds. The second class of valid inequalities tested in this dissertation represents node degree inequalities that can be used in situations where all the demand must be served in a given optical network. Our computational tests indicate that, unlike lifted cover inequalities, these inequalities do improve the lower bounds provided by our branch-and-price procedures. However, the use of these inequalities also has a tendency to make the problem harder in terms of finding good upper bounds. We suspect that in combination with fast custom upper bound heuristics for the WDM optical network design, the use of node degree inequalities can lead to higher quality of solutions for this problem.

In the last part of this dissertation we have performed computational tests with our approximate branch-and-price algorithm in the WDM optical network with different network restrictions. Specifically, we have looked at the WDM optical networks where bifurcation of flow is allowed,

networks with ample fiber capacity and networks that both have ample fiber capacity and allow bifurcation of flow. Our computational experiments indicate that the performance of our approximate branch-and-price algorithms improves as we relax restrictions in the network settings. Additionally, we find that the branch-and-price algorithm for the MIP-PATH-GLOBAL formulation has a superior performance to the branch-and-price algorithm for the MIP-PATH-LOCAL formulation. This is an interesting and important result, since our branch-and-price algorithm for the MIP-PATH-GLOBAL formulation is an exact solution procedure for WDM optical networks where bifurcation of flow is allowed.

Based on our comprehensive study of the WDM optical network design problem, we believe that the best way to further improve optimality gaps is to develop efficient upper bounding heuristic procedures. The lower bounds provided by our branch-and-price algorithms (especially the ones based on the MIP-PATH-LOCAL formulation) are probably not likely to benefit from the dynamic generation of valid inequalities. The reason is that the row-incomplete structure of the master problem used in our exact and upper bound branch-and-price algorithms makes the development and the dynamic use of valid inequalities very difficult and often impractical.

Another possible venue for future research is to try to improve the search strategy in the column generation part of our branch-and-price algorithms. For example, one could consider heuristic approaches instead of (or combined with) exact pricing used to solve the subproblem. This strategy was proposed by Chabrier [10] for the branch-and-price algorithm developed for the multicommodity multi-facility network design problem. In the case of WDM optical network design problem this strategy could be applied in the following way. We could use heuristics to define a set of **good** candidate lightpaths that would define an initial **pool of variables**. At a given node of the branch-and-bound tree we could first look at the reduced cost of the lightpaths in this pool, and only when this pool is exhausted we could perform exact pricing. If any good candidate lightpaths are identified by exact pricing, we could add these to the initial pool of variables. The updated pool of variables could be then used at other nodes of the branch-and-bound tree.

We leave these directions for the future research.

## BIBLIOGRAPHY

- [1] *ILOG Branch & Price & Cut Shortest Path Optimizers Prototype Manual*. ILOG, France, 2003.
- [2] E. H. L. Aarts and J. K. Lenstra. *Local Search in Combinatorial Optimization*. John Wiley and Sons, UK, 1997.
- [3] M. O. Ball and A. Vakhutinsky. Fault tolerant virtual path layout: Optimization models. In Sanso B and P. Soriano, editors, *Telecommunications Network Planning*, pages 210–217. Kluwer Academic, Boston, MA, 1998.
- [4] M. O. Ball and A. Vakhutinsky. Fault-tolerant virtual path layout in ATM networks. *INFORMS Journal on Computing*, 13(1):76–94, 2001.
- [5] D. Banerjee and B. Mukherjee. A practical approach for routing and wavelength assignment in large wavelength-routed optical networks. *IEEE JSAC*, 14(5):903–908, 1996.
- [6] D. Banerjee and B. Mukherjee. Wavelength-routed optical networks: Linear formulation, resource budgeting tradeoffs, and a reconfiguration study. *IEEE/ACM Transactions on Networking*, 8(5):598–607, 2000.
- [7] C. Barnhart, C. A. Hane, and P. H. Vance. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research*, 48(2):318–326, 2000.
- [8] P. Belotti and F. Malucelli. Multilayer network design: a row-column generation algorithm. *Proceedings of International Network Optimization Conference*, pages B2.422–B2.427, 2005.
- [9] D. Bienstock and O. Gunluk. Computational experience with a difficult mixed-integer multi-commodity flow problem. *Mathematical Programming*, 68:213–237, 1995.
- [10] A. Chabrier. Heuristic branch-and-price-and-cut to solve a network design problem. *Proceedings of CPAIOR*, 2003.

- [11] H. S. Chang, B. W. Kim, C. G. Lee, S. L. Min, Y. Choi, H. S. Yang, D. N. Kim, and C. S. Kim. FSA-based link assignment and routing in low-earth orbit satellite networks. *IEEE Transactions on Vehicular Technology*, 47(3):1037–1048, August 1998.
- [12] I. Chlamtac, A. Ganz, and G. Karmi. Lightpath communications: An approach to high bandwidth optical WAN's. *IEEE Transactions on Communications*, 40(7):1171–1182, 1992.
- [13] I. Chlamtac, A. Ganz, and G. Karmi. Lightnets: Topologies for high speed optical networks. *Journal of Lightwave Technology*, 11(5–6):951–961, 1993.
- [14] E. J. Cockayne and Z. A. Melzak. Steiner's problem for set terminals. *Quarterly Applied Mathematics*, 26:213–218, 1968.
- [15] G. Dahl, A. Martin, and M. Stoer. Routing through virtual paths in layered telecommunication networks. *Operations Research*, 47(5):693–702, 1999.
- [16] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.
- [17] M. Dror, M. Haouari, and J. Chaouachi. Generalized spanning trees. *European Journal of Operations Research*, 120:583–592, 2000.
- [18] C. Duin and S. Voß. The pilot method: A strategy for heuristic repetition with application to the Steiner problem in graphs. *Networks*, 34(3):181–191, 1999.
- [19] C. Duin and S. Voß. Solving group Steiner problems as Steiner problems. *European Journal of Operational Research*, 154:323–329, 2004.
- [20] C.W. Duin. *Steiner Problem in Graphs: Approximation, Reduction, Variation*. PhD thesis, University of Amsterdam, 1993.
- [21] R. Dutta and G. N. Rouskas. A survey of virtual topology design algorithms for wavelength routed optical networks. *Optical Networks Magazine*, January 2000.

- [22] A. Farahat, C. Barnhart, H. Liu, R. Schneur, and R. Tobin. Traffic grooming in optical networks. *Working Paper*, 28 September, 2004.
- [23] P. Fard, R. La, K. Lee, I. Marcus, and M. Shayman. Reconfiguration of MPLS/WDM networks using simulation-based Markov decision processes. *Proceedings of Conference on Information Sciences and Systems*, 2005.
- [24] C. Feremans. *Generalized Spanning Trees and Extensions*. PhD thesis, Université Libre de Bruxelles, 2001.
- [25] C. Feremans, M. Labbé, and G. Laporte. A comparative analysis of several formulations for the generalized minimum spanning tree problem. *Networks*, 39:29–34, 2002.
- [26] C. Feremans, M. Labbé, and G. Laporte. The generalized minimum spanning tree problem: Polyhedral analysis and branch-and-cut algorithm. *Networks*, 43(2):71–86, 2004.
- [27] M. Fischetti, J. J. Salazar-Gonzalez, and P. Toth. Symmetric generalized traveling salesman problem. *Operations Research*, 45(3):378–394, 1997.
- [28] I. Gamvros, B. Golden, S. Raghavan, and D. Stanojević. Heuristic search for network design. In H. Greenberg, editor, *Tutorials on Emerging Methodologies and Applications in Operations Research*, pages 1–1 to 1–46. Springer, 2004.
- [29] A. Gençata and B. Mukherjee. Virtual-topology adaptation for WDM mesh networks under dynamic traffic. *IEEE/ACM Transactions on Networking*, 11(2):236–247, April 2003.
- [30] B. Golden, S. Raghavan, and D. Stanojević. Heuristic search for the generalized minimum spanning tree problem. *INFORMS Journal on Computing*, 17(3):290–304, 2005.
- [31] Z. Gu, G. L. Nemhauser, and M. Savelsbergh. Lifted cover inequalities for 0-1 integer programs: Computation. *INFORMS Journal on Computing*, 10(4):427–437, 1998.
- [32] M. Haouari, J. Chaouachi, and M. Dror. Solving the generalized minimum spanning tree problem by a branch-and-bound algorithm. *Journal of the Operational Research Society*, 56(4):382–389, April 2005.

- [33] A. Haque, Y. P. Aneja, S. Bandyopadhyay, A. Jaekel, and A. Sengupta. Some studies on the logical topology design of large multi-hop optical networks. *Optical Networks Magazine*, July/August 2002.
- [34] H. Höller and S. Voß. An integer programming model for integrated SDH/WDM network planning. *Proceedings of International Network Optimization Conference*, pages B3.726–B3.733, 2005.
- [35] J. Q. Hu and B. Leida. Traffic grooming, routing, and wavelength assignment in optical WDM mesh networks. *manuscript available at <http://people.bu.edu/hqiang/papers/grw.pdf>*, 2002.
- [36] V. R. Konda and T.Y. Chow. Algorithm for traffic grooming in optical networks to minimize the number of transceivers. *Proc. IEEE 2001 Workshop on High performance switching and Routing*, pages 218–221, May 2001.
- [37] R. M. Krishnaswamy and K. N. Sivarajan. Design of logical topologies: A linear formulation for wavelength-routed optical networks with no wavelength changers. *IEEE/ACM Transactions on Networking*, 9(2):186–198, 2001.
- [38] K. Lee, L. Sudarsan, and M. Shayman. Integrated logical topology design and traffic grooming in re-configurable WDM networks. *Working Paper*, 2002.
- [39] K. Lee, L. Sudarsan, and M. Shayman. A local optimization algorithm for logical topology design and traffic grooming in IP over WDM networks. Technical Report TR 2003-3, Institute for Systems Research, 2003.
- [40] T.L. Magnanti and L. A. Wolsey. Optimal trees. *Network Models*, 7:503–615, 1995.
- [41] R.K. Martin. Using separation algorithms to generate mixed integer model reformulations. *Operations Research Letters*, 10(3):119–128, 1991.
- [42] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, Heidelberg, 1996.

- [43] E. Modiano and P. J. Lin. Traffic grooming in WDM networks. *IEEE Communications Magazine*, pages 124–129, July 2001.
- [44] B. Mukherjee, D. Banerjee, S. Ramamurthy, and A. Mukherjee. Some principles for designing a wide-area WDM optical network. *IEEE/ACM Transactions on Networking*, 4(5):684–696, 1996.
- [45] Y. S. Myung, C. H. Lee, and D. W. Tcha. On the generalized minimum spanning tree problem. *Networks*, 26:231–241, 1995.
- [46] P. Pop. *The Generalized Minimum Spanning Tree Problem*. PhD thesis, University of Twente, 2002.
- [47] P. C. Pop, W. Kern, and G. Still. A new relaxation method for the generalized minimum spanning tree problem. *European Journal of Operational Research*, to appear - available online at [www.sciencedirect.com](http://www.sciencedirect.com).
- [48] P. C. Pop, W. Kern, and G. J. Still. The generalized minimum spanning tree problem. Technical report, Department of Operations Research and Mathematical Programming, University of Twente, 2000. available at <http://www.math.utwente.nl/dos/ormp/preprints.htm>.
- [49] P. C. Pop, W. Kern, and G. J. Still. An approximation algorithm for the generalized minimum spanning tree problem with bounded cluster size. Technical report, Department of Operations Research and Mathematical Programming, University of Twente, 2001. available at <http://www.math.utwente.nl/dos/ormp/preprints.htm>.
- [50] P. Prathombutr, J. Stach, and E.K. Park. An algorithm for traffic grooming in WDM optical mesh networks with multiple objectives. *Telecommunications Systems*, 28(3–4):369–386, 2005.
- [51] S. Raghavan. On modeling the generalized minimum spanning tree. Technical report, University of Maryland, 2002.
- [52] R. Ramaswami and G. Sasaki. Multiwavelength optical networks with limited wavelength conversion. *IEEE/ACM Transactions on Networking*, 6(6):744–754, December 1998.



- [53] R. Ramaswami and K. Sivarajan. *Optical Networks: A Practical Perspective*. Morgan Kaufmann, San Francisco, second edition, 2002.
- [54] R. Ramaswami and K. N. Sivarajan. Design of logical topologies for wavelength-routed optical networks. *IEEE Journal on Selected Areas in Telecommunications*, 14(5):840–851, 1996.
- [55] G. Reinelt. TSPLIB—A traveling salesman problem library. *INFORMS Journal on Computing*, 3:376–384, 1991.
- [56] S. J. Shyu, P. Y. Yin, B. M. T. Lin, and M. Haouari. Ant-tree: an ant colony optimization approach to the generalized minimum spanning tree problem. *Journal of Experimental and Theoretical Artificial Intelligence*, 15:103–112, 2003.
- [57] M. Stoer and F. Wagner. A simple min cut algorithm. In J. van Leeuwen, editor, *Algorithms - ESA '94*, volume 855 of *Lecture Notes in Computer Science*, pages 141–147. Springer, 1994.
- [58] CS Sung and SH Song. Branch-and-price algorithm for a combined problem of virtual path establishment and traffic packet routing in a layered communication network. *Journal of the Operational Research Society*, (54):72–82, 2003.
- [59] Craig A. Tovey. Local improvement on discrete structures. In E. H. L. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 57–90. John Wiley & Sons, 1997.
- [60] F. Vanderbeck. Implementing mixed integer column generation. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*. Kluwer, 2005. Working paper version available at <http://www.math.u-bordeaux.fr/fv/papers/impPap.pdf>.
- [61] F. Vanderbeck and L. A. Wolsey. An exact algorithm for IP column generation. *Operations Research Letters*, 19:151–159, 1996.
- [62] K. Zhu and B. Mukherjee. Traffic grooming in an optical WDM mesh network. *IEEE Journal on Selected Areas in Communications*, 20(1):122–133, 2002.