UNIVERSITY OF ALBERTA

HEURISTIC METHODS FOR SUB-GRAPH TOPOLOGY ENHANCEMENT IN RING-BASED TRANSPORT NETWORK DESIGN

By

CHEE YOON LEE ©

A Thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Science

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

EDMONTON, ALBERTA

SPRING 2000

0-612-60138-2

University of Alberta

Library Release Form

Name of Author:	Lee, Chee Yoon
Title of Thesis:	Heuristic Methods for Sub-Graph Topology Enhancement in Ring-Based Transport Network Design
Degree:	Master of Science
Year this Degree is Granted:	Spring 2000

[illegible]

Lee, Chee Yoon
6, Aitken Road
Canning Garden
31400 Ipoh, Perak
MALAYSIA

Date: 8/Dec/1999

UNIVERSITY OF ALBERTA

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Heuristic Methods for Sub-Graph Topology Enhancement in Ring-Based Transport Network Design** submitted by **Chee Yoon Lee** in partial fulfillment of the requirements for the degree of **Master of Science**.

[illegible]

Dr. Wayne D. Grover, Supervisor

[illegible]

Dr. Bruce F. Cockburn, Internal

[illegible]

Dr. Janelle J. Harms, External

Date: Nov 29, 1999

Abstract

Designing a Dense Wavelength-Division Multiplexing (DWDM) optical network based on ring protection schemes is logically similar to designing SONET ring-based networks. An aspect that is common to much of the work on the optimized design of multiple-ring SONET (or WDM) networks is that they treat the design as a form of graph-covering problem. Although this provides good fully-restorable designs, there may be unnecessary lower bounds on the network cost when the ring-sets found are restricted to protect every span in the fiber graph.

This thesis explores the performance of and feasibility of using "Span Elimination" to advance the art of ring network design beyond the principle of pure coverage design. Span elimination is the problem of finding those key spans of an existing fiber graph on which it is more effective not to route any demands, thereby avoiding the requirement of ring coverage on the span. Four systematic span elimination strategies and their variants have been developed to realize the effects of span eliminations. They are Modular Aggregating Pre-Routing (MAPR), Iterated Routing and Elimination (IRE), Iterative MAPR (iMAPR) and Post-Inspection and Re-routing (PIR). The experimental tests on these heuristics are performed on four trial networks.

The span elimination heuristics are compared against the pure coverage design with no span eliminations. Experimental results show that the span elimination heuristics offer significant cost reduction from the designs that emerge from a coverage-based solver, arising from judicious span elimination. Each of the proposed span elimination heuristics has its own merits and performance in terms of solution quality and computational time.

Dedicated to my loving parents
Chai Chong and Yoke Ying

Acknowledgments

I would first like to express my thanks to my supervisor, Dr. Wayne D. Grover, for all the patience, support and guidance that he has shown over the past two years. I thank him for giving me the opportunity to work in the field of optical network planning, and for providing me with financial support throughout the graduate program. I would like to thank TR*Labs* for their financial support, and for the superb working environment (especially the great view of the southside of Edmonton). I would also like to thank G. Dave Morley for his patience, technical and computing support, and especially for his constructive and innovative suggestions in this research problem.

I thank Ernest Siu, John Doucette, Sik Heng Foo, Robert Novak, Sing Cheng, Kay Wee Ang, SC and AOE for the technical discussions and the countless number of memorable nights working late together in the lab. I thank Horace Yu for his spiritual support, Luke Chong for his patience and computing expertise, Clarence Kan, Xiangqian Xu, Matthieu Clouqueur, Demetrios Stamatelakis, Dr. Ray DeCorby, Dr. Witek Krzymien and Marcia Jeremiah for providing me with great support. I thank the staff of TR*Labs*, Rhoda Hayes, Linda Richens, Bernadette Holowach, Dr. Geoffrey Hayward, Dr. Michael Leung and Dr. Roger Pederson for their continuous support and the favorable environment to carry out my research. I thank my examiners, Dr. Bruce Cockburn and Dr. Janelle Harms for the questions and suggestions on improving this thesis. I would also like to thank the University of Alberta, in particular, the Faculty of Graduate Studies and Research, and the Department of Electrical and Computer Engineering.

Last but not least, I would like to thank my loving parents, brothers and sister, and especially my girlfriend, Wei Ling Tan, for their support and understanding throughout my graduate studies.

Table of Contents

1.0 Introduction 1

1.1 Concept of Span Eliminations 3

1.2 Research Objectives 6

1.3 Outline 6

2.0 Span Elimination Concepts and Prior Work 9

2.1 Introduction to Span Elimination 9

2.2 Prior Work in Span Elimination 12

2.3 Research Methods 13

2.3.1 Network Topology Considerations 13

2.3.2 Network Demand Routing 16

2.3.3 Cycle Generator 16

2.4 Experimental Network Models 16

3.0 Span Coverage Integer Program (SCIP) 21

3.1 Concepts 21

3.2 SCIP Formulation 23

3.2.1 Notations 23

3.2.2 IP Formulation 23

3.2.3 Complexity Analysis of SCIP 24

3.2.4 Network and Equipment Costs 26

3.2.5 Limiting the Maximum Cycle Size 27

3.3 Experimental Results: Pure-Coverage Designs with SCIP 28

3.3.1 Single-Modularity Ring Coverage Design 28

3.3.2 Multi-Modularity Ring Coverage Design 29

3.3.3 Computational Time for SCIP Results 29

3.4 Discussions and Summary 31

4.0 Modular Aggregating Pre-Routing (MAPR) 33

4.1 Introduction 33

4.2 Detailed Explanation of MAPR 35

4.2.1 Flow Chart 36

4.2.2 Modular Pseudo-cost function in MAPR (Single-modular case) 38

4.3 Summary 40

5.0 Iterated Routing and Elimination (IRE): Single-Modularity 41

5.1 Introduction 41

5.2 Implementing IRE 45

5.3 Summary 46

6.0 Initial Experimental Results and Discussions 48

6.1 Results and Analysis of MAPR and IRE 48

6.1.1 Non-Essential Spans (Candidates for Elimination) 48

6.1.2 MAPR 53

6.1.3 IRE 59

6.2 Interim Conclusions 64

6.2.1 Comparison of the Performance of MAPR and IRE 64

6.2.2 Discussions and Conclusions 67

7.0 Ideas for Modification and Improvements for MAPR 69

7.1 Iterative MAPR (iMAPR) 69

7.1.1 Introduction 69

7.1.2 iMAPR Pseudo-Cost Function 71

7.1.3 Summary of Complete iMAPR Procedure 80

7.1.4 Discussions and Summary 81

8.0 Post-Inspection and Re-Routing (PIR) 82

8.1 Introduction 82

8.2 Complete PIR Algorithm 90

8.3 Summary 94

9.0 Performance and Results 95

9.1 Results of Pure Shortest-Path Routing using Multi-Modular Capacity SCIP - Revised 95

9.2 Results and Analysis of iMAPR 96

9.3 Results of IRE using Multi-Modular SCIP 101

9.4 Results using PIR 105

9.5 Comparison 106

9.6 Concluding Discussion of Results 109

10.0 Correlation between SCIP and RingBuilder 112

10.1 The Issue 112

10.2 Results and Discussions 114

11.0 Summary of Thesis 116

11.1 Suggestions for Further Research 122

11.1.1 Test Case: The impact of using a different non-essential spans set on SCIP design cost (Using IRE heuristic on Net B) 122

11.1.2 Future Improvements for iMAPR and PIR heuristics 126

Bibliography 129

Appendix A. Test Network Standard Network Interface Files (SNIF) 132

Appendix B. Span Coverage Integer Program (SCIP): Implementation Validation 144

Appendix C. AMPL Model File for SCIP 147

Appendix D. MAPR using *fall* pseudo-cost 164

List of Tables

Table 2.1. Characteristics of the test networks. 16

Table 3.1. Network Equipment and Fiber cost 26

Table 3.2. The maximum cycle size defined for each test network. 27

Table 3.3. Min-cost results obtained using shortest-path routing and SCIP single-modular design. 28

Table 3.4. Min-cost results for multi-modular SCIP (CPU runtime limited to 24 hours). 29

Table 3.5. Computation times for SCIP. 30

Table 6.1. Test networks' non-essential spans (for IRE investigations). 48

Table 6.2. MAPR results for Net A. 54

Table 6.3. MAPR results for Net B. 54

Table 6.4. MAPR results for Net C. 55

Table 6.5. MAPR results for Net D. 55

Table 6.6. Least cost SCIP design obtained using MAPR(rise) and MAPR(impulse) 58

Table 6.7. The selection of design modularity, the number of non-essential spans and the number of SCIP runs for each test network in IRE investigation. 60

Table 6.8. The minimum SCIP results using IRE single-modular capacity design. 64

Table 6.9. The minimum SCIP results and the total computational time using pure shortest-path (no eliminations), MAPR (rise) and IRE. 65

Table 7.1. Aggregation pressure coefficient, $\alpha_\Delta(r_1)$ *and* $\alpha_\Delta(r_2)$, for corresponding Δ, r_1 and r_2 values determined through trial and error. 78

Table 9.1. Min-cost results obtained for network using shortest-path routing and SCIP multi-modular capacity design with CPU runtime limited to 900 seconds compared to CPU runtime limited to 24-hour. 95

Table 9.2. iMAPR results for Net A. 97

Table 9.3. iMAPR results for Net B. 98

Table 9.4. iMAPR results for Net C. 99

Table 9.5. iMAPR results for Net D. 100

Table 9.6. The most-improved SCIP results for each test network using iMAPR. 101

Table 9.7. The most-improved SCIP design results using IRE multi-modular design............ 104

Table 9.8. The most-improved SCIP design obtained using PIR.. 105

Table 9.9. The most-improved multi-modular SCIP design cost and the required computational time using pure shortest-path, iMAPR, IRE and PIR.. 106

Table 9.10. Qualitative Comparison of the Span Elimination Heuristics. 111

Table 11.1. The new set of non-essential spans for Net B. 124

Table 11.2. The most-improved SCIP design results using IRE(multi) with Set 1 and Set 2 non-essential spans. 125

Table B.1. Cost results obtained from SCIP(Chee Yoon) and SCIP(Morley)......................... 145

Table B.2. Cost results obtained from SCIP(Chee Yoon) and SCIP(Morley) - Revised......... 146

Table D.1. MAPR(fall) results. 166

List of Figures

Figure 1.1. The basic elements in a coverage-oriented multi-ring network design. 2

Figure 1.2. Illustration of the span elimination concept. 4

Figure 1.3. Span elimination 'backfired'. 5

Figure 2.1. The approaches used by Span Elimination strategy. (a) Pre-Processor to Multi-Ring Design. (b) Post-Processor to Multi-Ring Design. 10

Figure 2.2. Different approaches that are used to detour demand (1-4) routing paths. (a) Span-Level. (b) Path-Level 11

Figure 2.3. Illustration of the effects of maximum cycle size constraint on essential spans. (a) Original network graph. (b) Span (2-3) and (2-4) become essential when the maximum cycle size is set to three. 14

Figure 2.4. Topology of Net A. 17

Figure 2.5. Topology of Net B 18

Figure 2.6. Topology of Net C 19

Figure 2.7. Topology of Net D. 20

Figure 3.1. SCIP block diagram. 22

Figure 3.2. The upper bound for the number of feasible designs. 25

Figure 3.3. Effects of modular capacity on single modular SCIP design cost (Nets A-D, pure shortest-path). 32

Figure 4.1. Example of MAPR concept: (a) Original network topology and spans' capacity requirements. (b) Remaining five demands from node 2 to node 6 routed using shortest-paths. 33

Figure 4.2. Example of demands routed with OC-12 oriented flow aggregation. 34

Figure 4.3. Example of eliminating a non-essential span resulting from MAPR. (a) Span 3 is eliminated by OC-12 aggregating routing. (b) OC-12 SCIP result. 35

Figure 4.4. Block diagram: Implementing MAPR. 36

Figure 4.5. Periodic MAPR pseudo-cost function for d_i=10 and M=12. (a) *rise*. (b) *impulse*. 39

Figure 5.1. An example illustrating the concepts of IRE. 42

Figure 5.2. IRE Breadth-First Search (BFS) Tree. 43

Figure 5.3. Block Diagram: Implementing IRE. 45

Figure 5.4. Communications between IRE program and SCIP 47

Figure 6.1. Net A: Minimal bi-connected sub-graph and non-essential spans. 49

Figure 6.2. Net B: Minimal bi-connected sub-graph and non-essential spans. 50

Figure 6.3. Net C: Minimal bi-connected sub-graph and non-essential spans. 51

Figure 6.4. Net D: Minimal bi-connected sub-graph and non-essential spans. 52

Figure 6.5. Effects of modular capacity on design cost with span eliminations by MAPR (Nets A-D, rise pseudo-cost function). 56

Figure 6.6. Effects of modular capacity on design cost with span eliminations by MAPR (Nets A-D, impulse pseudo-cost function). 57

Figure 6.7. SCIP results for Net A using IRE. 60

Figure 6.8. SCIP results for Net B using IRE. 61

Figure 6.9. SCIP results for Net C using IRE.. 61

Figure 6.10. SCIP results for Net D using IRE. 62

Figure 6.11. Typical IRE curve pattern. (a) Typical IRE cost vs. eliminations curve. (b) Typical ring network characteristics for each region (this model is obtained using IRE results for Net A). 63

Figure 6.12. Percentage Total Network Cost Savings for MAPR/SCIP(rise) and IRE/SCIP. 65

Figure 6.13. Scatter plot for MAPR and IRE results normalized to corresponding shortest-path and no span elimination results to show the relationship between solution quality and computational time.. 66

Figure 7.1 Desired c_i^+ and c_i^- exponential curves. 73

Figure 7.2 An example of a non-periodical pseudo-cost function for iMAPR with $d_i = 15$ for $0 \leq w_i \leq 192$. 78

Figure 7.3. Pseudo-code for calculating iMAPR pseudo-cost for all w_i. 79

Figure 7.4. Schematic depiction of iMAPR. 80

Figure 8.1. A network example where demands are routed using pure shortest-path. 83

Figure 8.2. IRE results for the network in Figure 8.1 using multi-modular SCIP. (a) The full network graph with demands routed using pure shortest-path. (b) Span (2-3) is eliminated. (c) Span (2-5) is eliminated. (d) Following the SCIP design in (b), span (2-5) is eliminated. 84

Figure 8.3. Phase I of PIR where the supermodular capacities of span (3-6) have been re-routed. 86

Figure 8.4. PIR Phase II results following PIR Phase I in Figure 8.3. (a) Resulting network from PIR phase one. (b) Span (2-3) is eliminated. (c) Span (2-5) is eliminated. (d) Following the elimination of span (2-3) in (b), span (2-5) is eliminated. 88

Figure 8.5. SCIP design results on the most-improved sub-graph found by PIR. 89

Figure 8.6. Phase I of PIR showing the processes that are involved in reducing the total number of forcer spans by means of re-routing the supermodular capacities. 90

Figure 8.7. Phase II of PIR: Search for the minimum number of supermodular spans by eliminating non-essential spans and re-routing trials. 91

Figure 9.1. SCIP results for Net A using IRE(multi) 102

Figure 9.2. SCIP results for Net B using IRE(multi). 102

Figure 9.3. SCIP results for Net C using IRE(multi). 103

Figure 9.4. SCIP results for Net D using IRE(multi) 103

Figure 9.5. Percentage Total Network Cost Savings for iMAPR, IRE(multi) and PIR relative to multi-modular SCIP reference design. 107

Figure 9.6. Scatter plot for iMAPR, IRE(multi) and PIR results normalized to corresponding shortest-path and no span elimination results to show the relationship between solution quality and computational time. 108

Figure 10.1. The total network cost obtained using RingBuilder v2.0 and SCIP. 114

Figure 11.1. The new minimal bi-connected sub-graph and non-essential spans for Net B 123

Figure 11.2. SCIP results for Net B using IRE(multi) with Set 1 and Set 2 non-essential spans..... 125

Figure 11.3. The proposed change to the iMAPR pseudo-cost function with d_i=15. 128

Figure B.1. SCIP Design Cost vs. Sample No. to verify SCIP(Chee Yoon) against SCIP(Morley). 145

Figure D.1. Pseudo-cost function fall for d_i=10 and M=12.. 165

Symbols and Abbreviations

α	Aggregation pressure coefficient.
$\delta_{j,i}$	SCIP design parameter : 1 if ring candidate *j* intersects span *i*, 0 otherwise.
a	No. of samples (for investigating the correlation between SCIP and TR*Labs*' RingBuilder).
A^+	A coefficient for iMAPR upper pseudo-cost function.
A^-	A coefficient for iMAPR lower pseudo-cost function.
ADM	Add-Drop Multiplexers.
B^+	A coefficient for iMAPR upper pseudo-cost function.
B^-	A coefficient for iMAPR lower pseudo-cost function.
BFS	Breadth-First Search.
BLSR	Bi-directional Line Switched Rings.
c_j	SCIP design parameter : fixed cost of ring candidate *j*.
$c_i(w_i)$	Pseudo-cost of span *i* at with working capacity w_i.
CPU	Central Processing Unit.
cycle	A path with the first and last vertices the same and with some combination of three or more vertices.
d_i	Geographical distance of span *i*.
DCS	Digital Cross-Connect Systems.
Demand (*x*-*y*)	PTP demand from source node *x* to destination node *y*.
DS-N	Digital Signal - Level N.
DWDM	Dense Wavelength-Division Multiplexing.
FCRIP	Fixed Charge and Routing Integer Programming.
G	Constant Multiplier (iMAPR pseudo-cost function)
iMAPR	Iterated Modular Aggregated Pre-Routing.
impulse	MAPR pseudo-cost function.

IP	Integer Program.
IRE	Iterated Routing and Elimination.
k	The number of non-essential spans.
MBG	Minimal Bi-Connected Graph - the minimum number of spans required by the graph to achieve bi-connectivity.
MAPR	Modular Aggregated Pre-Routing.
$M(w_i)$	The next modular capacity that is smaller or equal to w_i where $M(w_i) \in \{0,12,48,96,192\}$.
m	Modular capacity.
m_j	SCIP design parameter: modular capacity of ring candidate j.
M	Modular capacity.
N	Number of vertices.
n_R	SCIP design parameter: number of ring candidates in set R.
OxC	Optical Cross-Connects.
OADM	Optical Add-Drop Multiplexers.
OC-N	Optical Carrier - Level N.
Path $(x\text{-}y)\text{-}(y\text{-}z)$	A path using span $(x\text{-}y)$ and $(y\text{-}z)$.
PIR	Post Inspection and Re-routing.
PTP	Point-To-Point.
Q	Upper bound for the number of ring candidates.
R	SCIP design parameter: set of ring candidates.
r	Pseudo-cost curve number, used in iMAPR.
Ring $(x\text{-}y)\text{-}(y\text{-}z)\text{-}(z\text{-}x)$	A ring created by using span $(x\text{-}y)$, $(y\text{-}z)$ and $(z\text{-}x)$
rise	MAPR pseudo-cost function.
S	SCIP design parameter: number of spans in the network.
SCIP	Span Coverage Integer Program.
SNIF	Standard Network Interface File.

SONET	Synchronous Optical Networks.
Span (x,y)	A span connecting node x and node y.
STS-N	Synchronous Transport Signal - Level N.
UPSR	Uni-directional Path Switched Rings.
v_i	Vertex i.
w_j	SCIP design parameter: working capacity of span i.
$\overline{w}$	Average Network Working Capacity.
WDM	Wavelength-Division Multiplexing.
X_j	SCIP design parameter: integer number of ring candidate j from cycle set C.

1.0 Introduction

There has recently been increasing pressure on optical capacity, brought about by the tremendous surge in both narrowband and broadband services. This unprecedented growth in demand for transport capacity, largely driven by Internet applications, has driven network carriers to draw solutions for their, soon-to-be exhausted, fiber networks. With the advent of optical components (e.g. tunable optical sources, filters, amplifiers and wavelength converters), optical multiplexers and cross-connects are now becoming available, enabling the much anticipated Wavelength-Division Multiplexing (WDM) based optical networks. With the favorable economics of WDM networks, network carriers have found a straightforward solution to their network problems. To date, WDM is the widely regarded answer for the very high bandwidth requirements, and the foundation of optical transport networks in the future [1]-[3].

There has been a great deal of development on the implementation and technology of WDM networks [4]-[6]. WDM systems with 32 wavelengths capable of supporting SONET systems operating at 10 Gb/sec each are commercially available. Denser multiplexing schemes, such as Dense WDM systems with 160 wavelengths developed by Nortel Networks [7], are currently under field trial by MCI Worldcom [8]. When combined with SONET systems operating at 10 Gb/sec, this DWDM system is capable of delivering a total aggregate capacity as high as 1.6 Tb/sec, an equivalent of 28 million simultaneous Internet connections over a single strand of fiber. With such a high concentration of traffic on a single fiber, a single point of failure in a fiber network can potentially be disastrous. A single-fiber failure scenario with no fast protection scheme could be catastrophic to the network carriers, and could have significant financial and social consequences: therefore, some form of fast network restoration is essential.

In recent years, much work has been done on solving the problem of protecting optical networks from a single point of failure. The work can basically be divided into two approaches, namely, mesh networks and ring networks [9,10]. Although the mesh protection scheme is known to have the lowest redundancy in transmission capacity (since it uses an undedicated spare capacity routing), ring architectures are often preferred in practice because of their simpler and faster switching mechanism, typically under 150 msec [11]. This point has since been debated with the appearance of fast mesh restoration schemes [12] and increasingly fast optical technologies [13]. However, though not as capacity-efficient as mesh networks, ring networks can be more economical in metropolitan applications where the dominant costs are incurred in nodal equipment [14]. For this reason, SONET rings are now in common use for survivable transport networking and are promising architectures for protecting optical networks from single-point failure.

The problem of designing WDM optical networks based on ring architectures is logically similar to designing SONET ring-based networks [15]. However, the design of multi-ring networks is known to be an exceptionally complex combinatorial optimization problem [16,17]. In recent years, a number of approximate design methods have been proposed for the multiple-ring design problem [16]-[21] which approach the design as a form of the min-cost graph-covering problem. Some schemes take a capacitated coverage view (the ring capacities placed must be adequate for carrying all demand crossing the underlying span) [16]-[20], or an uncapacitated, purely logical coverage view (at least one ring covers every span) [17,21].

The basic building blocks for a multi-ring network design are shown in Figure 1.1. At the highest level, the problem inputs include the network topology, the demand matrix, the ring technologies that are currently under consideration, and the cost model for the network equipment. The SONET ring technologies that are commonly used for survivable transport networking are Bi-directional Line-Switched Rings (BLSR) and Uni-directional Path-Switched Rings (UPSR) [22], which have already undergone a fairly extensive study from the optimization perspective. The solution is a min-cost ring design that consists of a set of ring systems specifying the type (BLSR or UPSR), capacity (modular capacity of each ring system), assignment of demands (the working capacities carried by each ring's span modules) and inter-ring transitions (the connections between each ring system) together to satisfy all the capacity requirements in the network at minimum cost.

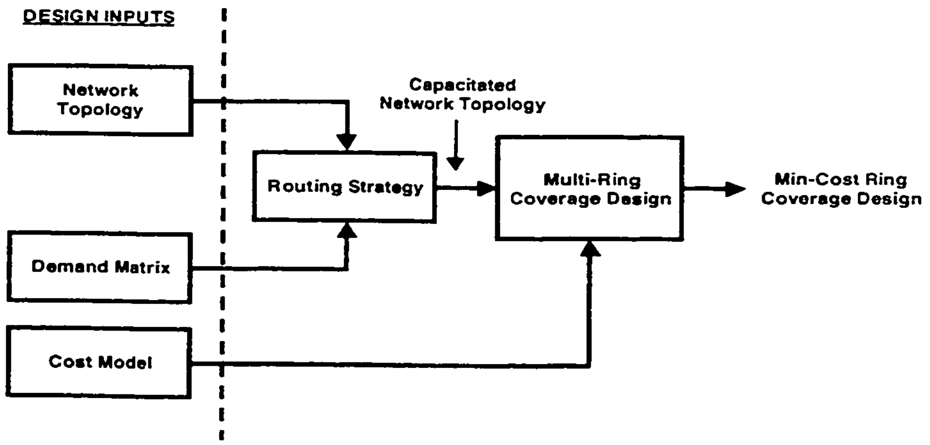

DESIGN INPUTS
Network Topology
Capacitated Network Topology
Routing Strategy
Multi-Ring Coverage Design
Min-Cost Ring Coverage Design
Demand Matrix
Cost Model

Figure 1.1. The basic elements in a coverage-oriented multi-ring network design.

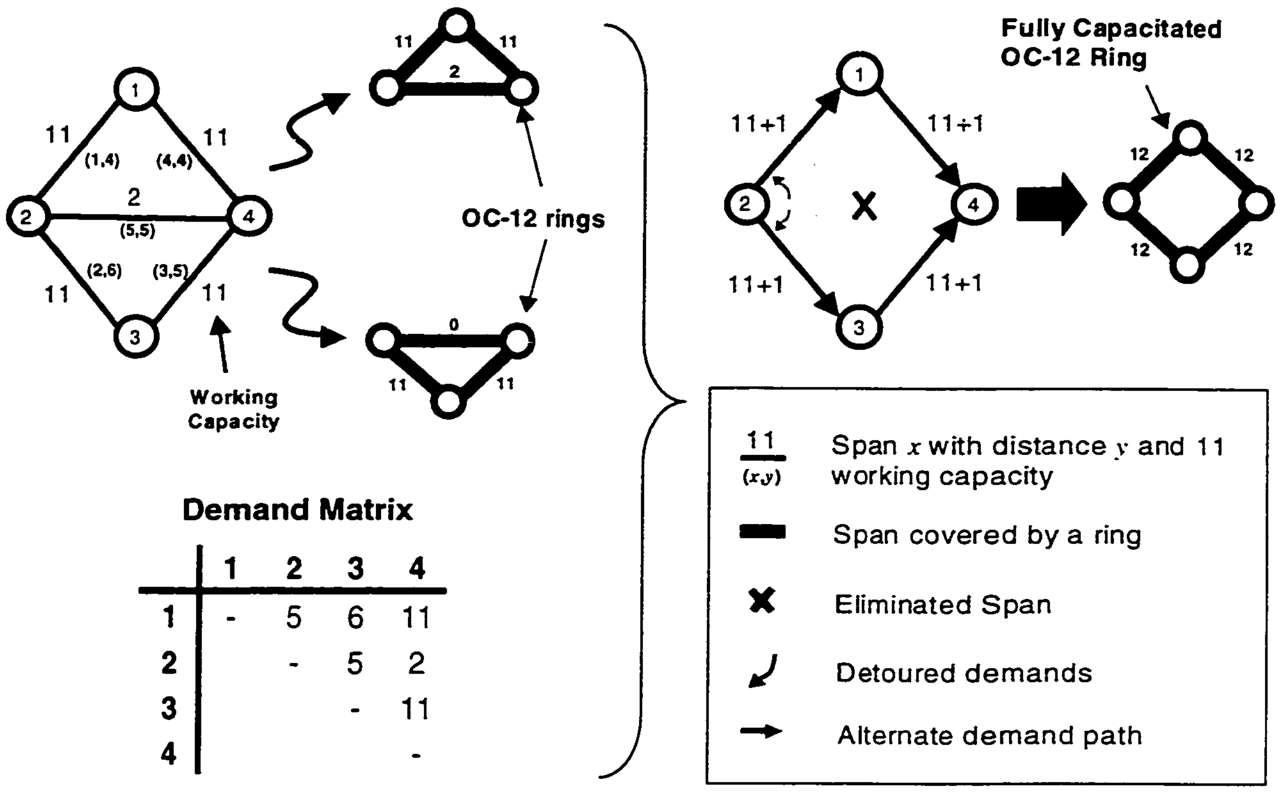
The *Network Topology* is the underlying fiber graph that consists of nodes and spans where the nodes are the Add Drop Multiplexers (ADM) or, in the context of an all-optical network, Optical Add Drop Multiplexers (OADM), and the spans are the physical fiber. Here it is assumed that the physical characteristics of the network topology, e.g. node placement, span distances, the link budget requirements of each span, have been investigated and realized.

The *Demand Matrix* is derived from Point-To-Point (PTP) customer traffic requirements. Here, each demand represents an individual digital signal carrier, which could be a DS-3*, STS-1 or STS-N or a wavelength. The *Cost Model* for the physical network equipment represents the ADM common equipment cost, the Add/Drop port cost, the ring-to-ring transition port cost and the fiber cost. The adopted cost model generally follows a certain level of economy-of-scale. The *Routing Strategy* routes the PTP demands from the demand matrix onto the given network topology, thereby defining the "w_i" capacity requirements on each span. The *Multi-Ring Coverage Design* is an algorithm where, for the given capacitated network topology and the corresponding cost model, the objective is to find a set of survivable rings that fully serves all the demands with minimum total cost. The result is the min-cost ring coverage design.

1.1 Concept of Span Eliminations

In general, the capacity requirements of each span are determined by routing demands over the shortest path from end-to-end based on the geographical fiber distances. Intuitively, this can lead to coverage solutions with one or more very low-utilized rings, that is, rings that were placed, in essence, according to the strict coverage requirement but which serve little demand. Consider the simple graph and demand matrix in Figure 1.2 where the capacities accumulated on each span have been determined by the shortest-path routing of the PTP demands over the fiber graph. Assuming OC-12 4-fiber BLSRs, a minimum of two rings is required in a min-cost 'coverage' design. Moreover, both of these rings are forced to cover span (2-4), though together they have only 2/24th capacity utilization on that span.

* The rate for DS-3 is 44.736 Mb/s and the rate for STS-1 is 51.880 Mb/s. Note that this work focuses on these highly multiplexed transport signals representing a unit of PTP demand from the source node to the destination node depicted by the demand matrix, not on individual calls, sessions, packets, etc.



Demand Matrix

	1	2	3	4
1	-	5	6	11
2		-	5	2
3			-	11
4				-

(a) (b)

Figure 1.2. Illustration of the span elimination concept.

In Figure 1.2, it is fairly apparent by inspection that "eliminating" span (2-4) would force the demand flow using span (2-4) to take another route or routes (with demand bundle splitting) and, as constructed in Figure 1.2b, would result in a single perfectly filled OC-12 ring to cover the network graph. Note that the capacity utilization of the ring increases although fewer rings are used due to eliminating span (2-4). In the language of [16], the capacity and capture efficiency are both improved illustrating that cost optimization based on span elimination is possible and feasible. The example, of course, shows the ideal situation. The challenge in practice is to identify those spans of a graph which, for a given demand pattern and ring modularities, will have desirable effects, if eliminated.

However, if the wrong span is eliminated, the results can backfire: total (inter-ring) routing costs can increase due to excess detouring from the shortest-path, and ring counts can increase due to exceeding a modularity threshold. For example, consider the same network graph and demand pattern as that illustrated in Figure 1.2a. Suppose we modify the demand pair (2-4) to three units; using the shortest-path, span (2-4) in the resulting capacitated network graph will have three units of demand as shown in Figure 1.3 below.

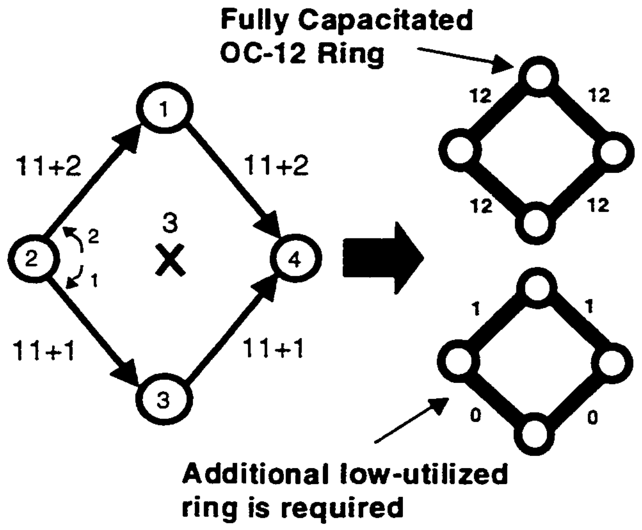

Fully Capacitated OC-12 Ring
12
12
12
12
1
11+2
11+2
3
2
1
2
4
11+1
11+1
3
1
1
0
0
Additional low-utilized ring is required

Figure 1.3. Span Elimination 'backfired'.

If span (2-4) is eliminated from the network graph and the affected demands are detoured using other paths, the resulting sub-graph requires two rings, one of which is a single fully utilized OC-12 ring and the other a copy of the ring, but with only 2/48th ring capacity utilization. The overall result is a significant increase in the total ring design cost (additional network equipment is required), illustrating that there are situations where span elimination has reverse effects; instead of reducing cost, an increase in cost is obtained.

To summarize, span eliminations can lead to a better-optimized min-cost multi-ring design; however, span eliminations can backfire. These two factors have led to the development of efficient systematic methods to implement span eliminations. Essentially, physical network equipment, such as ADMs, amplifiers and optical fibers, are saved in exchange for additional port cards in the existing ADMs. Although the motivation is the potential cost benefit of these exchanges, the selection of which spans to eliminate remains crucial. It is known that experienced manual planners, with some trial and error, can fairly effectively visualize the span elimination opportunities in complex real-sized problems [23]. However, like any other similar design problems, a systematic algorithmic approach would be desirable to consistently attain the most-improved results.

At the practical perspective, implementing span eliminations can have significant impact on decision making especially when carriers are planning their next growth increment. Using span eliminations, planners can provide options and test different strategies to optimize the use of their backbone network. For example, planners may find that, they can stop using an existing facility route* for the next planned growth increment and least it to potential customers, while still be able to maintain the current customer demands by using alternate facility routes. In addition, planners may want to move selected demands off an existing facility route to retire old equipment or to completely disuse a facility route and sell it. In other words, at the planner's view, the facility route for use is available for use if the planners which to establish a transmission line over it, but the design is not compelled to do so. Therefore, span eliminations can provide planners with the information that they need, enabling them to draw out the best strategy for their backbone network.

1.2 Research Objectives

The objective of this thesis is to explore a different dimension of cost optimization in ring network design based on topology reduction using span eliminations. Specifically, this thesis proposes several systematic strategies to effectively implement the concept of span eliminations and to provide good solutions to the optimized min-cost multi-ring design. It is hypothesized that by removing the proper spans from the network, the total network design cost can be significantly reduced.

The effectiveness of the proposed strategies will be evaluated and compared to designs obtained using pure shortest-path routing without span eliminations, based on the reduction or increase of the total network cost and runtime required by each strategy to achieve the given solutions.

1.3 Outline

Chapter 2 contains an introduction to the key concepts and terminology for span eliminations and a review of the prior work relating to this area. This chapter also describes the characteristics of the network models used in this research.

* facility route refers to a potential network span.

Chapter 3 introduces the Span Coverage Integer Program (SCIP) which will be used to evaluate the proposed span elimination strategies. SCIP is a pure (ideal) capacitated span coverage integer program that strictly finds a min-cost subset of rings that are required to cover all the non-zero span working capacities in the network. Some experiments, which used this coverage tool on networks using shortest-path routing, are described. These benchmark results will be used to compare the efficiency of the proposed span elimination heuristics. With the simplicity and repeatable aspects of SCIP (also to be described in this chapter), it is also claimed that it can be applied effectively as a surrogate for the full-blown ring coverage heuristics, e.g. RingBuilder[16], in purely relative or comparative design studies.

Chapter 4 and **Chapter 5** present the first two span elimination heuristics that were developed and studied in this thesis, namely Modular Aggregated Pre-Routing (MAPR) and Iterated Routing and Elimination (IRE). The concepts and methods of implementing MAPR and IRE are described in these chapters.

Chapter 6 describes experiments performed using MAPR and IRE on the selected network models evaluated using SCIP. These heuristics are compared against the results using shortest-path routing (also using SCIP) in terms of solution quality and computational runtimes. An interim conclusion summarizes several observations obtained from the experiments which affect the subsequent development of other span-elimination heuristics.

Motivated by the findings in Chapters 4 to 6, which outline several limitations and provide deeper understanding of span eliminations, **Chapter 7** details the modifications and improvements that are made to MAPR. The improved MAPR heuristic is called Iterative MAPR (iMAPR), which basically implements a series of MAPR runs.

Chapter 8 introduces a new span elimination heuristic called Post-Inspection and Re-routing (PIR). From the evaluation of MAPR and IRE, it was realized that there are situations where these heuristics fail to give good solutions. A simple example of this situation is presented and exploited to develop PIR. The concepts and methodologies for implementing PIR are described in this chapter.

Chapter 9 contains the results of the experiments performed using the improved strategies iMAPR, and IRE and the new strategy, PIR. These heuristics are evaluated in terms of their solution quality and computational runtimes. A conclusion is drawn here on which span elimination heuristic performs the best overall together with a detailed summary of a comparison between the characteristics and performance of each investigated span elimination heuristic.

Chapter 10 evaluates SCIP against the full-blown ring design heuristic implemented in the software package, RingBuilder. This chapter essentially collects sample results using both of these coverage tools and finds out how well these samples are statistically correlated.

The thesis is concluded and summarized in **Chapter 11** with suggestions for future research.

2.0 Span Elimination Concepts & Prior Work

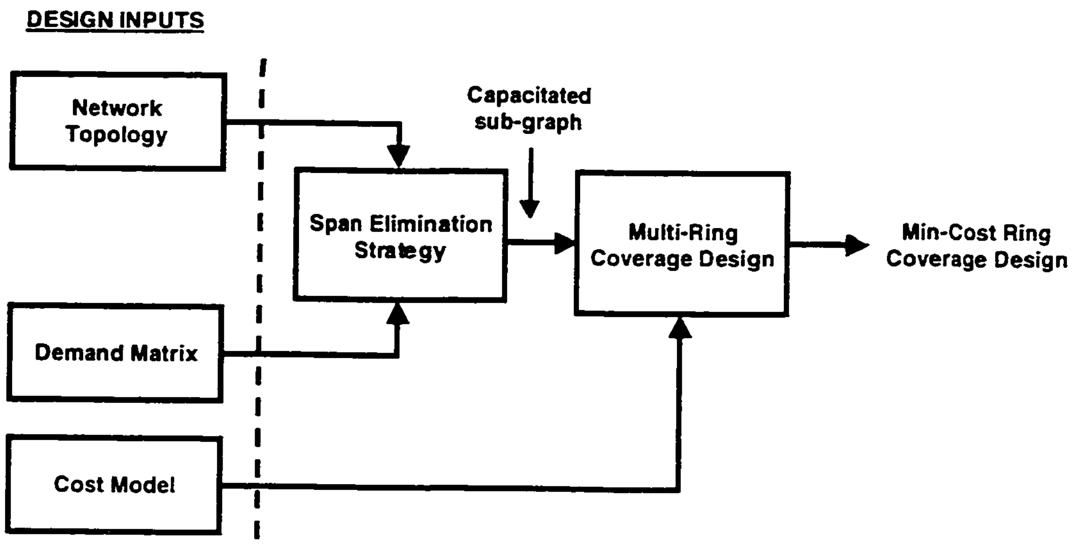
This chapter is dedicated to describing the concepts and methodologies of span elimination. Later in the chapter, the experimental requirements are laid out as well as the network models that were investigated throughout this research.

2.1 Introduction to Span Elimination

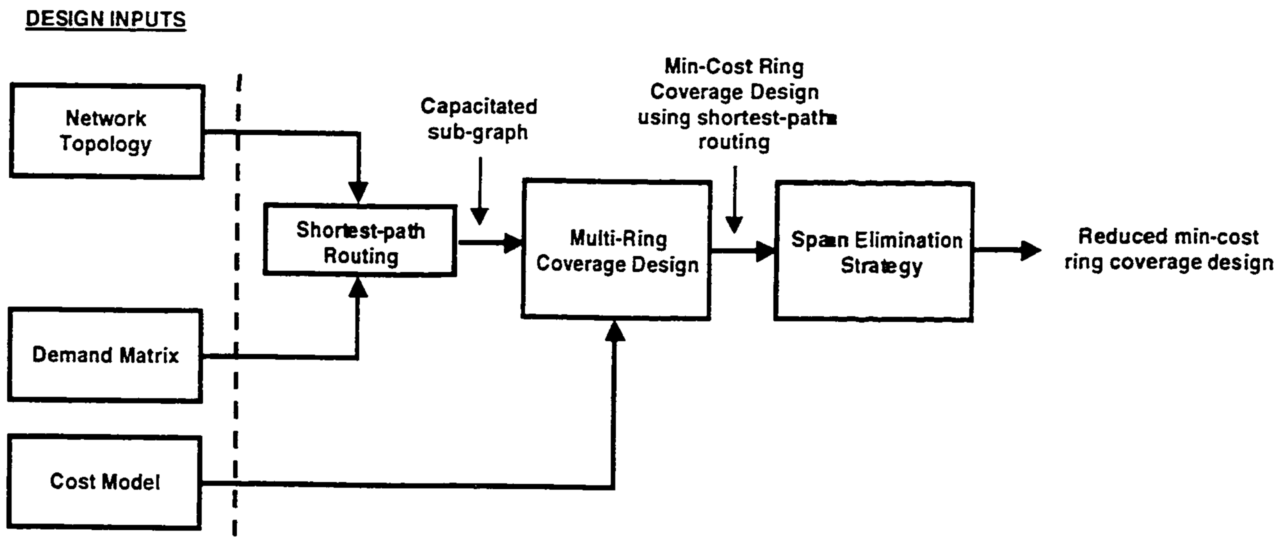
As mentioned in the opening chapter, span elimination should be an integral part in a min-cost multi-ring network design process. In a capacitated network, where the capacity requirement in each span is determined by shortest-path routing over the network graph, it is obvious that this can lead to some spans having little demand to serve. By themselves, such spans will ultimately lead to coverage solutions having one or more low-utilized rings. Multi-ring coverage algorithms are unable to harness the knowledge of these low-capacitated spans because they are required to cover all capacitated spans. This is where a span elimination strategy comes in as a systematic method of determining and eliminating spans that will deliver a better optimized min-cost multi-ring coverage design.

The development of span elimination strategies is approached in two systematic categories. First, the span elimination strategy is viewed as a *pre-processor* to the multi-ring coverage design as shown in Figure 2.1a. The goal of this approach is to resolve the issue of low-capacitated spans by avoiding their use in the resulting capacitated network. This is done by detouring the demand routing paths* either at the span-level or at the path-level (refer to Figure 2.2) to take up the slack capacity of other spans rather than restricting them solely to achieving the shortest distance as with a shortest-path algorithm. The result of this modification on the routing algorithm could deliver better network efficiency and increase the number of uncapacitated spans: the fact that demand paths are detoured to fill up other spans will leave some spans unused. These unused spans, satisfying the conditions to be discussed later in the Chapter 2.3.1, are eliminated and a multi-ring coverage design is obtained from the resulting sub-graph (with unused spans eliminated).

* Two edges are said to be *adjacent* if they share a common node. A *path* is defined as a sequence of adjacent edges $(v_1, v_2), (v_2, v_3), \dots, (v_{k-2}, v_{k-1}), (v_{k-1}, v_k)$, in which all the vertices $v_1, v_2, \dots, v_k$ are distinct.

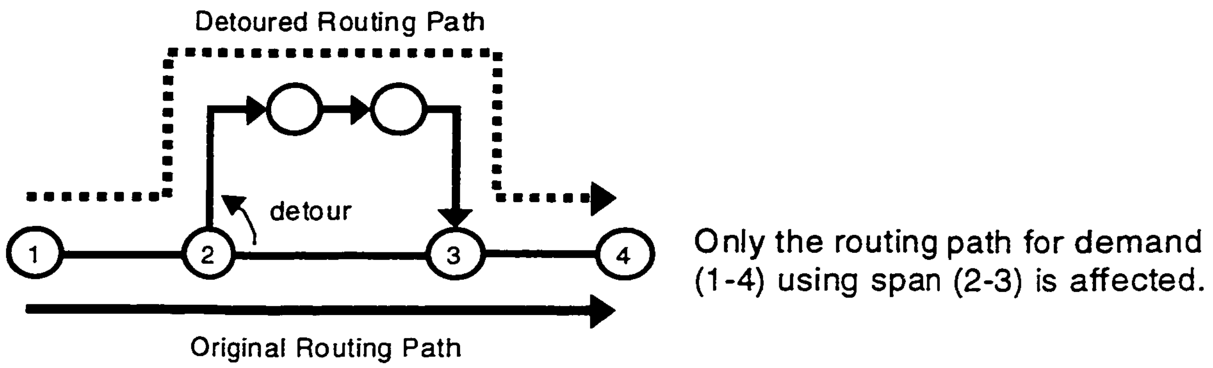

DESIGN INPUTS
Network Topology
Demand Matrix
Cost Model
Span Elimination Strategy
Capacitated sub-graph
Multi-Ring Coverage Design
Min-Cost Ring Coverage Design

(a)

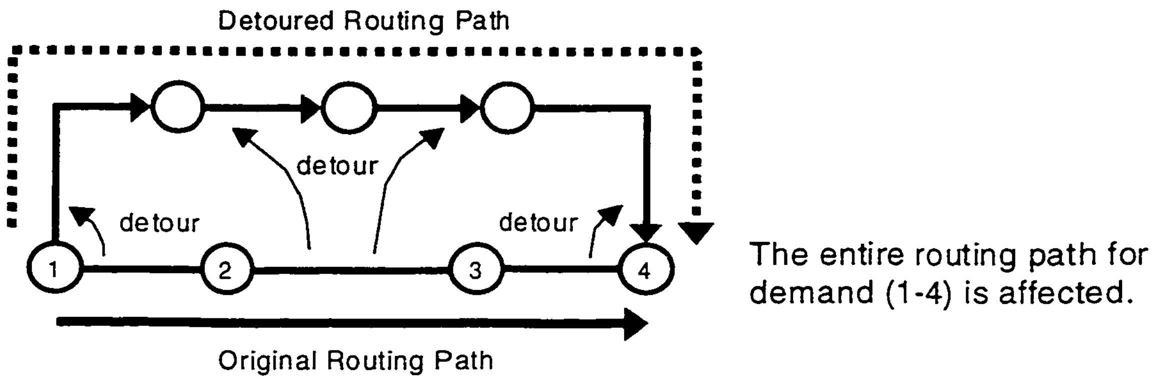

DESIGN INPUTS
Network Topology
Demand Matrix
Cost Model
Shortest-path Routing
Capacitated sub-graph
Multi-Ring Coverage Design
Min-Cost Ring Coverage Design using shortest-path routing
Span Elimination Strategy
Reduced min-cost ring coverage design

(b)

Figure 2.1. The approaches used by Span Elimination strategy. (a) Pre-Processor to Multi-Ring Design. (b) Post-Processor to Multi-Ring Design.


Detoured Routing Path
detour
1
2
3
4
Only the routing path for demand (1-4) using span (2-3) is affected.
Original Routing Path

(a)



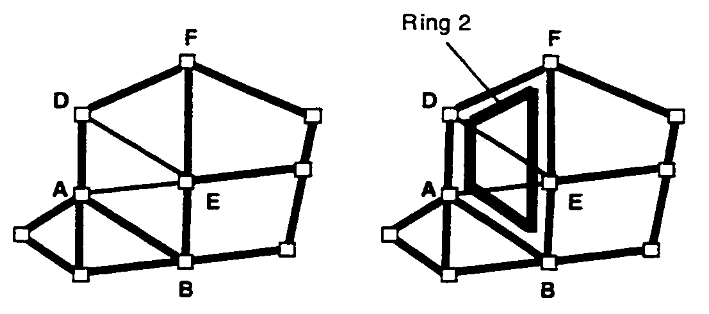
(b)

Figure 2.2. Different approaches that are used to detour demand (1-4) routing paths. (a) Span-Level. (b) Path-Level.

In a second basic approach, span elimination is intended to serve as a *post-processor* to the multi-ring coverage design (see Figure 2.1b). It starts with a coverage design resulting from the shortest-path routing. Spans that satisfy the conditions to be described in Chapter 2.3.1 are identified and eliminated. Affected demands that are served by the eliminated spans are detoured over other alternate paths taking up the slack capacity of other spans to their modular sizes. The idea is that if this type of procedure is successfully completed without pushing some other rings over their modular boundaries, a better and improved design may be obtained.

2.2 Prior Work in Span Elimination

The first mention of the span elimination issue can be traced back to the work by Flanagan [23]. Flanagan briefly describes the application of span elimination in ring design suggesting that spans with less traffic be eliminated. The traffic affected by the elimination can then be re-routed over other rings. The idea is as quoted [23]:



"Fig. 8. Span Elimination" (adapted from [23])

"Spans with less traffic can be eliminated from the ring design. If span DE in Figure 8 has very little traffic, it can be eliminated from the ring design. Traffic between D and E can be rerouted on ring 2 through F. Combining these two techniques – multiple spans and span elimination – ensures that the ring design closely matches the traffic demand."

More work was done by Doshi and Harshavardhana [18] in their work called *Ring Deloading*. Their technique specifically identifies the under-utilized rings which they call the *straggler rings*. These rings are the direct result of demands being placed on spans which are over their maximum modular capacity, where additional ring(s) are required to cover the excess capacity. The object of ring deloading is to see if the demands in the straggler rings can be removed by re-routing them over alternate paths without pushing some other rings over their modular boundaries. These rings can then be eliminated to significantly reduce the total network cost.

Another on-going work that implements span elimination is by Morley and Grover [30,34] in Fixed Charge and Routing (FCRIP). FCRIP is one of the proposed mathematical programming approaches to designing a ring network at min-cost. It simultaneously solves for the ring set and the demand routing decisions for the ring design. As such, this approach does not implicitly assume a strict full span coverage requirement but inherently solves the problem of sub-graph topology determination, which involves one or more span eliminations.

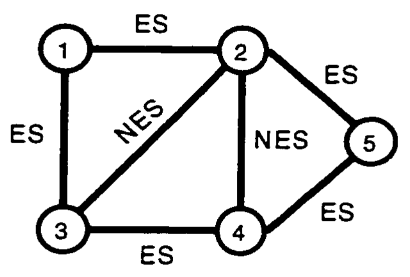
2.3 Research Methods

Before reaching the core of this research, several preliminary considerations have to be established. In the sub-chapters that follow, the key terminologies, requirements, parameters and conditions which are adapted in this research are discussed.

2.3.1 Network Topology Considerations

In order for a graph to be survivable, it has to be bi-connected. In a bi-connected graph, each pair of vertices *a* and *b* in the graph has at least two vertex-disjoint paths between them. This requirement makes it feasible to protect the network from a single point of failure (either a span failure or a node failure). Having satisfied this prerequisite, it is then essential to identify and exclude from consideration certain spans that cannot be eliminated, because their removal would result in the graph becoming one-connected. Any span which is required in order for rings to cover all the network nodes when the limit in the number of ring nodes is taken into consideration must also be excluded from removal. First a minimal bi-connected sub-graph (MBG) of the initial fiber graph must be found, where a minimum number of spans is required by the graph to retain full bi-connectivity* for all node pairs. This sub-graph is then said to consist of *essential spans* for bi-connectivity, a necessary condition for a ring cover to exist. Essential spans are, by definition, excluded from consideration for span elimination.

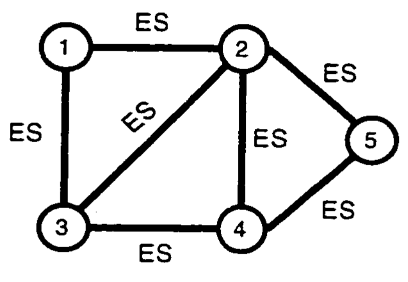
In addition to the essential spans derived from MBG considerations, other spans may be added to the essential spans set if their removal would prohibit one or more other spans from being covered given a design maximum cycle size (number of hops in a ring). For instance in Figure 1.2, if the maximum cycle size were three, span (2-4) would be essential because the minimum cycle size to cover the resulting sub-graph if it were eliminated in Figure 1.2b would be four. The remaining non-essential spans are eligible for the span elimination process. Figure 2.3 gives another explicit example of this occurrence. Figure 2.3a shows the original network graph, the essential spans and the non-essential spans when there is no limitation in the maximum cycle size. When the maximum cycle size is set to three, span (2-3) becomes essential because if span (2-3) were eliminated, span (1-2), (1-3) and (3-4) would become un-coverable since the minimum cycle size for cycles to cover these spans now is four. This explanation also applies for span (2-4): span (2-4) is essential since if this span were eliminated, spans (2-5), (3-4) and (4-5) would become un-coverable.



Essential Spans (ES)	Non-essential Spans (NES)
(1-2)	(2-3)
(1-3)	(2-4)
(2-5)	
(3-4)	
(4-5)	

Maximum Cycle Size = 4

(a)



Essential Spans (ES)	Non-essential Spans (NES)
(1-2)	
(1-3)	
(2-3)	
(2-4)	NONE
(2-5)	
(3-4)	
(4-5)	

Maximum Cycle Size = 3

(b)

Figure 2.3. Illustration of the effects of maximum cycle size constraint on essential spans. (a) Original network graph. (b) Span (2-3) and (2-4) become essential when the maximum cycle size is set to three.

A distinction needs to be made here. Flanagan's work describes the non-essential spans as spans that carry less traffic. However, it is apparent here that non-essential spans need not necessarily be spans that serve little demand. The important issue is to eliminate spans (regardless of the affected demand size) where the affected demands can be re-routed to take up the slack capacities of other spans without introducing additional rings to the coverage design.

There is a difference between *minimal* bi-connected sub-graph (implemented in this research) and *minimum* bi-connected sub-graph. Minimal bi-connected sub-graph describes a sub-graph that retains the full bi-connectivity requirements with minimum number of spans and no further eliminations is possible. There can be more than one minimal bi-connected sub-graph. But, there can only be one minimum bi-connected sub-graph, which refers to a sub-graph that retains the full bi-connectivity requirements and has minimum total span weight (either the total distance or the total cost of spans).

* Note that there may be more than one distinct but equivalent MBG for a given network graph. In this situation, one of the MBGs is arbitrarily chosen.

The algorithm to determine a minimum bi-connected sub-graph implements an exhaustive search strategy to remove a span and test the resulting sub-graph for bi-connectivity. For a given graph with N nodes and S spans, there can be a maximum of S-N spans that can be removed from the graph such that the remaining sub-graph will still be bi-connected (assuming a bi-connected sub-graph with only degree two nodes). Therefore, there are 2^{S-N} number of combinations where S-N spans can be removed. Thus, the upper bound complexity to implement exhaustive search here is $O(2^{S-N})$. And on top of that, the algorithm for testing a sub-graph for bi-connectivity is described in [26] and has a complexity of $O(N+S)$. Here, the overall complexity for the minimum bi-connected sub-graph algorithm is $O(2^{S-N}(N+S))$. Due to the algorithm complexity, this algorithm is not utilized in this research (therefore, minimum bi-connected sub-graph is not considered for this research).

Instead, for simplicity sake, a minimal bi-connected sub-graph is chosen. In this research, the MBGs (as described earlier) for each respective test network are determined by inspection - a span is removed from the graph and after each removal, the following two criteria must be met:

1. Ensure that the resulting sub-graph is still bi-connected.
2. Verify that rings can completely cover the network nodes, taking the limit in number of ring nodes into consideration.

Here, all the spans are tested and removed until further removing any span from the resulting sub-graph will violate one of the above criteria. The objective here is to find the bi-connected sub-graph with minimum number of spans [*]. The set of spans that are removed are then defined as the set of non-essential spans.

Here, it is realized that there can be more than one MBG for a given network graph and choosing different MBGs can lead to a different set of solutions. A test case is completed to partially assess this issue in Chapter 11 and is otherwise left for future investigation.

[*] Since there are no guarantee that based on the two criteria, a minimal bi-connected sub-graph is obtained and since this procedure is only done once for each test network, the procedure is repeated again (several times, if required) to ensure that the resulting bi-connected sub-graph is in-fact a minimal bi-connected sub-graph.

2.3.2 Network Demand Routing

Throughout this work, the benchmark coverage design results used to compare the performance of the span elimination strategy will employ the shortest-path algorithm for working demand routing. Specifically, the pure shortest-path based on Dijkstra's algorithm, which can be found in [24], will be employed.

2.3.3 Cycle Generator

Given the maximum cycle size (maximum number of logical hops permitted in a ring), the cycle generator searches the network graph exhaustively to find all the cycles that fall within the specified maximum boundary. This cycle generator implements Johnson's algorithm, which can be found in [25,26]. A cycle is defined as a path with the same first and last vertices, and having a combination of three or more vertices.

2.4 Experimental Network Models

Four experimental networks and demand matrices have been used to test the spar elimination strategies. The networks are labeled Net A, Net B, Net C and Net D. Some characteristics of each network are detailed in Table 2.1. Figures 2.4 to 2.7 show the topologies of these networks. The corresponding span numbers and distances are detailed in Appendix A, which contains the networks' standard network interface files (SNIF).

Table 2.1. Characteristics of the test networks.

Network	No. of Nodes	No. of Spans	Average Network Degree	No. of Demand pairs	Total Demands
A	11	23	4.18	38	80
B	30	59	3.93	263	1557
C	50	65	2.52	233	380
D	101	136	2.71	585	877

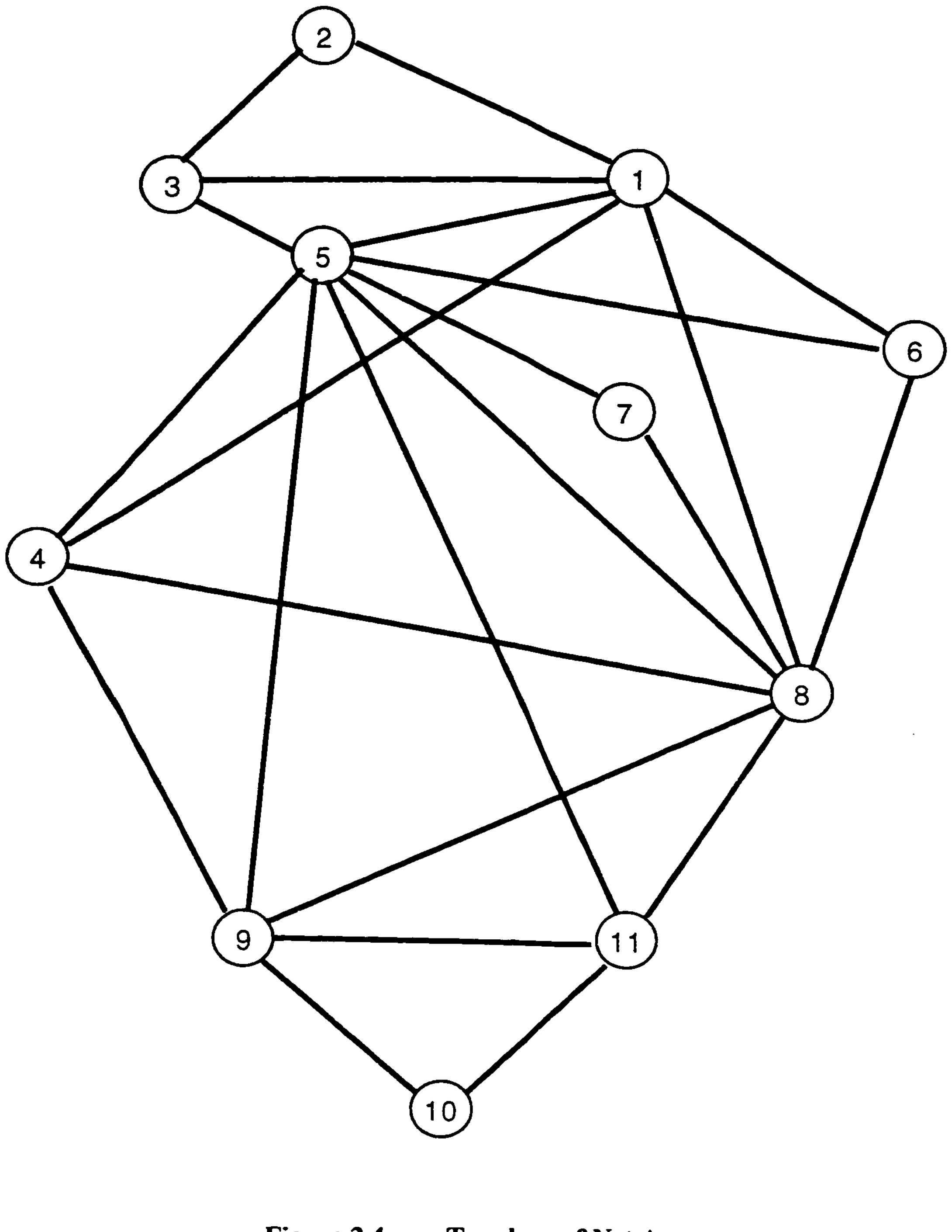


Figure 2.4. Topology of Net A.

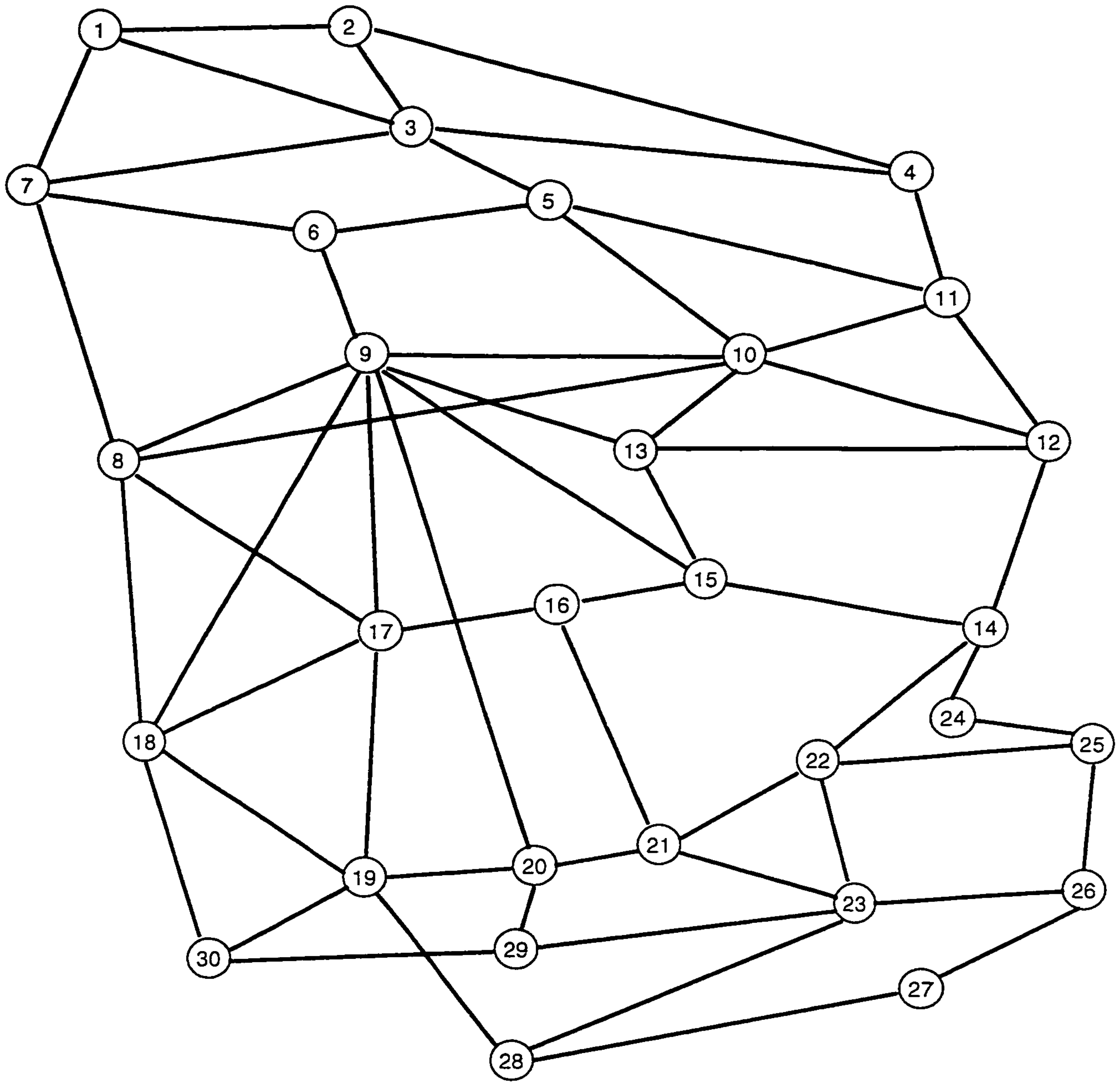


Figure 2.5. Topology of Net B.

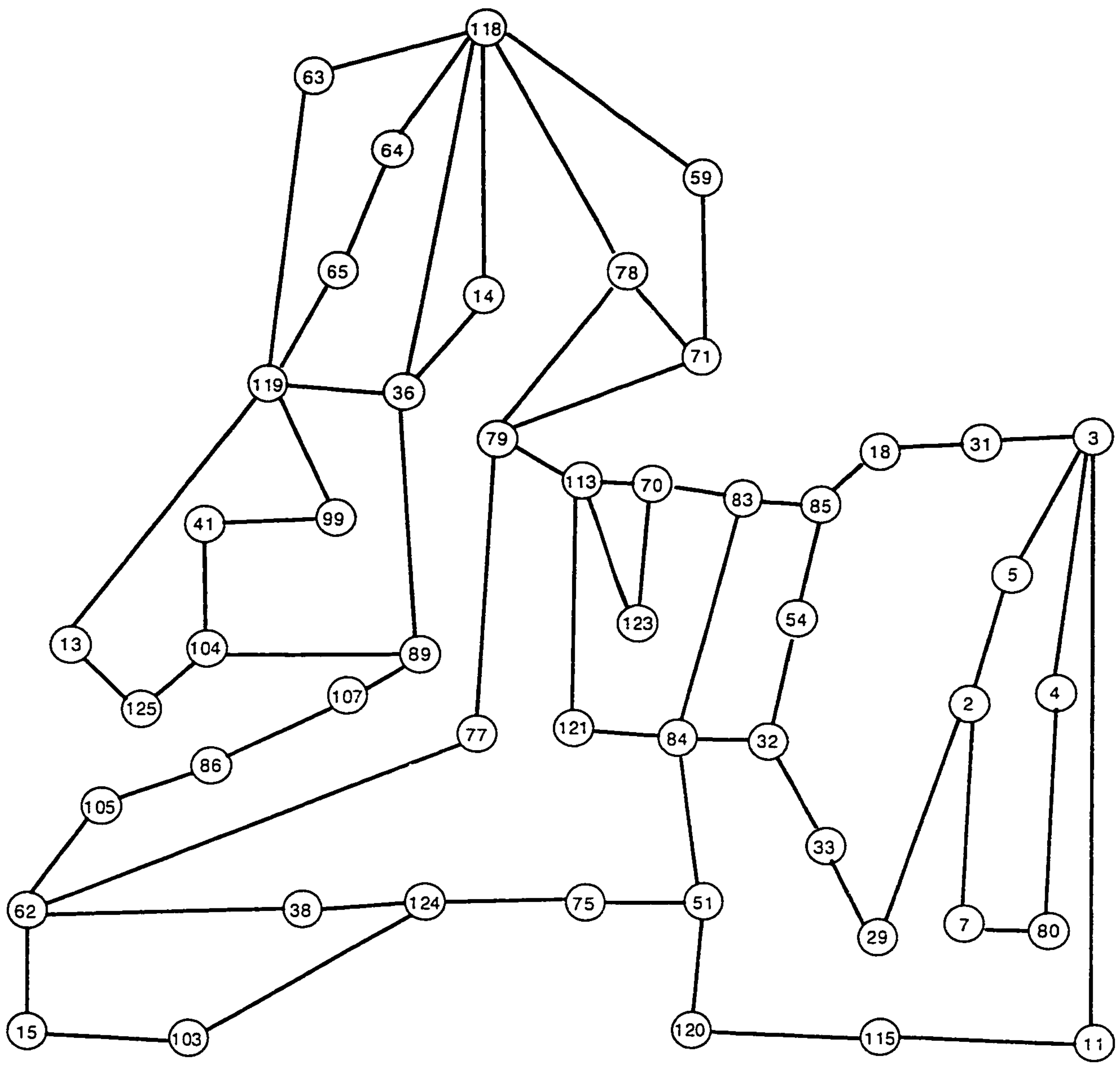


Figure 2.6. Topology of Net C.

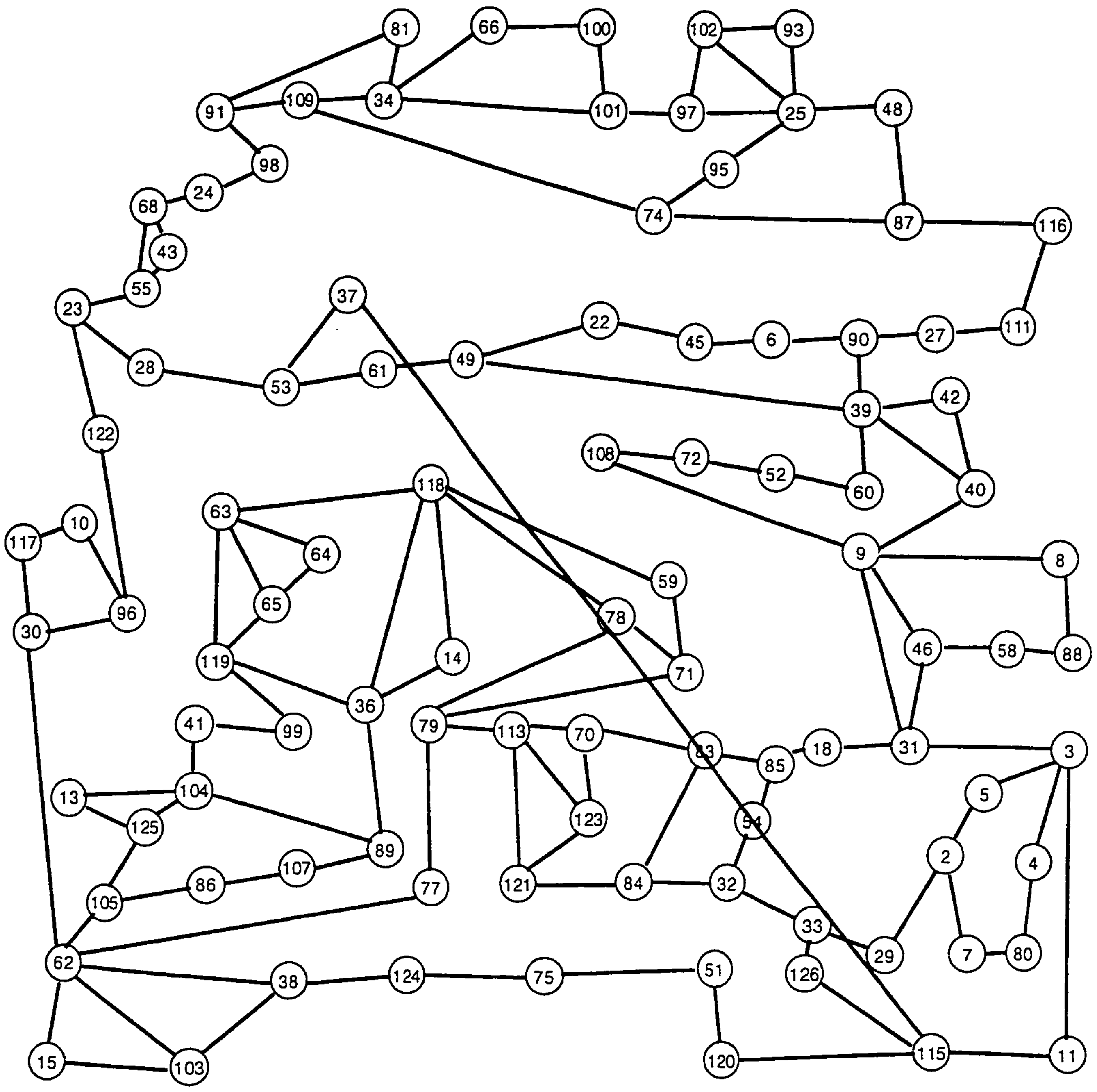


Figure 2.7. Topology of Net D.

3.0 Span Coverage Integer Program (SCIP)

This chapter introduces the Span Coverage Integer Program (SCIP) as an evaluation method for the proposed span elimination strategies. SCIP was first introduced by Kennington [27] and a variant is implemented by Morley [15]. A similar Integer Program (IP) is formulated here and is verified and validated. This verification test summary can be found in Appendix B. This chapter begins by detailing the concepts behind SCIP, and then its IP formulation will be laid out. Some experimental results are presented later in this chapter.

3.1 Concepts

SCIP can be thought of as an ideal capacitated span coverage problem formulation. It finds strictly the minimum cost set of rings required to cover all the non-zero span working capacities in the network. Referring to Figure 3.1, in this approach all the demands are already routed over the network topology. A separate process is executed to search for all possible network cycles (using the cycle generator). With the values set for all the SCIP design parameters (listed in Chapter 3.2.1), by taking the capacitated network topology, the set of cycles and the cost model as inputs, SCIP creates a list of the cost of each of the ring candidates for each different user-defined modular capacity size. The objective function is then to find the minimum cost set of ring candidates to cover the network's capacitated spans. Inherently, this formulation enables SCIP to perform single-modular or multi-modular ring network design depending on the number of user-defined modular capacity sizes.

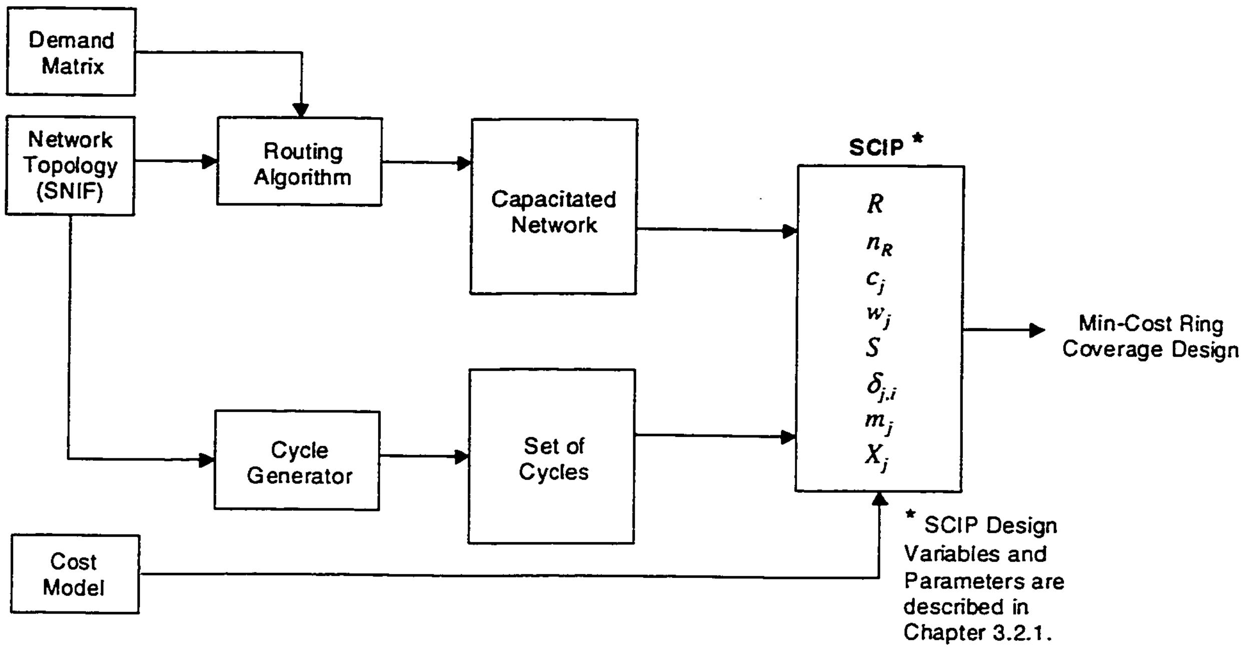


Figure 3.1. SCIP block diagram.

SCIP involves certain idealizations: (i) every node is implicitly assumed to have an ADM in each ring that transits the site; (ii) demands may transit between rings at any location; and, (iii) no demand is ever re-routed, nor is any span eliminated in the design. With these idealizations, SCIP is judged as a pure span coverage algorithm giving design results that strictly find the min-cost set of ring candidates to cover all capacitated spans without reflecting any of the practical factors of real ring systems, e.g. glass-through[*] nodes and inter-ring transition constraints. Such assumptions have rendered the formulation of SCIP relatively easy but this is at the expense of not attaining absolute cost minimization of a real ring network design. However, SCIP can be used as a surrogate for the full-blown ring design in purely comparative studies. Later, in Chapter 10, a second prediction using a full-blown ring coverage heuristic is described, namely TR*Labs*' RingBuilder v2.0, to determine how well this surrogate IP conforms to a real full-blown ring design. It will be seen that there is a strong statistical correlation between the results of this IP and those of RingBuilder, even though the accuracy of the IP has been compromised due to the idealizations.

Next, the SCIP formulation will be described in detail.

[*] In a practical ring system, there may exist glass-through nodes where the traffic crossing such nodes is *not* dropped but amplified to the next subsequent node.

3.2 SCIP Formulation

3.2.1 Notations

Sets:

R : set of ring candidates.

Design Parameters:

n_R : number of ring candidates in set R.

c_j : cost of ring candidate j.

w_j : working capacity of span j.

S : number of spans in the network.

$\delta_{j,i}$: 1 if ring candidate j intersects span i. 0 otherwise.

m_j : modular capacity of ring candidate j.

Variables:

X_j : the number of copies of ring candidate j placed on the corresponding cycle of the graph(integer).

3.2.2 IP Formulation

With a set of ring candidates R consisting of n_R ring candidates, the objective function is:

$$\textit{Minimize} : \quad \sum_{j=1}^{n_R} c_j \cdot X_j \tag{3.1}$$

subject to:

$$\sum_{j=1}^{n_R} m_j \cdot \delta_{i,j} \cdot X_j \geq w_i \qquad \forall i \in \{1,2,\ldots,S\} \tag{3.2}$$

$$X_j \geq 0, \text{ integer} \qquad \forall j \in \{1,2,\ldots,n_R\} \tag{3.3}$$

The objective function (3.1) is to minimize the total network equipment cost over all rings. The subset of rings that will result in minimum cost is determined by the integer variable X_j, which is decided by constraint set (3.2). This constraint restricts the modular span capacity of all rings that cross span i to be greater than or the same as the working capacity assigned to that span. This constraint inherently implies that every node has an ADM and that there are no constraints on the inter-ring transition locations. In addition, due to constraint set (3.2), SCIP is only suitable for shared-protection rings. This is because the capacity of a path-protection ring (in Uni-directional Path Switched Rings, UPSRs) is shared amongst all its spans and, therefore, this ring scheme cannot be modelled on a per-span basis. This restricts SCIP investigations to shared-protection rings (Bi-directional Line Switched Rings, BLSRs).

The set of ring candidates R may be determined by identifying all the possible rings in the topology, or by specifying a subset from all the possible rings. One can limit the number of ring candidates, n_R, by forcing a maximum cycle size in the cycle generator. The cost, c_j, for each ring candidate j includes the cost of fibre and ADM common equipment but excludes the cost of add-drop port cards. The value for c_j will be determined in section 3.2.4. All the experiments using SCIP are implemented in AMPL [28] and solved using CPLEX v6.0.2 [29]. The AMPL model file for this IP formulation is shown in Appendix C.

3.2.3 Complexity Analysis of SCIP

Consider the worst case scenario for SCIP where the given network graph is a *complete* graph (i.e. each pair of distinct vertices form an edge) of order N, where N is the number of vertices. In this case, the upper bound on the number of possible ring designs can be derived if the following assumptions are made:

- A ring design consists of a set of distinct rings only.
- Each ring has infinite capacity.
- Only one type of ring is considered.

Under these assumptions, a feasible design can be recognized as a set of ring candidates that covers every node in the network at least once. The upper bound for the number of ring candidates is derived. Each ring candidate is restricted to some combination of three or more active nodes up to a maximum of sixteen [1]. Here, the upper bound for the number of ring candidates is as follows [2]:

[1] In practice, the K1 and K2 bytes responsible for the Automated Protection Switching (APS) in the SONET ring networks support only up to 16 actively terminating nodes.

[2] This is a special case of the analysis obtained in [16,30].

$$Q = \sum_{i=3}^{16} \binom{N}{i} \tag{3.4}$$

If a design is defined as consisting of some combination of one or more ring candidates, based on the assumptions, it is realized that the maximum number of rings required to cover every node in the network is N-2. Consider a limiting case where every ring contains only three active nodes: to obtain a maximum coverage while still meeting the bi-connectivity requirement, every ring must have at least two nodes in common and one node disjoint from the other rings in the ring cover. In this case, the first ring covers three nodes and the remaining N-3 rings cover one additional node. In total there are N-2 rings. Based on this argument, the upper bound for the number of feasible designs is:

$$\sum_{j=1}^{N-2} \binom{Q}{j} \tag{3.5}$$

From this analysis, it is realized that the number of ring candidates grows $O(2^N)$ * with N, resulting in an even faster increase in the number of feasible designs. This is evident when the number of possible designs is plotted against the number of nodes, as shown in Figure 3.2 below.

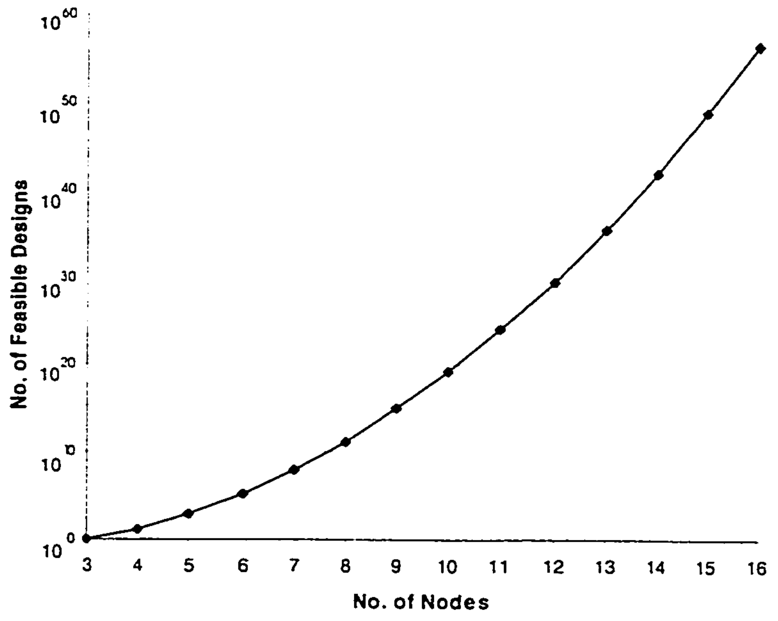


Figure 3.2. The upper bound for the number of feasible designs.

* since $\sum_{i=3}^{16} \binom{N}{i} < \sum_{i=3}^{N} \binom{N}{i} = 2^N - \frac{(N^2+N+2)}{2} < 2^N$, therefore the upper bound is $O(2^N)$.

It is apparent from Figure 3.2 that solving the multi-ring design problem of a large network using an algorithm with $O(2^N)$ can be onerous. Even for a network with N=16, there can potentially be 10^{55} designs (or constraints), which is clearly impractical to solve[1]. Therefore, when solving for large networks, the following restrictions have been adopted for practical interest:

- Enforcing a maximum cycle size, thereby reducing the number of ring candidates (Chapter 3.2.5).
- Restricting the SCIP computational time where the min-cost design prior to time exhaustion is held.

3.2.4 Network and Equipment Costs

As mentioned earlier, c_j is the combined cost of all the ADMs and the total fibre distances of ring candidate j. For accurate cost coverage design, c_j must be precise. Since these investigations deal with different span modular capacities, the ADM cost in c_j should also reflect some level of economy-of-scale to be realistic. As a result, a rule has been adopted: two times the cost for equipment that delivers four times the capacity. This rule applies only for the cost of ADMs. The cost of fibre remains fixed[2] for all different modular capacities. Keeping this in mind, the costs of ADMs and fibre are implemented as in Table 3.1.

Table 3.1. Network Equipment and Fiber cost

Span Modular Capacity	ADM Cost ($/ADM)	Fibre Cost ($/unit distance)
OC-12	71333	10
OC-48	142666	
OC-96	201730	
OC-192	285332	

[1] Consider even a single machine instruction that can solve a constraint executed at a speed of 1 ps, the total execution time for 10^{55} constraints is $10^{55}\cdot 1$ ps or 3.2×10^{35} years.

[2] The cost of fibre is fixed at $10/unit distance and directly refers to the single span module regardless of the fibre count in that span module.

3.2.5 Limiting the Maximum Cycle Size

The number of ring candidates can be limited by imposing a maximum cycle size. The main benefit of doing so is to reduce the number of ring candidates, which then decreases the total SCIP computational time (since there are fewer possible designs that need to be computed, see Equation 3.5 and Figure 3.2). The definition of computational time will be discussed in Chapter 3.3.3. The number of ring candidates must be limited so as to obtain, within a reasonable amount of time, a complete SCIP run. For example in Net B, if the maximum cycle size is set to sixteen, there are a total of 119,340 distinct cycles. There are, in this case, an astronomical number of possible designs, which would cause SCIP weeks or even months to complete. For research purposes, therefore, the maximum cycle size for Net B has been reduced to seven. The maximum cycle size for other networks is shown in Table 3.2 together with the resulting number of cycles.

Table 3.2. The maximum cycle size defined for each test network.

Network	Max. Cycle Size	No. of Cycles
A	16	307
B	7	330
C	16	63
D	18	295

Note that Net D has a maximum cycle size of eighteen. This is because the minimum cycle size required for Net D to attain full restoration is eighteen. An exception, therefore, has to be made if all nodes are assumed to include active ADMs as opposed to possible glass-through sites. This arises in Net D from the large void in its center. Consider, for instance, the shortest-path for replacement of span (111-116).

3.3 Experimental Results: Pure-Coverage Designs with SCIP

3.3.1 Single-Modularity Ring Coverage Design

Table 3.3 shows the SCIP results using a single modular capacity design. Each network has four different min-cost designs, each design using different ring modular capacities. The average network working capacity, $\overline{w}$, is the sum of all the span working capacity requirements divided by the total number of spans. $\overline{w}$ is later used to study the effects of ring modular capacities on SCIP design cost in single-modularity ring coverage design.

Table 3.3. Min-cost results obtained using shortest-path routing and SCIP single-modular design.

Network	Ave. Network Capacity	Design Modularity OC-	SCIP Design Cost (millions)
A	5.956	12	1.94 *
		48	3.29
		96	4.65
		192	6.57
B	78.508	12	34.72
		48	20.11
		96	17.84 *
		192	20.62
C	19.861	12	14.45
		48	12.48 *
		96	15.60
		192	22.04
D	29.515	12	46.36
		48	33.85
		96	33.70 *
		192	47.57

The demands for each test network in Table 3.3 are routed using pure shortest-path where $\overline{w}$ is later calculated for each resulting network. Then, for each of the selected design modularities (i.e. OC-12, OC-48, OC-96 and OC-192), a single-modular SCIP run is executed where the resulting SCIP design cost is tabulated in the *SCIP Design Cost* column. The least-cost SCIP design for each test network is indicated by (*) in the SCIP design cost column. The computational time taken by each SCIP run is shown later in Table 3.5.

3.3.2 Multi-Modularity Ring Coverage Design

Table 3.4 shows SCIP results using multi-modular design. There are four distinct modular sizes and costs used, shown in Table 3.1. Here, each network has a resulting min-cost coverage design where the most improved ring-set may consist of rings with different modular capacities. SCIP automatically selects the ring candidates, as well as the modular capacity for each of the ring candidates, that will lead to the minimization of the total network coverage cost. The *Ring Modularities* column shows the number of rings with their corresponding modular capacity used in the resulting min-cost design.

Table 3.4. Min-cost results for multi-modular SCIP (CPU runtime limited to 24 hours).

Network	SCIP Design Cost (millions)	Ring Modularities			
		12	48	96	192
A	1.94	6	-	-	-
B	16.73	-	3	8	3
C	11.08	4	7	1	-
D	26.85	11	9	6	-

3.3.3 Computational Time for SCIP Results

The computational time for each experiment in Tables 3.3 and 3.4 is shown in Table 3.5 below. The definition of computational time includes the CPU time for routing all the demands onto the network topology and the CPU time required by SCIP to determine the min-cost ring design. We have limited the CPU time for each individual SCIP experiment to 24 hours (86,400 seconds). Therefore, if an experiment has reached the 24-hour limit, the SCIP solution returned is the result prior to the point of time exhaustion. The incomplete experimental runs are indicated by (*) in the *Total Computational Time* column. Note that the computational time required by the incomplete experimental runs is not exactly 86,400 seconds due to the additional CPU time required to route all the PTP demands. All the experiments were run on a Sun UltraSparc HPC-450 with four processors, each running at 250 MHz.

Table 3.5. Computation times for SCIP.

Network		Ring Modular Capacity	Comp. Time (seconds)
A	SCIP (Single)	12	3.63
		48	2.27
		96	2.4
		192	2.78
	SCIP(Multi)	-	9.7
B	SCIP (Single)	12	86600.04 *
		48	19.89
		96	19.47
		192	1107.98
	SCIP(Multi)	-	86601.54 *
C	SCIP (Single)	12	0.12
		48	0.08
		96	0.34
		192	0.35
	SCIP(Multi)	-	39.91
D	SCIP (Single)	12	7.62
		48	12.53
		96	32.13
		192	901.51
	SCIP(Multi)	-	86496.02 *

* *indicates incomplete experimental runs. The feasible design after 24-hours is reported.*

As expected, Table 3.5 shows that obtaining min-cost designs for large networks with high average network degree (Net B and Net D) requires more computational time than it does for small networks (Net A) or networks with relatively low average network degree (Net C). Furthermore, it shows that SCIP(multi) requires significantly more computational time than SCIP(single). This is expected since SCIP(multi) has four times the number of ring candidates than its SCIP(single) counterpart.

3.4 Discussions and Summary

SCIP is simple and repeatable. Its IP formulation is clear and straightforward, leading to SCIP results being easily reproduced and verified. There are several recognized idealizations in SCIP. This IP assumes that every node has an ADM and that demands may transit between rings at any location. This leads to several drawbacks, for example, SCIP is not suitable for situations where the absolute design cost is desired, it does not take glass-throughs into consideration, and it is limited to investigations using shared-protection rings. However, the main interest is the use of SCIP as a quick surrogate for RingBuilder in relative cost comparison contexts. Experiments are described later, in Chapter 10, which investigate the correlation between SCIP and RingBuilder results.

In this chapter, SCIP has provided a set of reference designs that are at minimum cost for the property of full coverage, i.e., before spans are eliminated. These reference designs are used later to compare the designs obtained from the proposed span elimination strategies, thereby defining their performance.

Referring to Figure 3.3, the results obtained from single-modular designs suggest that there is a relationship between the selected ring modular capacity, m, and the average network working capacity, $\overline{w}$. It appears that the min-cost results occur only when the selected ring modular capacity is larger than the average network working capacity ($m / \overline{w} > 1$). This makes sense because when the selected ring modular capacity is smaller than the average network working capacity ($m / \overline{w} < 1$), overlapping rings occur more often since it is more likely that a single span module with the selected modular capacity is not adequate to cover the entire span's working capacity. However, selecting a larger modular capacity may result in a highly redundant network, hence, there is a best-fit modular capacity between these modular capacities where min-cost coverage design is obtained. This relationship is evident in Figure 3.3.

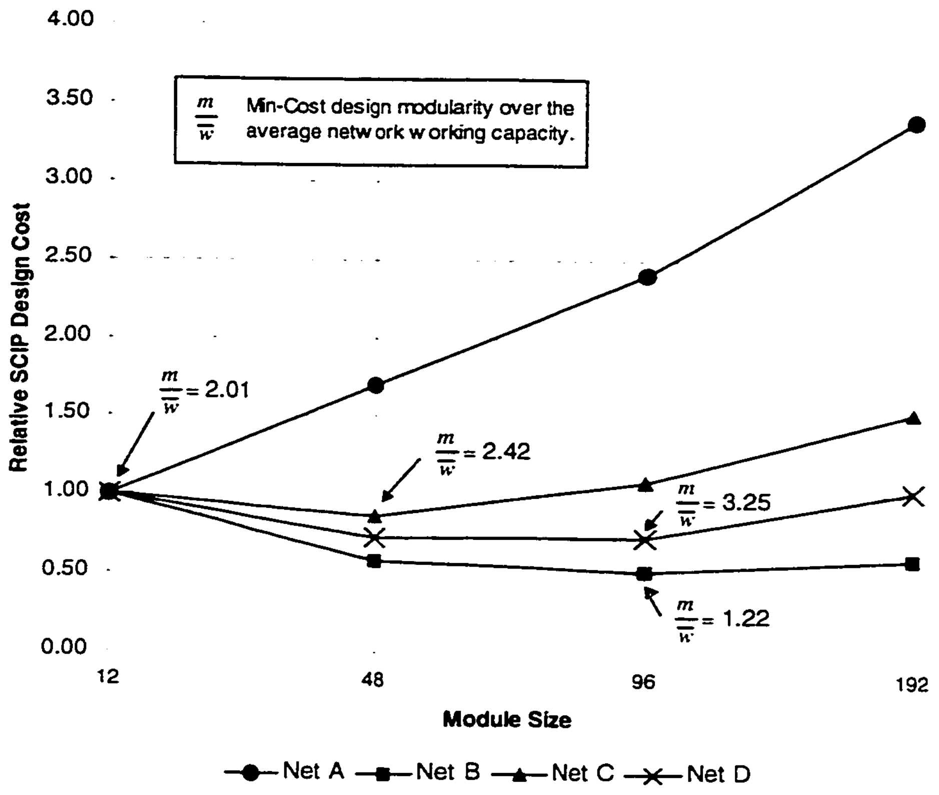


Figure 3.3. **Effects of modular capacity on single modular SCIP design cost (Nets A-D, pure shortest-path).**

As expected, the cost results obtained from the multi-modular capacity design always give a min-cost design that is either equal to or better than the single-modular capacity design (Table 3.3 and 3.4). In practice, it is not cost-effective to use a single-modular ring capacity to cover the entire network topology. The multi-modular capacity design is implemented since it gives better min-cost coverage results and flexibility in integrating different modular ring capacities.

To conclude, in this chapter, SCIP has been formulated and benchmark results have been produced to be used later for comparative assessment of the proposed span elimination heuristics.

4.0 Modular Aggregating Pre-Routing (MAPR)

MAPR is a pre-processing span elimination heuristic which is meant to precede a multi-ring coverage design. In this chapter, MAPR is described for single-modular capacity ring coverage design.

4.1 Introduction

As described earlier in Chapter 2, shortest-path routing generally does not result in an evenly distributed load on spans in the network (in the w_i sense). This leads to inefficient use of the span capacity in subsequently placed rings driven by the strict coverage imperative. Moreover, the quantities of demand that accumulate on spans under shortest-path routing have no particular regard for or relationship to the capacity modularity of the rings that may later be used in coverage design. For example, consider the network topology and the current span working capacity requirements shown in Figure 4.1a, and assume that there are still five unrouted demands from source node 2 to destination node 6. By placing one unit of demand at a time over the network topology using pure shortest-path, these five demands are routed over the network using span (2-3) and span (3-6) (denoted by path (2-3)-(3-6)) as they have the shortest distance from node 2 to node 6. As a result, span (2-3) would then have a total of five working units and the subsequent ring design would be forced to use two rings with OC-12 modular capacity to cover span (2-3). Together, however, they have only 5/24th capacity utilization on that span. It is fairly apparent by inspection that a better overall ring design could have been obtained by preventing demands from being accumulated on span (2-3).

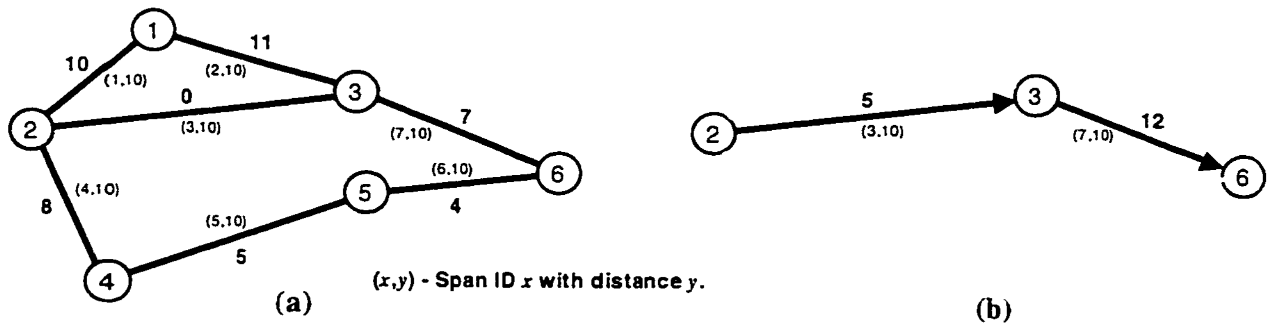


Figure 4.1. **Example of MAPR concept: (a) Original network topology and spans' capacity requirements. (b) Remaining five demands from node 2 to node 6 routed using shortest-path.**

The motivating idea behind MAPR is to *route demands in a manner that tends to aggregate flows on spans, leaving totals that are well-suited for later min-cost ring coverage designs.* Specifically, the idea is that once a span has some demand on it, it is important to make it look more attractive (cheaper) to subsequent least-cost routing determinations, but to cancel that effect as soon as a modularity boundary arises. Consider the initial network topology and span capacity requirements as shown in Figure 4.1a, and the five remaining unrouted demands from node 2 to node 6. There are three distinct paths from node 2 to node 6, i.e., paths (2-1)-(1-3)-(3-6), (2-3)-(3-6) and (2-4)-(4-5)-(5-6). Since span (2-3) is unused, the cost of routing using path (2-3)-(3-6) will be the most expensive. The five demands are then routed, one unit at a time, using paths (2-1)-(1-3)-(3-6) and (2-4)-(4-5)-(5-6). After one unit of demand is routed on path (2-1)-(1-3)-(3-6), it is found that the working capacity requirements on span (1-3) have reached modular capacity (using OC-12 span module). Using path (2-4)-(4-5)-(5-6) is now more attractive. The remaining four demands are completely routed using path (2-4)-(4-5)-(5-6). These steps are detailed in Figure 4.2 below.

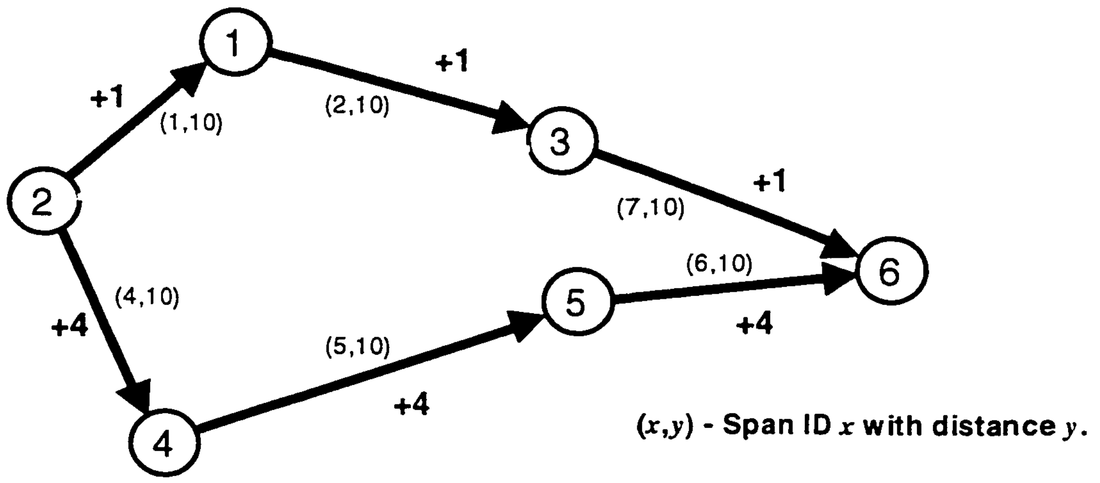


Figure 4.2. Example of demands routed with OC-12 oriented flow aggregation.

The effects of this routing also create candidates for span elimination when all the demand is drawn away by aggregation pressure onto routes over other "well-used" spans. Any non-essential span having zero working capacity on it after MAPR may be considered eliminated. For example, span (2-3) in Figure 4.2 is a non-essential span (refer to Chapter 2.3.1). A single run of MAPR in general network investigations will possibly result in a set of several simultaneous span eliminations*. After the non-essential spans are eliminated, the min-cost coverage design is obtained using SCIP implemented with single-modular capacity design. Figure 4.3 below shows the result of subsequent SCIP executed on the resulting network in Figure 4.2 with span (2-3) eliminated.

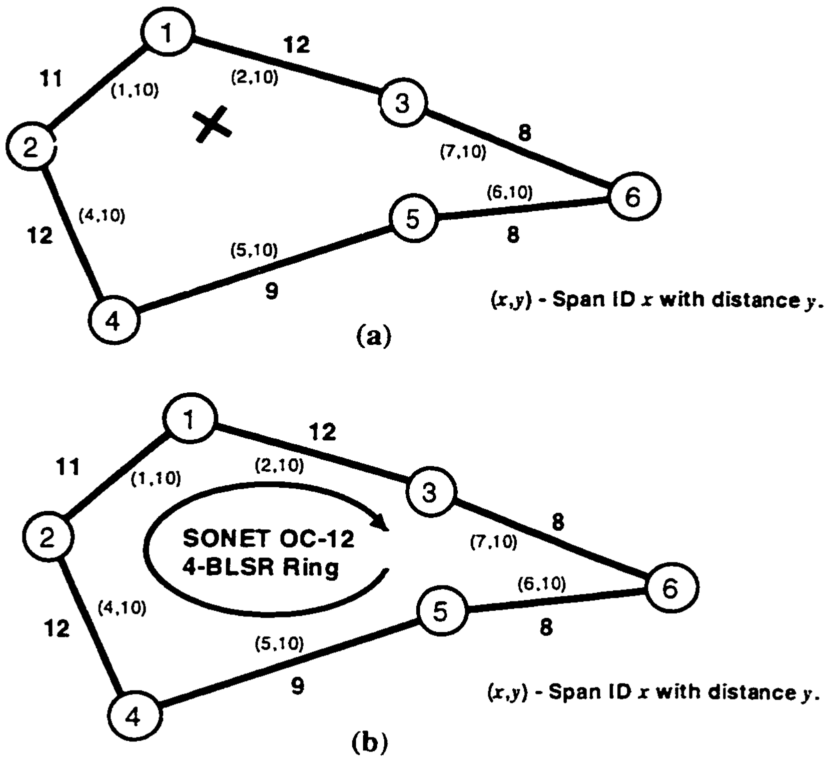


Figure 4.3. **Example of eliminating a non-essential span resulting from MAPR. (a) Span 3 is eliminated by OC-12 aggregating routing. (b) OC-12 SCIP result.**

4.2 Detailed Explanation of MAPR

This chapter describes the methodology used to implement MAPR. MAPR applies the principle of aggregating routing described earlier in the chapter, which is enabled by introducing pseudo-cost functions. First, the steps that are involved in implementing MAPR are described and then several pseudo-cost functions that are used by MAPR are specified.

* Not shown in the examples in Figures 4.1 and 4.2. The full procedure of MAPR begins with routing the first unit demand in the demand matrix until all demands are satisfied. MAPR ends when all the resulting non-essential spans with zero working capacity are eliminated.

4.2.1 Flow Chart

A schematic depiction of MAPR is shown in Figure 4.4.

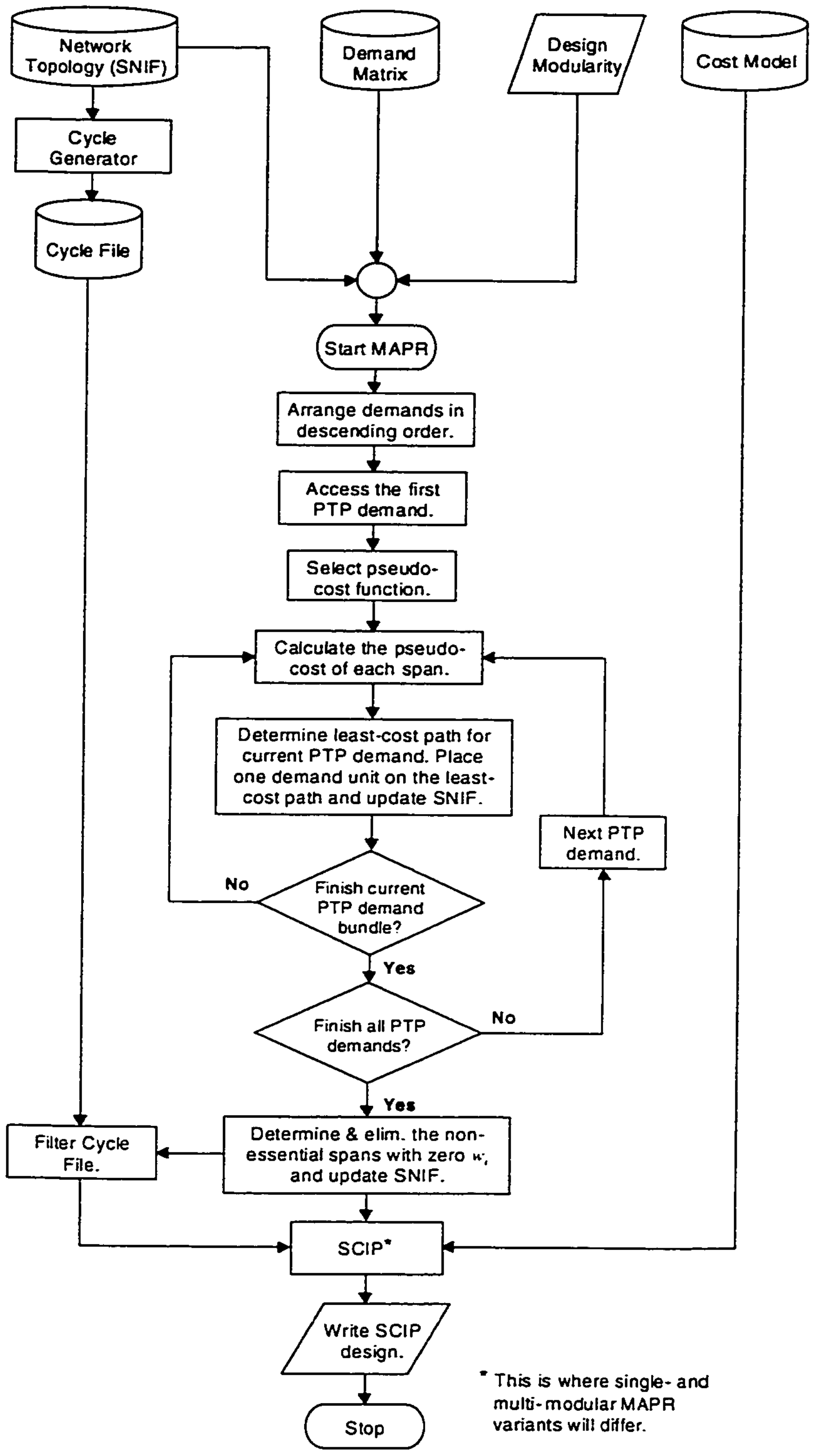

Network Topology (SNIF)
Demand Matrix
Design Modularity
Cost Model
Cycle Generator
Cycle File
Start MAPR
Arrange demands in descending order.
Access the first PTP demand.
Select pseudo-cost function.
Calculate the pseudo-cost of each span.
Determine least-cost path for current PTP demand. Place one demand unit on the least-cost path and update SNIF.
Next PTP demand.
No
Finish current PTP demand bundle?
Yes
Finish all PTP demands?
No
Yes
Filter Cycle File.
Determine & elim. the non-essential spans with zero w, and update SNIF.
SCIP*
Write SCIP design.
Stop
* This is where single- and multi-modular MAPR variants will differ.

Figure 4.4. **Block diagram: Implementing MAPR.**

The design inputs (for single-modular design) contain four components: network SNIF file, demand matrix, design modularity, and cost model. Initially, a separate pre-processing is required where the cycle generator (Chapter 2.3.3) is used to find all the network cycles. These cycles are defined in the cycle file.

MAPR begins with three inputs: the network SNIF file, the demand matrix and the design modularity. First, the PTP demand elements from the demand matrix are sorted in descending order of size, starting with the largest PTP demand. The pseudo-cost function (described in the next chapter) to be used is then selected, calculated (using the design modularity), and assigned for every span in the network. A single unit demand is routed at each iteration over the currently apparent 'least-cost' path from source to destination. If there are equal least-cost paths, the unit of demand is routed over an arbitrary choice amongst them. Each time a demand is routed, some span working capacity requirements change, the pseudo-cost of these spans is recalculated, and another unit demand iteration follows. The aggregation or attracting and clumping together of demands results from the capacity-dependent span pseudo-cost model. After all the PTP demands have been routed using the steps described above, any non-essential spans with zero working capacity (zero w_i) are eliminated from the network description file.

In the ring network design phase, the cycles that use the eliminated non-essential spans are removed from the cycle file. Using the resulting sub-graph and the filtered cycle file, SCIP is executed and the min-cost ring coverage design is then obtained.

4.2.2 Modular Pseudo-cost function in MAPR (Single-modular case)

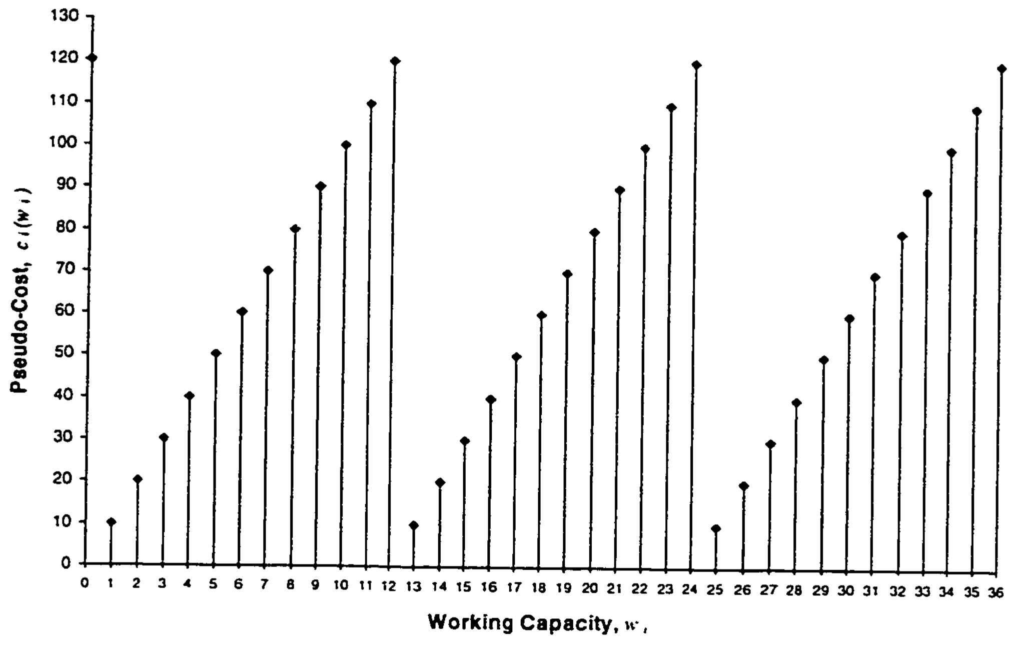
The pseudo-cost assigned to each span is determined by its geographical distance and its current capacity requirement (number of demands so far accumulated on the span). Two pseudo-cost functions have been tested; namely *rise* (Eq. 4.1) and *impulse* (Eq. 4.2). (Another pseudo-cost function, *fall*, was also tested but initial experiments did not show any reasonable improvement in design cost, and it was, therefore, removed from further work. Refer to Appendix D for further details).

$$c_i(w_i) = \begin{cases} d_i \cdot M, & (w_i \bmod M) = 0; \\ d_i \cdot (w_i \bmod M), & (w_i \bmod M) \neq 0; \end{cases} \tag{4.1}$$

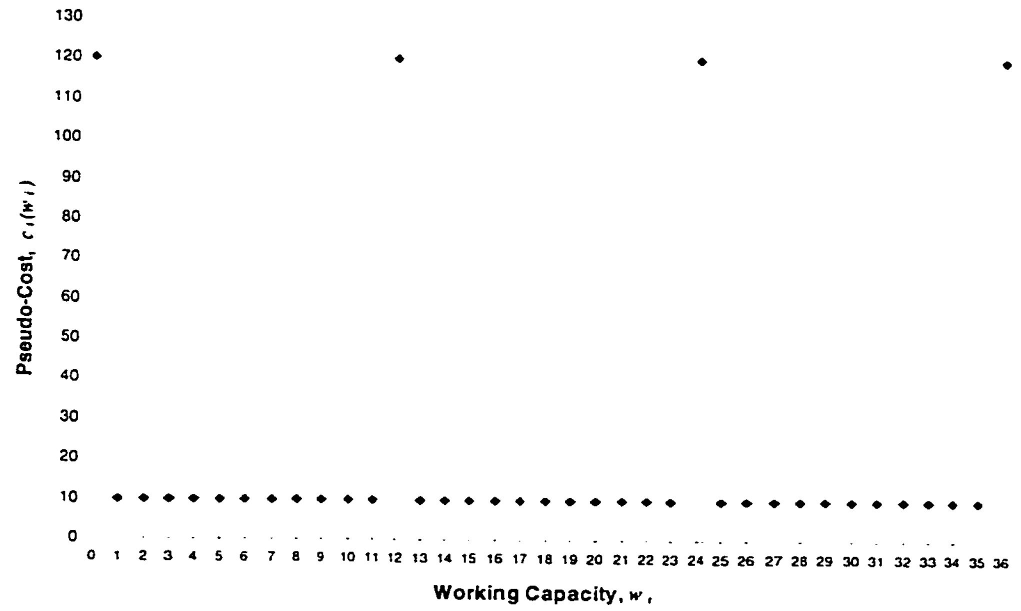
$$c_i(w_i) = \begin{cases} d_i \cdot M, & (w_i \bmod M) = 0; \\ d_i, & (w_i \bmod M) \neq 0; \end{cases} \tag{4.2}$$

$c_i(w_i)$	- Pseudo-cost of span *i*.
d_i	- Geographical distance of span *i*.
w_i	- Accumulated demands crossing span *i*.
M	- Span modularity constant.

The *rise* pseudo-cost function assigns a maximum routing cost to a span when it has a zero capacity accumulation or when it reaches a capacity accumulation that is a multiple of the modularity constant M. For example, a subset of the full family SONET rings that are strictly defined (by standards) is characterized by M=12 for rings with OC-12 modular capacity, M=48 for rings with OC-48 modular capacity, M=96 for rings with OC-96 modular capacity and M=192 for rings with OC-192 modular capacity. The span pseudo-cost is at its highest when it has accumulated enough capacity that it might be perfectly accommodated in a ring subsequently chosen in the coverage design process. In contrast, as soon as the span has one more demand unit crossing it than one of the module capacities, its cost is set at its lowest value. The logic is that in this state an extra module is implied anyway by any subsequent coverage design process, so it makes sense to make this span more attractive (lower routing cost) in the next iteration. For w_i values between these extremes, under the *rise* pseudo-cost function, the pseudo-cost rises linearly with w_i steadily reducing the aggregation pressure on this span as it rises towards a next full-module total. Under the *impulse* pseudo-cost function, the span has a minimum constant routing cost as long as it has any slack at all vis-à-vis the next modular value at exact 'full module' values of w_i. Figure 4.5 below shows the example of $c_i(w_i)$ vs. w_i on span i with d_i=10 and M=12 using *rise* and *impulse*.



(a)



(b)

Figure 4.5. Periodic MAPR pseudo-cost function for d_i=10 and M=12. (a) *rise*. (b) *impulse*.

4.3 Summary

MAPR was implemented and investigated as a pre-processor tool which is meant to precede the multi-ring coverage design. MAPR routes demand using the least-cost paths determined by calculating a w_i-dependant pseudo-cost of each span. As such, demands are routed more efficiently with proficient piling of working capacities onto each span, leaving totals that are well-suited for later ring coverage design. Subsequently, non-essential spans that have zero working capacities in the network are eliminated, saving network equipment costs.

Chapter 6 investigates MAPR further to define its performance during several experiments. Specifically, the solution quality that this heuristic presents and the required computational time are of interest. A comparative study using these results against the results from designs without span eliminations (presented in Chapter 3) will be discussed as well as the results from another span elimination heuristic, namely, Iterated Routing and Elimination (IRE), at the single-modularity design perspective.

5.0 Iterated Routing and Elimination (IRE): Single-Modularity

5.1 Introduction

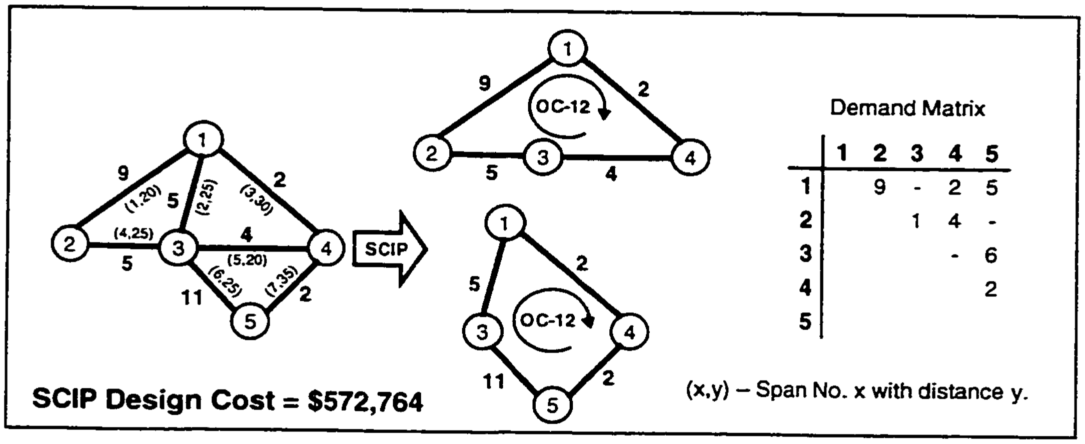
Unlike MAPR, IRE is a post-processing span elimination heuristic. IRE uses a greedy approach to ultimately eliminate each and every non-essential span from the network. Following each elimination, the affected demands are detoured over alternate paths. Since there are many different combinations of non-essential spans that can be eliminated at each stage of IRE, a breadth-first-search (BFS) strategy is used to systematically eliminate spans.

To illustrate IRE, consider the example shown in Figure 5.1. This uses OC-12 design modularity and the cost model described in Table 3.1 (ADM cost = $17,333/ADM and $10/unit distance fibre cost). In the first iteration, the PTP demands are routed over the network graph using pure shortest-path with the resulting network and span capacity requirements shown in Figure 5.1a. An SCIP run is executed thereafter which gives a design cost of $572,764. Then, the network non-essential spans are determined which are, in this case, span (1-3) and span (3-4).

On the next iteration, the sub-graphs resulting from the single elimination of each non-essential span are examined. First, span (1-3) is eliminated. Since there are five demand units affected by this elimination, these affected demands must again be rerouted using other paths. Because the re-routing of demands is done at the path-level (refer to Figure 2.2), the working capacities of other spans that were used to transport each of the five affected demands must also be removed. Since the five affected demands in span (1-3) came from the PTP demand (1-5) using demand path (1-3)-(3-5), five units of working capacity are removed from span (3-5) as well, thereby reducing the working capacity requirements on span (3-5) from eleven to six. Now, the five demand units must again be rerouted over the resulting sub-graph using MAPR(rise)-like routing principles where all five demands are routed using path (1-4)-(4-5). An SCIP run is executed next on the resulting sub-graph, which gives a design cost of $501,081 (Figure 5.1b). Next, returning to the network graph before span (1-3) was eliminated, span (3-4) is eliminated and, using the procedures described earlier, the four affected demands are re-routed where three demands are routed using path (2-1)-(1-4) and one demand is routed using path (2-3)-(3-5)-(5-4)*. Executing SCIP on the

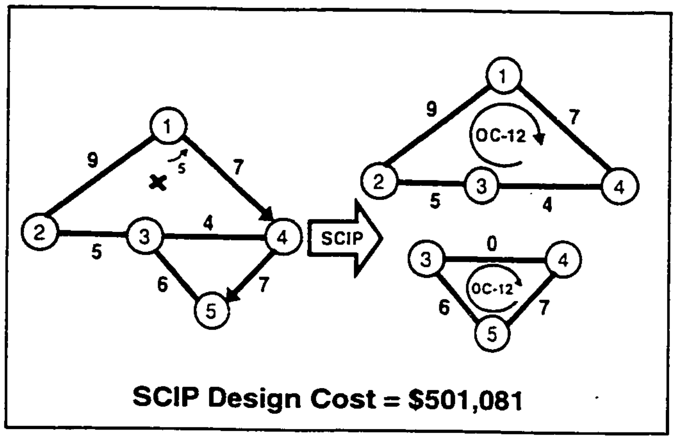
* With OC-12 design modularity and MAPR(rise)-like routing principles, routing using path (2-1)-(1-4) appears to be more expensive since the working capacity accumulated on span (2-1) has reached a single full module after three units of demand are routed. Therefore, the remaining one unit of demand is routed using the less expensive path (2-3)-(3-5)-(5-4).

subsequent sub-graph results in a design cost of $501,181 (Figure 5.1c). Here, of the two SCIP designs, the sub-graph that resulted from the elimination of span (1-3) appears to give the least-cost design. Therefore, this sub-graph is permanently adopted and the work proceeds to the next iteration.

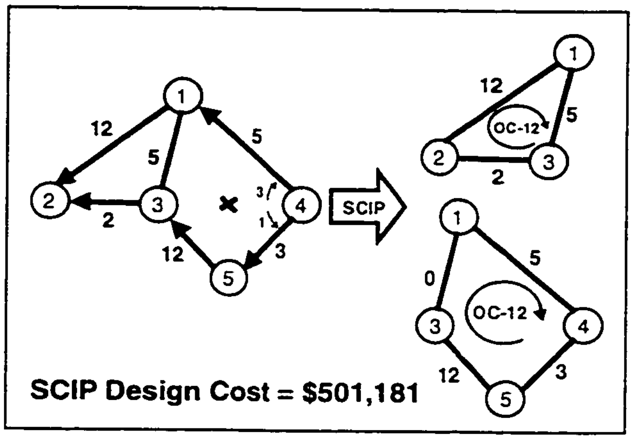

SCIP
OC-12
SCIP Design Cost = $572,764
Demand Matrix
(x,y) – Span No. x with distance y.

	1	2	3	4	5
1		9	-	2	5
2			1	4	-
3				-	6
4					2
5					

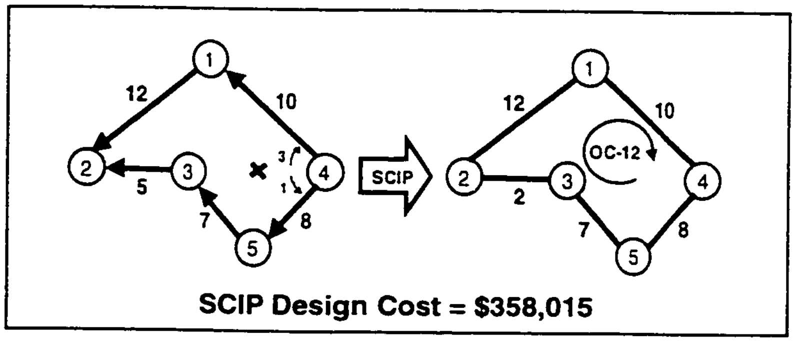
(a) Original network with demand routed using pure shortest-path.



(b) 1st iteration test removal of span (1-3) from original network.



(c) 1st iteration test removal span (3-4) from original network.

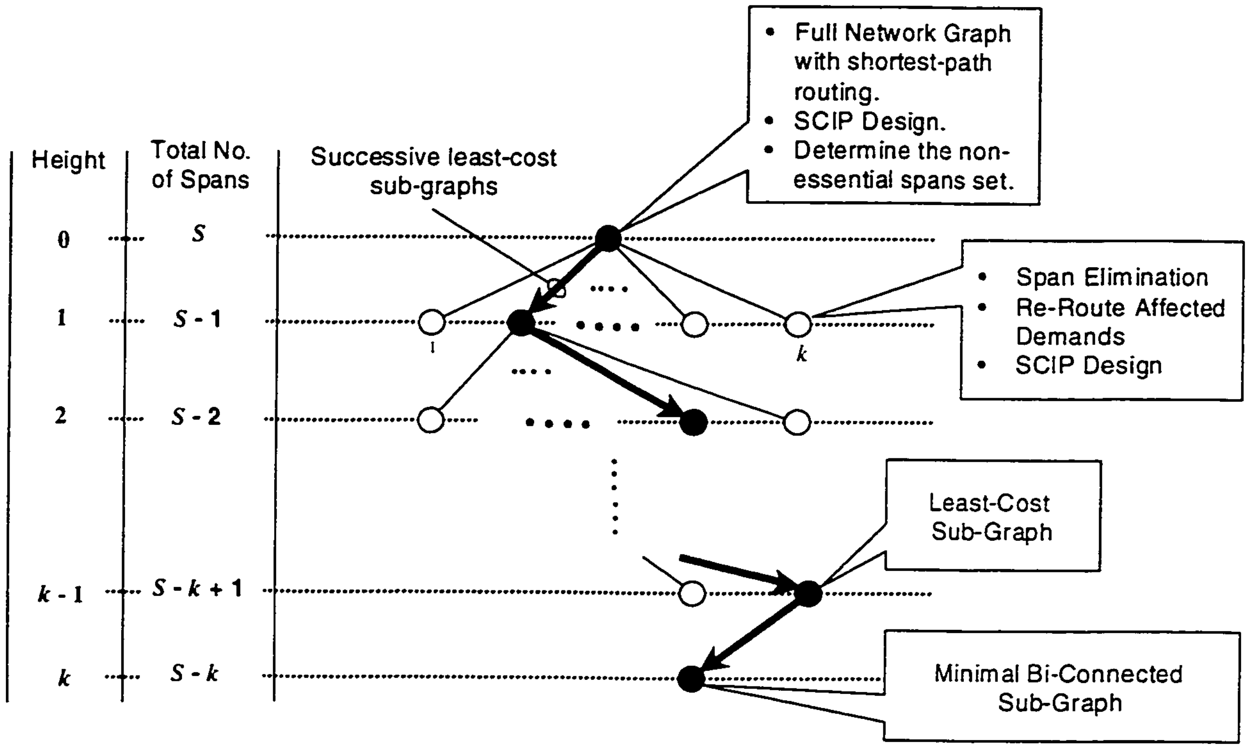


(d) 2nd iteration test removal of span (3-4) from least-cost network after 1st iteration.

Figure 5.1. An example illustrating the concepts of IRE.

Span (3-4) is the only non-essential span left in this iteration. By eliminating span (3-4), three demands are re-routed using path (2-1)-(1-4) and the remaining demand is re-routed using path (2-3)-(3-5)-(5-4). A subsequent SCIP run on the resulting sub-graph gave a design cost of \$358,015 (Figure 5.1d). Since there are only essential spans left in the network graph, the IRE process stops here where the most-improved multi-ring coverage design found from using IRE is shown in Figure 5.1d with a design cost of \$358,015.

The objectives and operations of IRE can also be represented by using a tree to illustrate the BFS aspect. This starts with a capacitated ring coverage design found by using pure shortest-path routing and SCIP which will be the root of the tree as shown in Figure 5.2.



Height
Total No. of Spans
Successive least-cost sub-graphs
Full Network Graph with shortest-path routing.
SCIP Design.
Determine the non-essential spans set.
0
S
1
S - 1
1
k
2
S - 2
k - 1
S - k + 1
k
S - k
Span Elimination
Re-Route Affected Demands
SCIP Design
Least-Cost Sub-Graph
Minimal Bi-Connected Sub-Graph

Figure 5.2. IRE Breadth-First Search (BFS) Tree.

All the non-essential spans in the full network topology are identified. The number of non-essential spans is represented by the parameter *k*. During the first iteration when the depth of the tree is equal to one, every single elimination of non-essential spans of the root graph is examined followed by the MAPR(rise)-like re-routing of the affected demands. SCIP is executed thereafter to evaluate the ring design cost of the subsequent sub-graphs. Span elimination, re-routing of affected demands, and SCIP design are represented by the vertices in the tree (except for the root vertex) shown in Figure 5.2. Since there are *k* non-essential spans, the first level depth search will have *k* vertices. After all the *k* span eliminations have been assessed for SCIP design cost at this depth, the elimination that led to the least-cost SCIP design is kept. The solid dark vertex, as shown in Figure 5.2, represents the least-cost sub-graph found. The set of non-essential spans is then updated by removing the non-essential span that had resulted in the least-cost sub-graph, whence the set has *k*-1 remaining non-essential spans.

The next iteration is then considered, where the root of this iteration is the least-cost sub-graph found earlier in the first iteration. Here, each element in the set of non-essential spans is tested, the overall least-cost sub-graph is selected, the next iteration with *k*-2 number of non-essential spans is tested, and so on until the leaf node of the BFS tree is reached, at which time all the non-essential spans have been eliminated. The overall result is a greedy sequence of span eliminations forming a trajectory through the space of all possible span elimination sequences. The leaf node at the end of this sequence is a minimal bi-connected sub-graph where further span elimination would leave the sub-graph non-restorable.

The maximum depth of the tree corresponds to the total number of non-essential spans, i.e., if there are *k* non-essential spans, the maximum depth of the tree will also be *k*. Since, at each iteration, there will be one less non-essential span to be examined, the total number of SCIP runs can be calculated using equation (5.1) below:

$$\text{Total SCIP runs} = k + (k-1) + (k-2) + \ldots + 1 = k^2 - \frac{k(k-1)}{2} = \frac{k(k+1)}{2} \qquad \textbf{(5.1)}$$

where, *k* is the number of non-essential spans.

Here, if *k*=20, the total number of SCIP runs in IRE will be 210. A flow chart for implementing the IRE strategy is described in the next section.

5.2 Implementing IRE

A schematic depiction of IRE is shown in Figure 5.3 below.

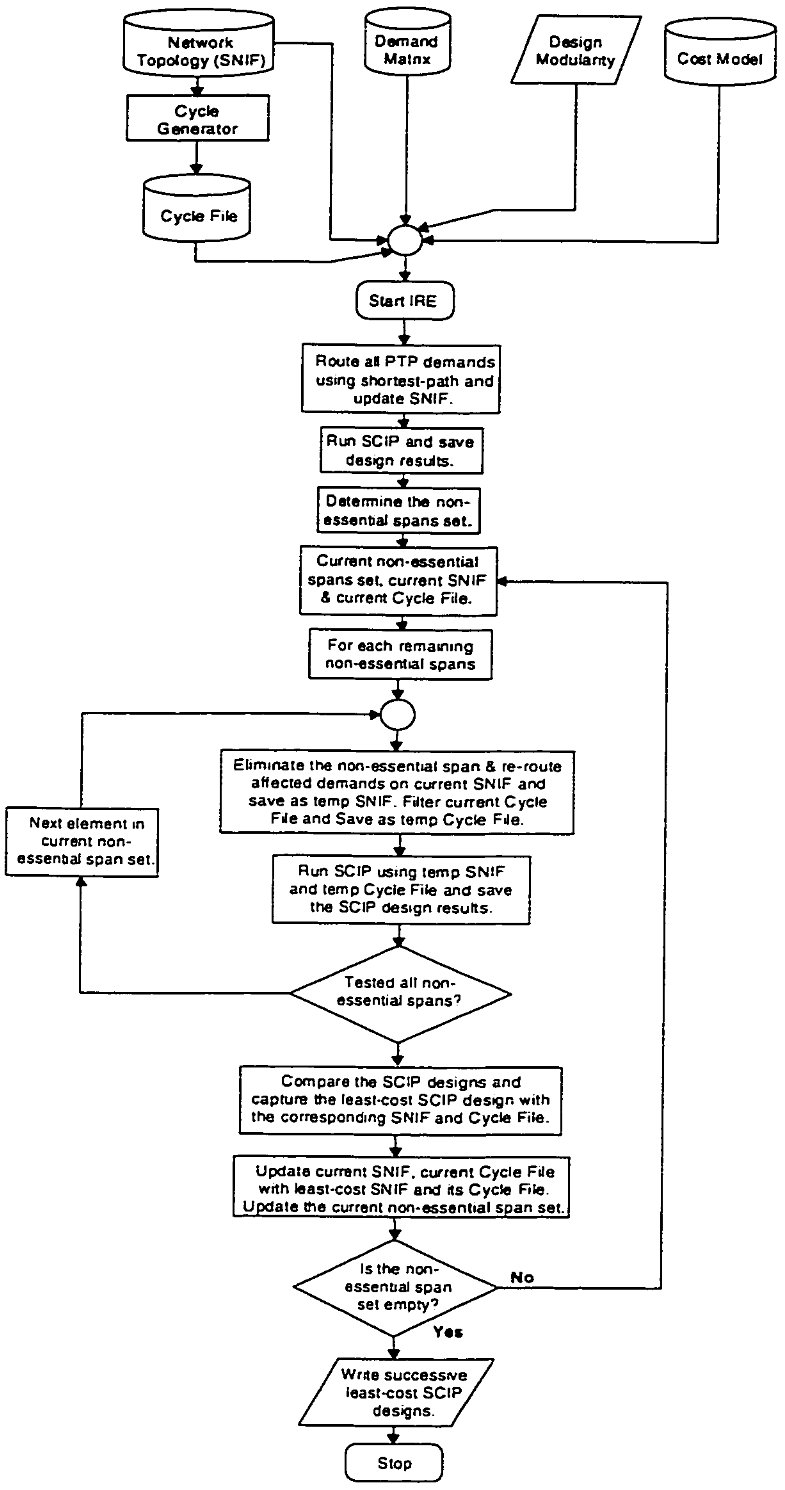

Network Topology (SNIF)
Demand Matrix
Design Modularity
Cost Model
Cycle Generator
Cycle File
Start IRE
Route all PTP demands using shortest-path and update SNIF.
Run SCIP and save design results.
Determine the non-essential spans set.
Current non-essential spans set, current SNIF & current Cycle File.
For each remaining non-essential spans
Eliminate the non-essential span & re-route affected demands on current SNIF and save as temp SNIF. Filter current Cycle File and Save as temp Cycle File.
Next element in current non-essential span set.
Run SCIP using temp SNIF and temp Cycle File and save the SCIP design results.
Tested all non-essential spans?
Compare the SCIP designs and capture the least-cost SCIP design with the corresponding SNIF and Cycle File.
Update current SNIF, current Cycle File with least-cost SNIF and its Cycle File. Update the current non-essential span set.
Is the non-essential span set empty?
No
Yes
Write successive least-cost SCIP designs.
Stop

Figure 5.3. Block Diagram: Implementing IRE.

IRE (single-modular design) begins with five inputs: network SNIF, demand matrix, design modularity, cost model, and cycle file (generated by the cycle generator detailed in Chapter 2.3.3). First, the PTP demands are routed over the network topology using pure shortest-path, and an SCIP design is obtained for the subsequent capacitated network topology. The set of non-essential spans is determined by finding a minimal bi-connected sub-graph and any additional spans that are essential for the hop-limited cycle coverage (Chapter 2.3.1).

Then, using the current set of non-essential spans, current capacitated network SNIF, and current cycle file, the IRE iterations begin. Each non-essential span is tentatively eliminated from the SNIF file, the affected demands are re-routed at the path-level using an MAPR(rise)-like routing principles, and the cycle set is filtered to eliminate cycles using that span. Finally, SCIP is executed on the resulting sub-graph to obtain the min-cost ring coverage design. This is repeated with subsequent non-essential spans until all non-essential spans have been examined, in terms of their impact on SCIP design cost, for prospective single-span elimination. After all non-essential spans have been examined, the elimination that leads to the lowest SCIP cost is permanently adopted.

This is repeated until all non-essential spans have been permanently eliminated. A graph of successive least cost SCIP designs can then be plotted against the number of span eliminations. From this, the most-improved least cost SCIP design amongst other least cost SCIP designs and the corresponding number of span eliminations can be determined.

5.3 Summary

In this chapter, IRE has been investigated as a post-processor span elimination heuristic and its implementation has been illustrated. The full schematic detail of the methods that were used to implement IRE is shown in Figures 5.2 and 5.3. From the flow chart, the IRE program can be viewed as making function "calls" to the SCIP program. Figure 5.4 below shows the relationship between the IRE program and SCIP.

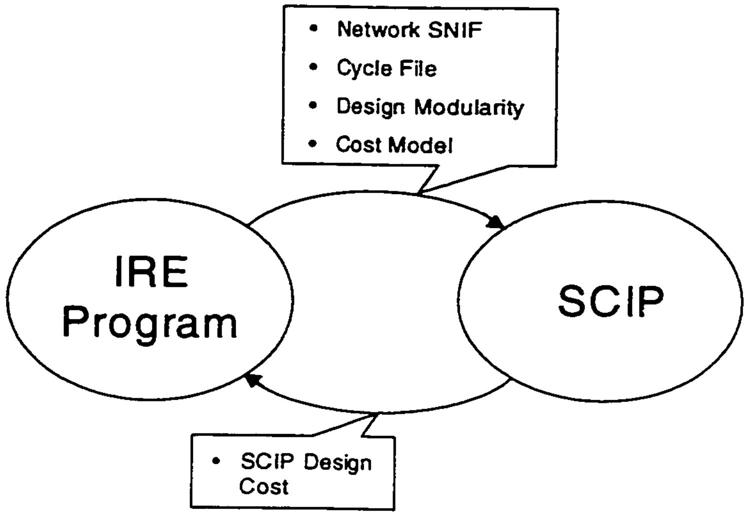


Figure 5.4. Communications between IRE program and SCIP

Basically, the IRE program can be viewed as 'remodeling' the demand patterns in the network graph to accommodate span eliminations. Each time a span is eliminated, a new SNIF is obtained (as a result of detouring the affected demands which, in turn, defines the new capacity requirements of some network spans) with its corresponding new cycle file (the cycles that use the eliminated span are removed). The design modularity and the cost model remain the same with respect to the current IRE investigation. Using the new SNIF and its cycle file, design modularity and cost model, the IRE program calls SCIP and returns the design cost of the new SNIF. Using this result to choose the span elimination that give the overall least cost SCIP design amongst the other SCIP designs at this iteration, IRE continues to the next iteration where it eliminates yet another span, and so on.

It is important to realize that the number of SCIP runs in IRE rises quadratically with the number of non-essential spans (Equation 5.1). This means that a network with a high average network degree with a large number of non-essential spans will require a high number of SCIP runs. This factor increases the computational time for IRE, as will be seen in the next chapter.

6.0 Initial Experimental Results and Discussions

6.1 Results and Analysis of MAPR and IRE

6.1.1 Non-Essential Spans (Candidates for Elimination)

Table 6.1 shows the non-essential span sets for each test network presented in Chapter 2.4. The method used to identify these spans was illustrated in Chapter 2.3.1. These sets of non-essential spans are used in the IRE investigations (a specific MBG has to be chosen prior to IRE to determine which set of non-essential spans to use). A graphical view of the chosen minimal bi-connected sub-graph and non-essential spans is shown in Figures 6.1 to 6.4 for Nets A, B, C and D, respectively.

Table 6.1. Test networks' non-essential spans (for IRE investigations).

Network	A	B	C	D
Max. Cycle Size	16	7	16	18
No. of Non-Essential Spans, k	11	18	7	21
Non-Essential Spans	(1-3) (1-4) (1-5) (1-8) (4-8) (5-6) (5-8) (5-9) (5-11) (8-9) (9-11)	(1-3) (3-4) (7-3) (6-9) (5-11) (12-10) (10-8) (10-13) (13-15) (9-15) (9-17) (8-17) (17-19) (19-18) (22-25) (9-18) (23-29) (21-23)	(32-84) (36-118) (36-119) (70-113) (71-79) (78-118) (83-84)	(9-31) (9-46) (25-97) (25-102) (30-96) (34-101) (34-109) (36-118) (36-119) (38-62) (39-40) (62-103) (63-65) (63-119) (70-113) (71-79) (78-118) (83-85) (89-104) (113-123) (104-125)

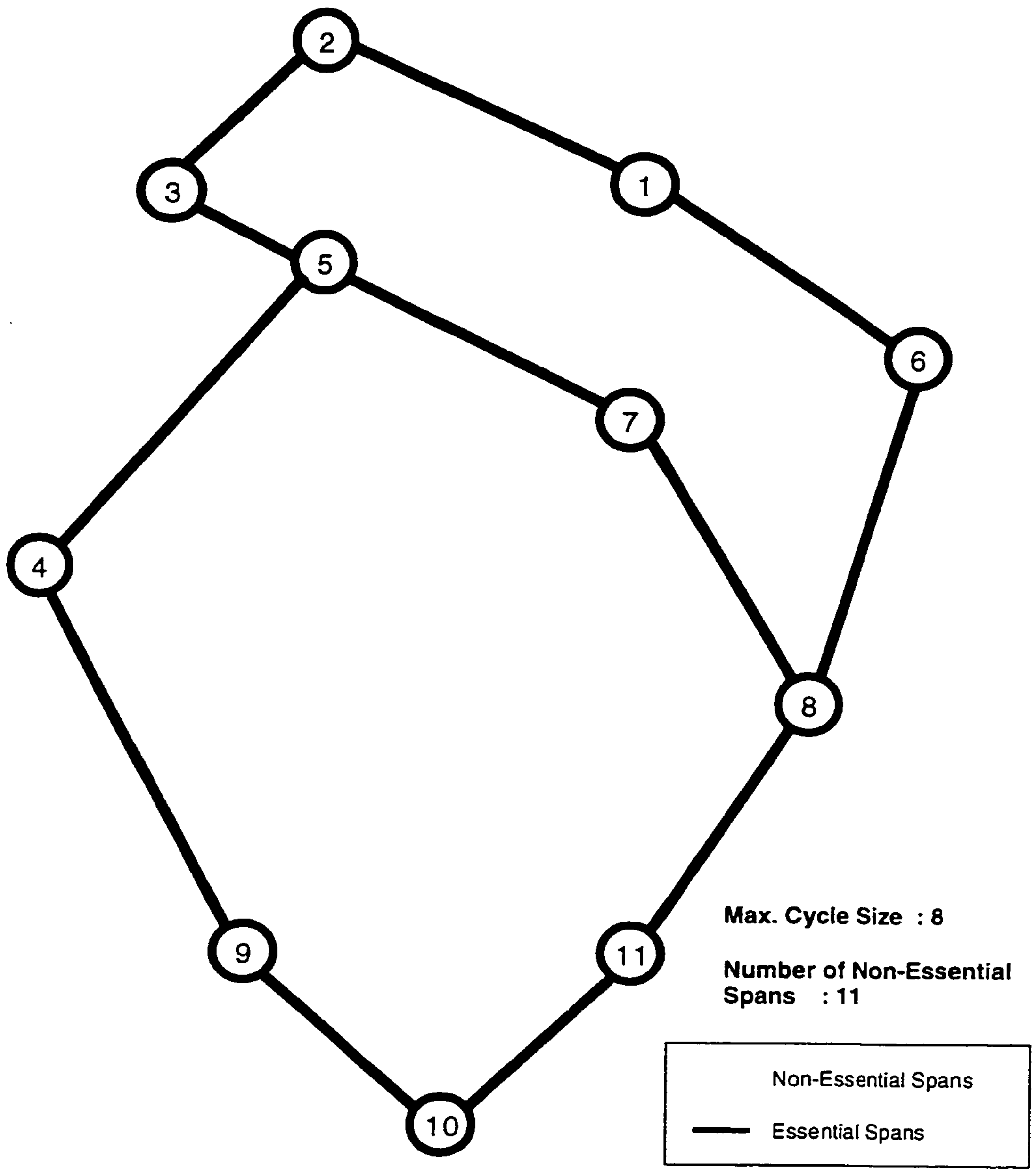


Figure 6.1. Net A: Minimal bi-connected sub-graph and non-essential spans.

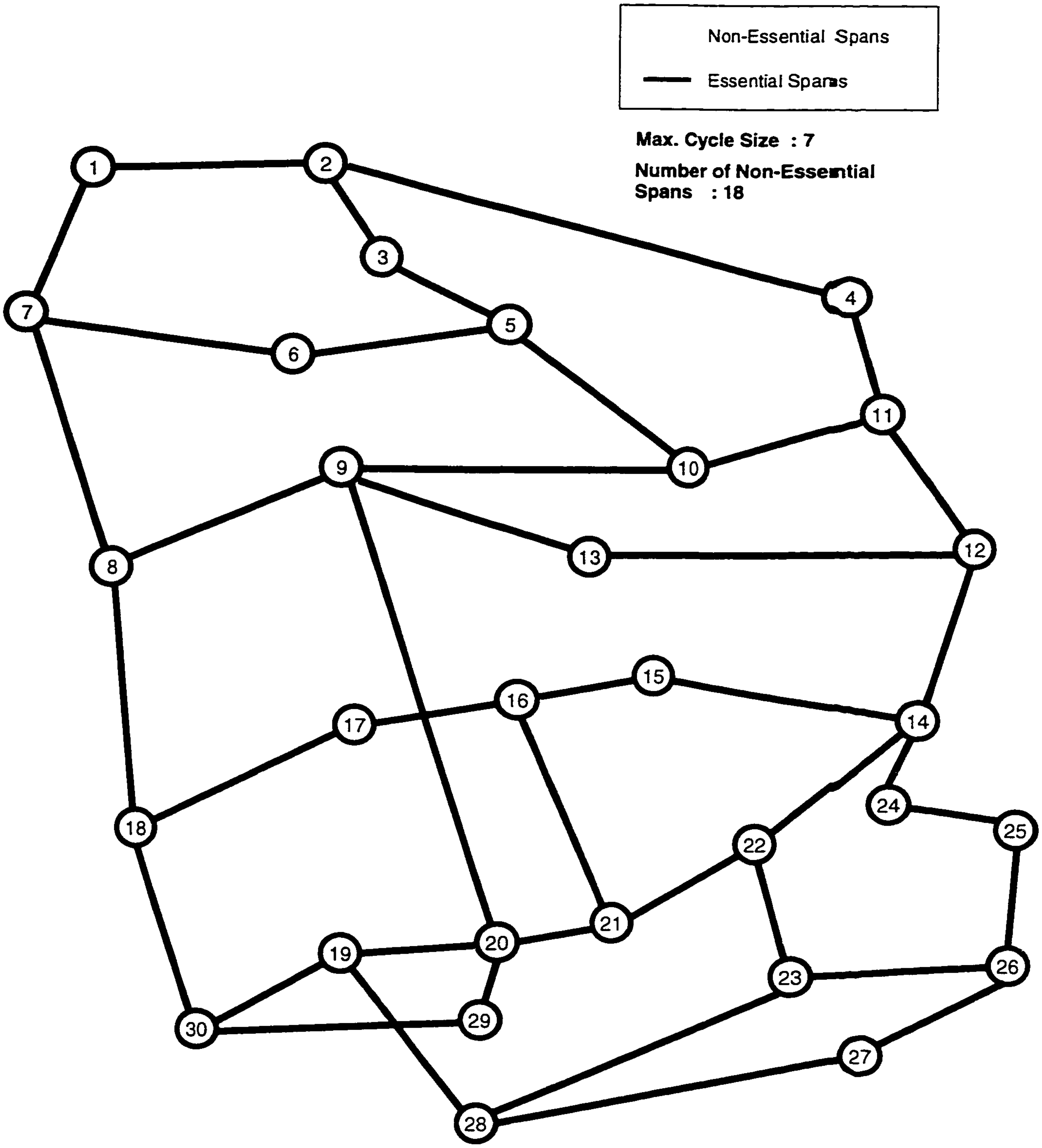


Figure 6.2. Net B: Minimal bi-connected sub-graph and non-essential spans.

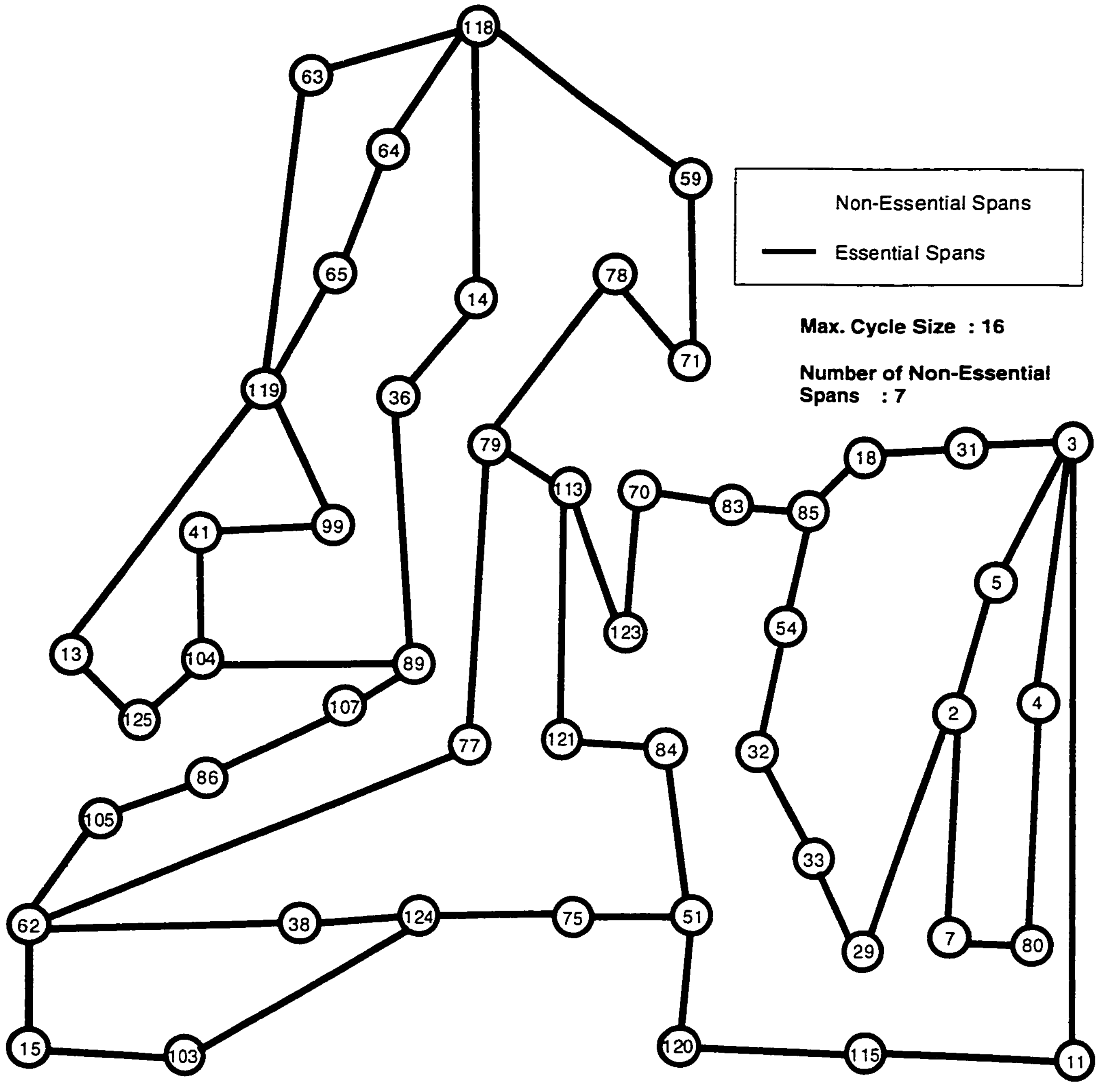


Figure 6.3. Net C: Minimal bi-connected sub-graph and non-essential spans.

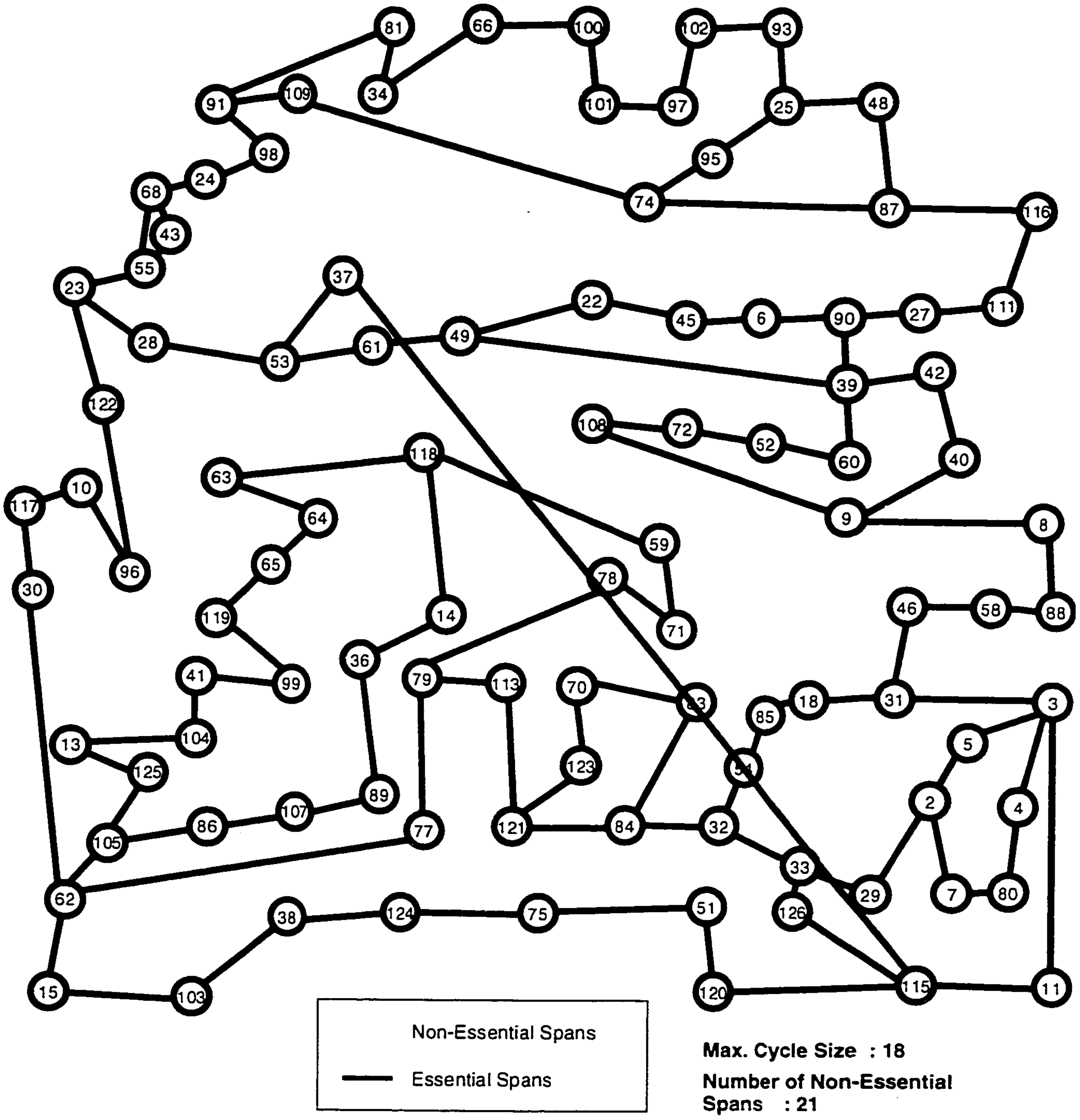


Figure 6.4. Net D: Minimal bi-connected sub-graph and non-essential spans.

Note that the non-essential spans shown in Table 6.1 may differ from the spans eliminated by MAPR. For example, assume that after routing all demands in MAPR on Net B, the resulting network has zero working capacity on span (2-4) but has working capacities on span (3-4). In this situation, MAPR will eliminate span (2-4) since it has zero working capacity, but eliminating this span will not leave the network one-connected; span (3-4) will become essential because (i) it has working capacity and (ii) it is restricted by the bi-connectivity requirements of node 4 (each vertex must have at least two connecting edges). This differs from the non-essential spans set shown for Net B in Table 6.1 where span (2-4) is an essential span and span (3-4) is a non-essential span. Therefore, determining and eliminating the non-essential spans in MAPR which have zero working capacity may vary from Table 6.1, and may require an additional check to ensure that the sub-graph after elimination is still bi-connected.

As described earlier in this section, only a single set of non-essential spans is investigated for each respective test network. Later for future improvements, Chapter 11 shows a test case and elaborates the impact of using a different set of non-essential spans on IRE (The test case is performed using IRE on Net B).

6.1.2 MAPR

Tables 6.2 to 6.5 show the results obtained from SCIP for Net A, Net B, Net C and Net D, respectively, following rise and impulse pseudo-cost rules in MAPR. The *Design Modularity* column represents the permitted modular capacity. The *#Elim* (Number of Span Eliminations) column represents the number of spans that were removed from the corresponding network. The *Elim. Span* (Eliminated Span) column shows the spans that were eliminated from the corresponding network by MAPR's aggregating routing. The *SCIP Design Cost* column shows the min-cost multi-ring coverage design obtained by SCIP following the MAPR run. The least cost result for each network is shown by the (*) in the *SCIP Cost* column. The *Comp. Time* (Computational Time) represents the total required CPU time, as described in Chapter 3.3.3, and the *Relative SCIP Design Cost* is the SCIP design cost normalized by the SCIP results for the shortest-path routing case (i.e. no eliminations) using the *smallest* design modularity (refer to Table 3.3 for SCIP design cost for each test network using OC-12).

Tables 6.2 to 6.5 indicate that, as the permitted design modularity increases, there exists a point at which the best-fit design modularity occurs where a minimum SCIP design cost is obtained. Increasing the permitted design modularity increases the number of span eliminations. When the minimum SCIP design cost obtained for rise is compared with the minimum SCIP design cost obtained for impulse, rise gives a better min-cost design than impulse. In addition, impulse requires a relatively larger permitted design modularity to obtain the min-cost SCIP design with respect to rise. As for the computational time, when the computational times obtained for both MAPR(rise) and MAPR(impulse) are compared with respect to the computational times obtained for shortest-path and no eliminations (Table 3.5), it can be seen that the MAPR algorithm generally has a faster computational time. These observations will be discussed and summarized later in this section.

Table 6.2. MAPR results for Net A.

MAPR Pseudo-Cost	Design Modularity	#Elim	Elim. Span	SCIP Design Cost (Millions)	Comp. Time (seconds)	Relative SCIP Design Cost
rise	12	6	(1-4),(4-8),(5-8),(5-11),(8-9),(9-11)	1.94 *	0.13	1.00
	48	9	(1-3),(1-4),(1-8),(5-6),(5-8),(5-9),(5-11),(8-9),(9-11)	2.29	0.08	1.18
	96	10	(1-3),(1-4),(1,5),(1-8),(5-6),(5-8),(5-9),(5-11),(8-9),(9-11)	2.83	0.1	1.46
	192	10	(1-3),(1-4),(1,5),(1-8),(5-6),(5-8),(5-9),(5-11),(8-9),(9-11)	4.00	0.08	2.07
impulse	12	9	(1-2),(1-3),(1-6),(4-9),(5-7),(5-8),(5-9),(7-8),(9-10)	2.22	0.01	1.15
	48	10	(1-2),(1-3),(1-4),(1-6),(4-9),(5-7),(5-8),(5-9),(7-8),(9-10)	2.00 *	0.01	1.03
	96	10	(1-2),(1-3),(1-4),(1-6),(4-9),(5-7),(5-8),(5-9),(7-8),(9-10)	2.83	0.01	1.46
	192	10	(1-2),(1-3),(1-4),(1-6),(4-9),(5-7),(5-8),(5-9),(7-8),(9-10)	4.00	0.01	2.07

Table 6.3. MAPR results for Net B.

MAPR Pseudo-Cost	Design Modularity	#Elim	Elim. Span	SCIP Design Cost (Millions)	Comp. Time (seconds)	Relative SCIP Design Cost
rise	12	0	-	34.65	4825.37	1.00
	48	0	-	19.54	0.27	0.56
	96	3	(17-18),(19-20),(9-8)	17.03 *	3.69	0.49
	192	6	(12-10),(9-17),(17-18),(19-20),(28-23),(21-23)	19.47	74.89	0.56
impulse	12	0	-	38.05	5779.99	1.10
	48	3	(13-15),(9-17),(8-18)	24.85	1.02	0.72
	96	7	(2-4),(10-13),(13-15),(16-21),(19-20),(25-26),(21-23)	21.89	5.26	0.63
	192	13	(1-3),(3-4),(12-10),(10-13),(9-15),(9-17),(17-18),(16-21),(19-18),(9-8),(23-29),(25-26),(21-23)	19.47 *	75.83	0.56

Table 6.4. MAPR results for Net C.

MAPR Pseudo-Cost	Design Modularity	#Elim	Elim. Span	SCIP Cost (Millions)	Comp. Time (seconds)	Relative SCIP Design Cost
rise	12	2	(32,84),(36-118)	15.69	0.15	1.09
	48	2	(32,84),(36-118)	12.78 *	0.18	0.88
	96	4	(32,84),(36-118),(36-89),(71-79)	14.79	0.20	1.02
	192	4	(32,84),(36-118),(36-89),(71-79)	20.89	0.24	1.45
impulse	12	0	-	18.21	0.61	1.26
	48	4	(32,84),(36-118),(36-89),(71-79)	13.93 *	0.01	0.96
	96	4	(32,84),(36-118),(36-89),(71-79)	14.79	0.03	1.02
	192	4	(32,84),(36-118),(36-89),(71-79)	20.89	0.02	1.45

Table 6.5. MAPR results for Net D.

MAPR Pseudo-Cost	Design Modularity	#Elim	Elim. Span.	SCIP Cost (Millions)	Comp. Time (seconds)	Relative SCIP Design Cost
rise	12	6	(9-31),(25-97),(32-84),(36-118),(38-62),(121-123)	53.76	32.85	1.16
	48	7	(9-31),(25-97),(32-84),(36-118),(38-62),(78-79),(121-123)	38.02	0.33	0.82
	96	7	(9-31),(25-97),(32-84),(36-118),(38-62),(36-89),(121-123)	33.49 *	0.87	0.72
	192	8	(9-31),(25-97),(32-84),(36-118),(38-62),(36-89),(71-79),(121-123)	46.71	0.27	1.01
impulse	12	8	(9-31),(25-97),(32-84),(36-89),(36-118),(38-62),(63-119),(121-123)	66.43	1.71	1.43
	48	9	(9-31),(25-97),(32-84),(36-89),(36-118),(38-62),(63-119),(71-79),(121-123)	48.82	0.05	1.05
	96	9	(9-31),(25-97),(32-84),(36-89),(36-118),(38-62),(63-119),(71-79),(121-123)	47.13	0.13	1.02
	192	9	(9-31),(25-97),(32-84),(36-89),(36-118),(38-62),(63-119),(71-79),(121-123)	46.71 *	0.36	1.01

Using the results gathered from Table 6.1 through Table 6.4, the relationship between SCIP design cost and design modularity can be shown. Figure 6.5 and Figure 6.6 below show the effects of design modularity on SCIP design cost with span eliminations by MAPR using rise and impulse for each test network. The *x*-axis is the design modularity. The *y*-axis (Relative SCIP Design Cost) is the SCIP design cost obtained from MAPR normalized by the SCIP results for shortest-path routing case (i.e., no eliminations) with the smallest permitted design modularity.

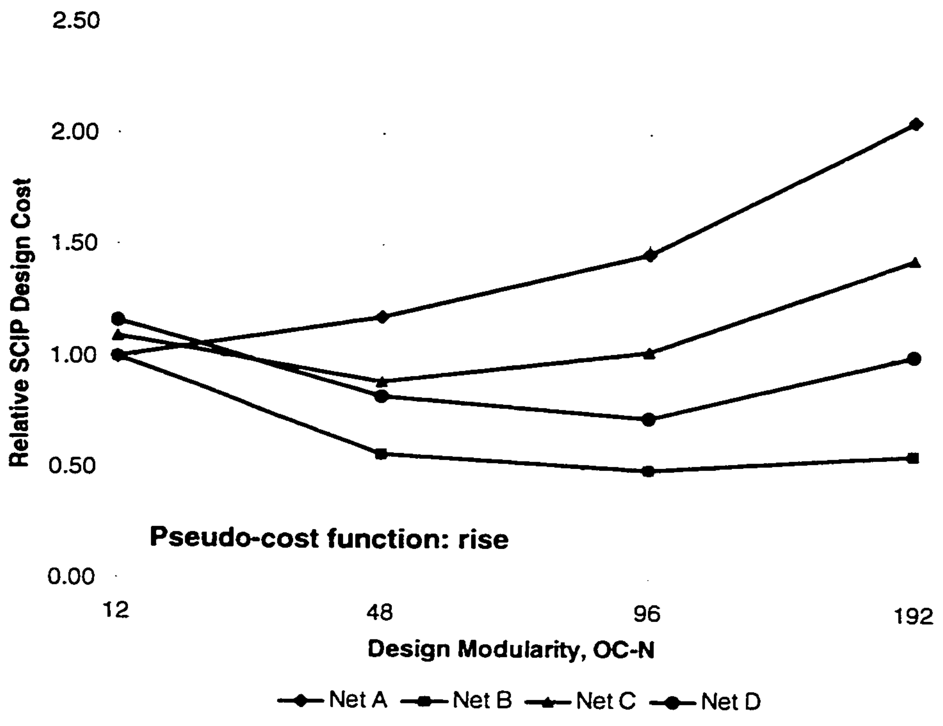


Figure 6.5. **Effects of modular capacity on design cost with span eliminations by MAPR (Nets A-D, rise pseudo-cost function).**

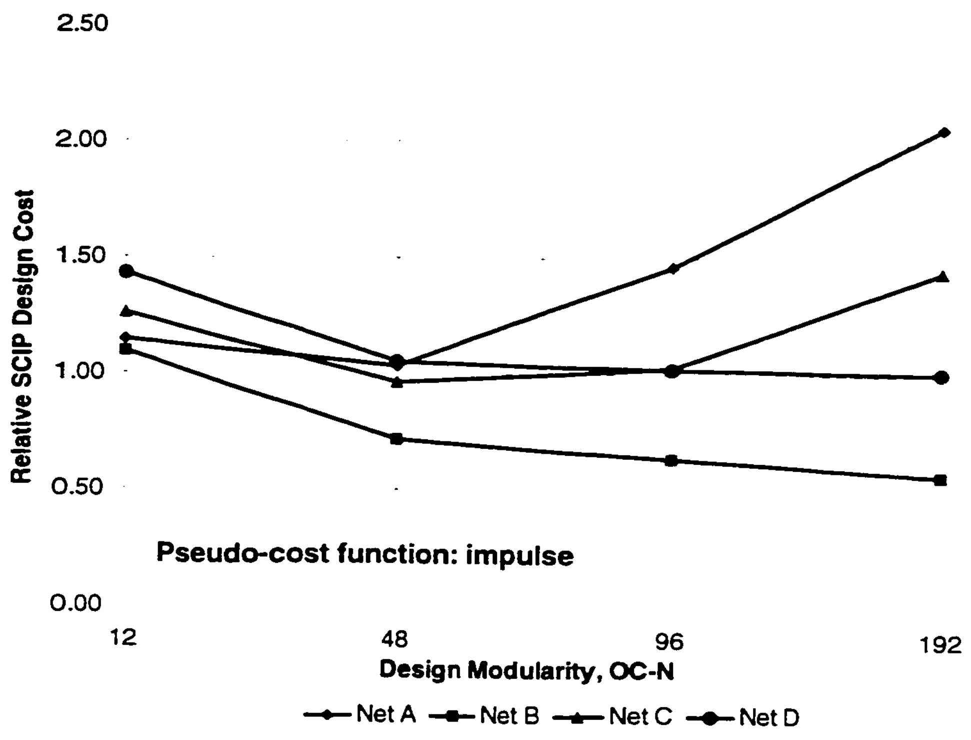


Figure 6.6. Effects of modular capacity on design cost with span eliminations by MAPR (Nets A-D, impulse pseudo-cost function).

Examining the results from Figures 6.5 and 6.6, it can be found that there exists a point for each network at which, for the given design modularity, the SCIP design cost is minimum. It is also apparent that further increasing the design modularity beyond this minimum point increases the SCIP design cost. When aggregation pressure is introduced to the demand routing, demands are routed over longer paths (relative to shortest-path) to efficiently fill up span modules that are currently in use (since they are cheaper to route), allowing the more expensive unused spans to remain at zero capacity. The effect of this routing method increases the total routing distance consumed by the network thereby increasing the network's average working capacity on each span relative to shortest-path routing. Therefore, when the design modularity is small, the increase in span capacity requirements is more likely to exceed any surplus or slack capacity in the span module, resulting in more rings being placed. Also, when the design modularity is small, more span modules are easily packed to their full modular capacity, hence the likelihood of unused spans remaining at zero working capacity decreases. Consequently, the number of span eliminations is relatively fewer (Table 6.2 to 6.5).

However, when the design modularity gets larger, it is more likely that the slack capacity in the span will be large enough to accommodate the increase in span capacity requirements, resulting in fewer rings being placed. Furthermore, increasing the design modularity increases the likelihood of unused spans remaining at zero working capacity, so relatively more span eliminations are observed. Nonetheless, using larger design modularity does not always lead to better cost results since the network equipment that handles larger capacity costs more (say two times the cost for four times the capacity). Hence, the design cost increases when the design modularity used becomes too large even though there are more span eliminations. Table 6.6 below summarizes the least cost results obtained by MAPR for rise and impulse.

Table 6.6. Least cost SCIP design obtained using MAPR(rise) and MAPR(impulse).

Network	MAPR(rise)			MAPR(impulse)		
	Design Modularity	SCIP Cost (Millions)	Comp. Time (seconds)	Design Modularity	SCIP Cost (Million)	Comp. Time (seconds)
A	12	1.94	0.13	48	2.00	0.01
B	96	17.03	3.69	192	19.47	75.83
C	48	12.78	0.18	48	13.93	0.01
D	96	33.49	0.87	192	46.71	0.36

As shown in Table 6.6, the least cost SCIP designs obtained from MAPR(rise) in all cases out-perform MAPR(impulse). When comparing MAPR(impulse) designs with designs using pure shortest-path routing and no span eliminations over all design modularities, only seven out of sixteen MAPR(impulse) designs showed an improvement in cost. Most of these cases occur when relatively large design modularity is permitted (compare the MAPR(impulse) results in Tables 6.2 to 6.5 against the shortest-path routing with no span eliminations designs in Table 3.3). A span implementing the impulse pseudo-cost function (Figure 4.5b) exhibits extreme aggregating pressure due to the minimum constant routing pseudo-cost for the span as long as it has any slack capacity, except when the span is unused or is at exact full modular value. As a result, demands tend to be routed over longer paths (relative to shortest-path and MAPR(rise)) since used span modules with minimum pseudo-cost 'attract' the demands until they are fully packed. Hence, subsequent demands are likely to be routed using these spans with minimum pseudo-cost, where the resulting paths are likely to be longer, leaving more spans unused and increasing the average working capacity on each span. Therefore, although more span eliminations are obtained, the design costs are still higher since there is not enough slack capacity in the spans to accommodate the increase in span capacity requirements leading to relatively more rings being placed. Only when relatively large design modularity is used does MAPR(impulse) show cost improvement over pure shortest-path routing with no span eliminations. Based on these interim results on MAPR(impulse), it was decided to focus only on MAPR(rise) since eleven out of sixteen designs showed cost improvements over pure shortest-path.

It is also important to note that the MAPR algorithm requires less computational time than its counterpart using pure shortest-path with no eliminations. The explaination for this is that MAPR eliminates the resulting non-essential spans that have zero capacity requirements and removes the cycles that use the eliminated non-essential spans from the cycle file before SCIP is executed. Therefore, with fewer cycles, MAPR exhibits relatively faster computational runtimes since SCIP has fewer possible designs to compute with respect to shortest-path and no eliminations.

6.1.3 IRE

A similar experimental process was used to investigate IRE. This heuristic was run on the four test networks with the corresponding demand matrices presented in Chapter 2.4. Previous experiments on MAPR showed that there exists a point for each network where, for a given design modularity, the SCIP design cost is at minimum. It was found that the minimum design costs occur when the design modularities are 12, 96, 48 and 96 for Net A, Net B, Net C and Net D respectively. Note that these results are similar to the results obtained in Table 3.3 for pure shortest-path with no span eliminations. With these findings, as a practical matter (since the number of SCIP runs in IRE rises quadratically with number of non-essential spans), the IRE investigations have been reduced to one design modularity per test network. The design modularities that have led to the minimum SCIP design cost described earlier have been adopted since they have shown the potential of giving the least min-cost results. Table 6.7 below summarizes the design modularity, the number of non-essential spans, and the required number of SCIP runs (Eq.5.1) for each test network.

Table 6.7. **The selection of design modularity, the number of non-essential spans and the number of SCIP runs for each test network in IRE investigation.**

Network	Module Size	No. of Non-Essential Spans, k	No. of SCIP runs
A	12	11	66
B	96	18	171
C	48	7	28
D	96	21	231

Figures 6.7 to 6.10 show the results of using IRE on Net A, Net B, Net C and Net D respectively. These results correspond to a specific combination of span eliminations that were found during the IRE span elimination process on each of the test networks. The eliminated spans for each IRE iteration number are displayed adjacent to each point shown on the graphs.

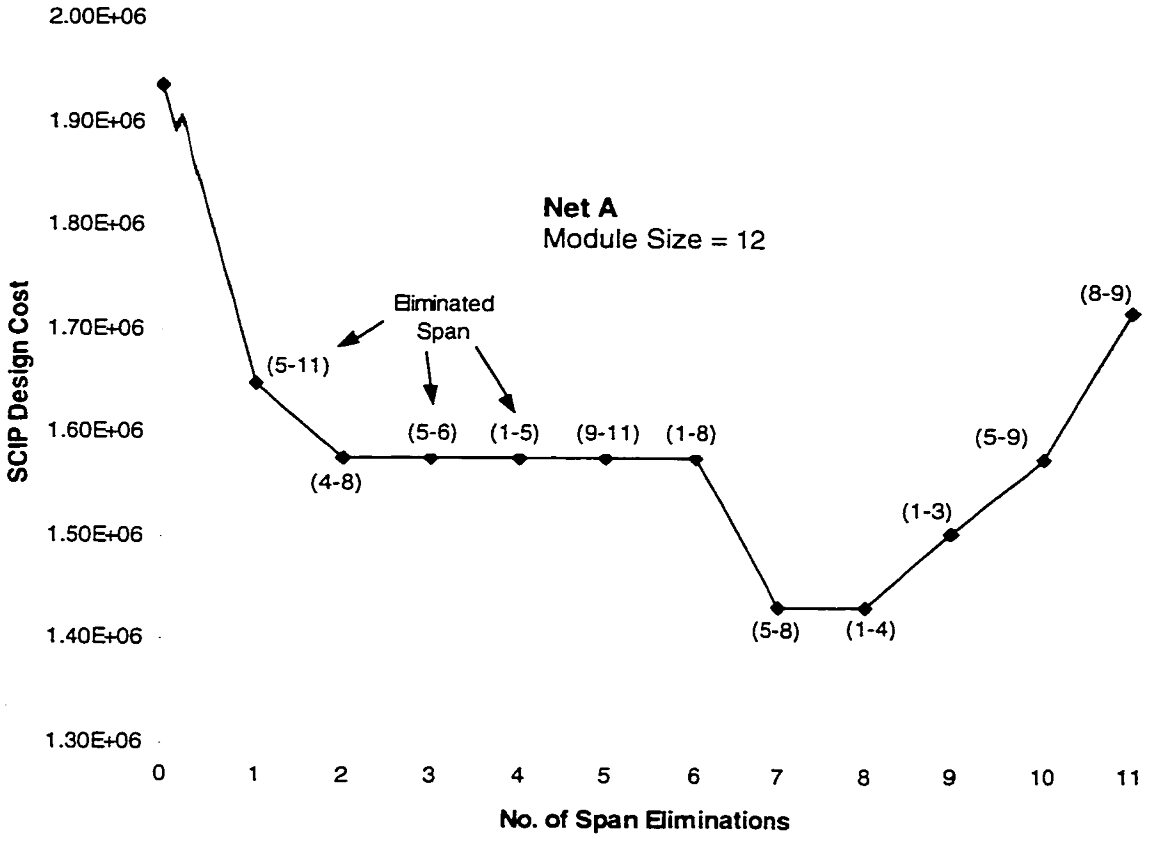


Figure 6.7. **SCIP results for Net A using IRE.**

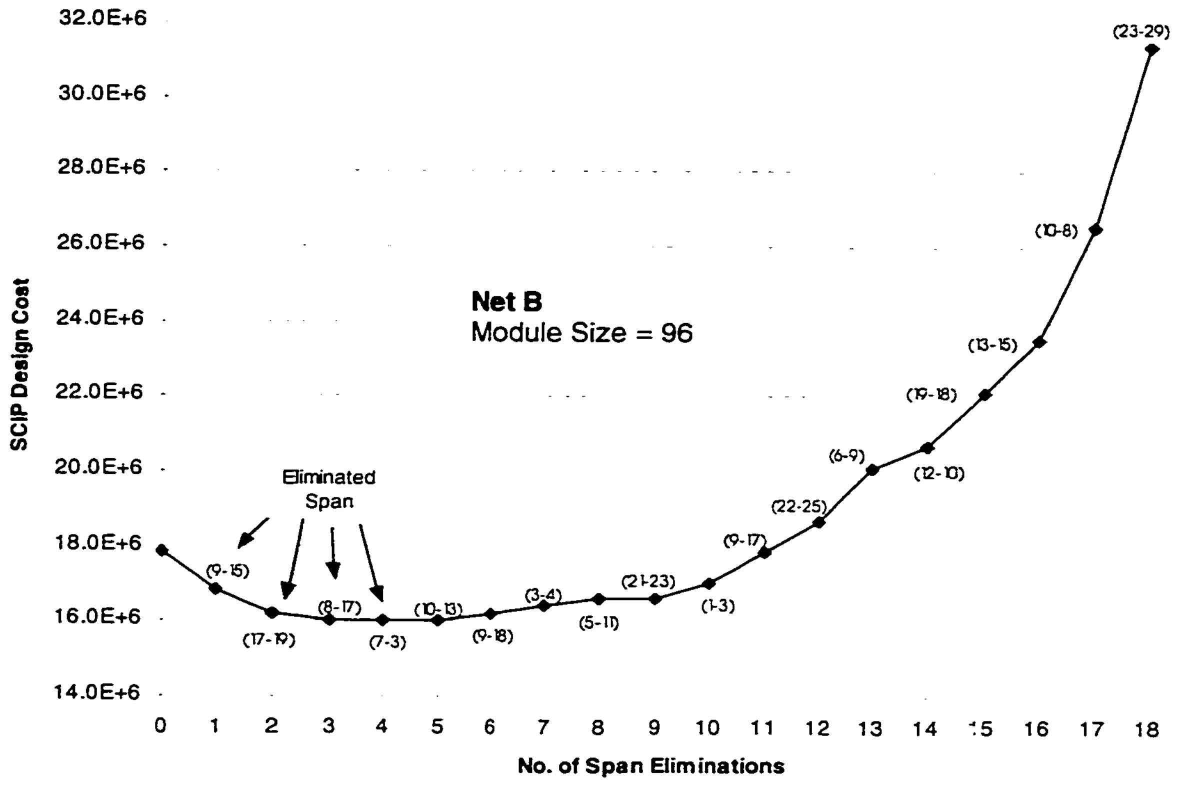


Figure 6.8. SCIP results for Net B using IRE.

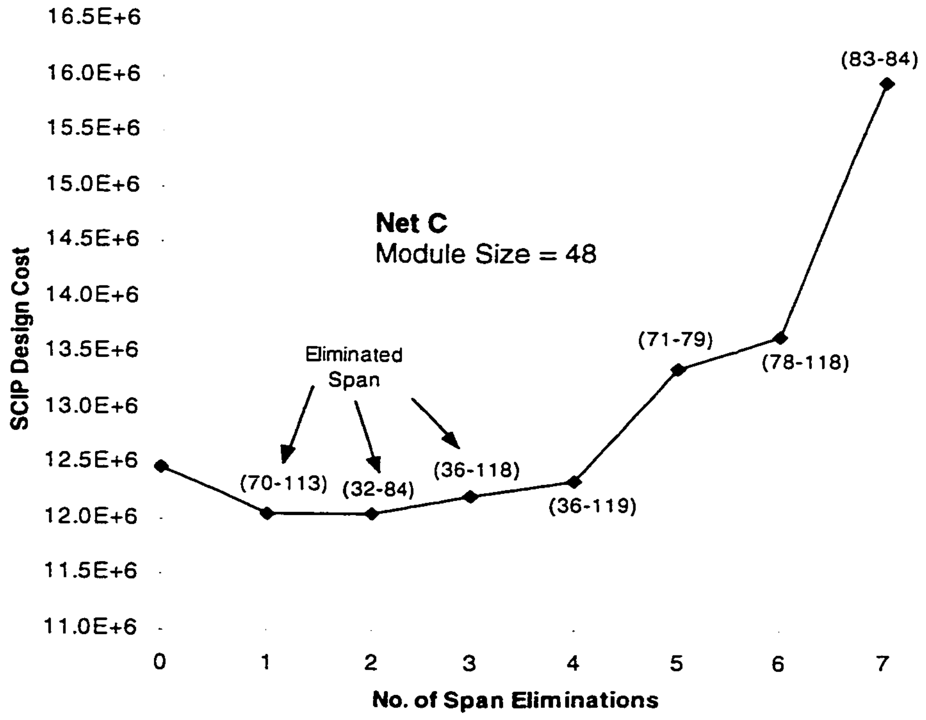


Figure 6.9. SCIP results for Net C using IRE.

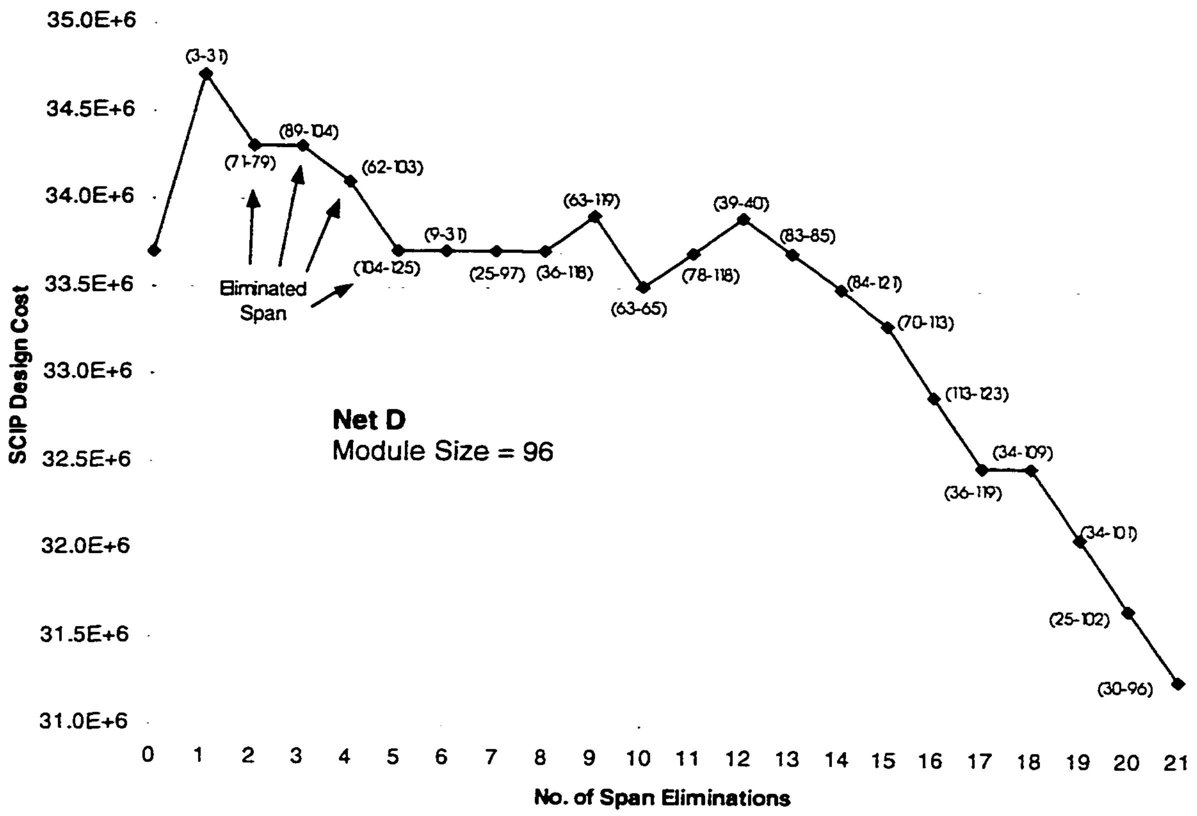
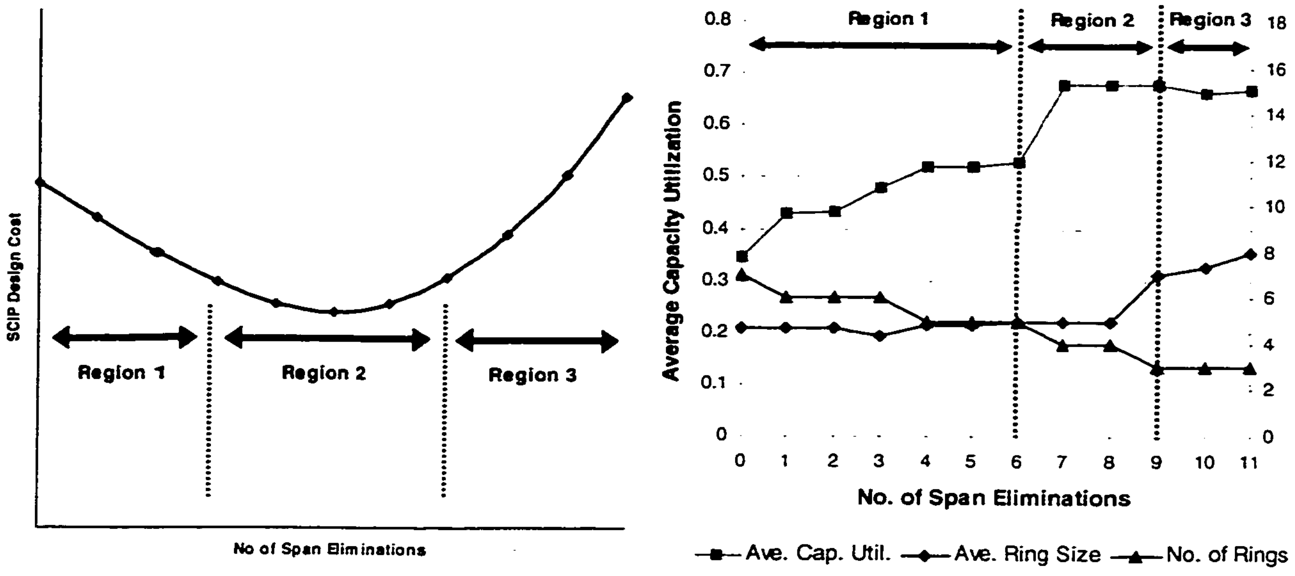


Figure 6.10. **SCIP results for Net D using IRE.**

It can be observed that a consistent* pattern occurs in all the IRE curves (with the exception for Net D in Figure 6.10, to be discussed later in this section) and that an optimum point (with respect to the points obtained in each respective figure) exists as the number of eliminations increases. In other words, it is possible to do too many span eliminations. The first falling section of the curves occurs when too many rings emerge in the design forced by the coverage requirement. These are lightly loaded rings forced upon the design to cover relatively light loaded spans. In this region costs can be reduced by eliminating spans and detouring the affected demands, sending them over longer alternate paths via other rings. Thus cost decreases as more spans are eliminated. A minimum value (most-improved design) is reached when the rings and spans achieve the best possible utilization. When more spans are removed, however, demands may be diverted over longer and longer paths, which eventually drives up the capacity requirements until ring capacity is exceeded and additional rings are triggered. These observations are further summarized in Figure 6.11.

* in the sense of the general behavior to be described in Figure 6.11.



(a) (b)

Figure 6.11. Typical IRE curve pattern. (a) Typical IRE cost vs. span eliminations curve. (b) Typical ring network design characteristics for each region (this model is obtained using IRE results for Net A).

Figure 6.11a illustrates the typical IRE cost curve divided into three regions and Figure 6.11b shows the typical ring network characteristics for each region. Specifically, the IRE results for Net A are used to portray this figure illustrating the average capacity utilization, the average ring size and the number of rings for each region to provide a clearer understanding of span eliminations. Region 1 represents the initial phase of span eliminations in IRE. In this region, there are many spatially distinct and lightly loaded rings forced by the strict coverage requirements resulting in relatively more low-utilized rings. Eliminating spans and detouring the affected demands over alternate paths via other rings gradually reduces the SCIP design cost until a minimum value is reached. Here the rings and spans achieve the best possible utilization requiring relatively fewer number of rings and higher average capacity utilization as shown in Region 2. Further span eliminations progressively drive up the capacity requirements until the ring capacity is exceeded. Subsequently, Region 3 tends to have many large similar stacked rings triggering more and higher design cost.

A further explaination is required for the IRE curve for Net D depicted in Figure 6.10 which shows that the overall SCIP design cost drops as more spans are eliminated and the minimum SCIP design cost occurs when all the non-essential spans are eliminated. This curve is in fact portraying Region 1 (and/or parts of Region 2) of Figure 6.11a. This curve shows that using design modularity of OC-96 for Net D is large enough to accommodate the increase in span capacity requirements due to the span elimination and re-routing of demands over longer paths, without triggering additional rings.

Table 6.8 below summarizes the cost results for the most-improved design with the corresponding number of span eliminations, the span elimination sequences and the computational time obtained using IRE for each test network.

Table 6.8. The minimum SCIP results using IRE single-modular capacity design.

Network	Module Size	No. of Span Elim.	Span Elim. Sequence	SCIP Design Cost (Millions)	Comp. Time (seconds)
A	12	8	(5-11)-(4-8)-(5-6)-(1-5)-(9-11)-(1-8)-(5-8)-(1-4)	1.43	37.45
B	96	5	(9-15)-(17-19)-(8-17)-(7-3)-(10-13)	16.02	14,243.50
C	48	2	(70-113)-(32-84)	12.05	83.20
D	96	11	(3-31)-(71-79)-(89-104)-(62-103)-(104-125)-(9-31)-(25-97)-(36-118)-(63-119)-(63-65)-(78-118)-(39-40)-(83-85)-(84-121)-(70-113)-(113-123)-(36-119)-(34-109)-(34-101)-(25-102)-(30-96)	31.27	3,813.58

6.2 Interim Conclusions

This section compares the performance of MAPR and IRE for single-modular capacity design. Several interim conclusions were drawn which affect the direction of the ideas and investigations of other span elimination heuristics in the second half of this work.

6.2.1 Comparison of the Performance of MAPR and IRE

Table 6.9 below shows the results found for MAPR and IRE. These results were compared against SCIP for the shortest-path routing case with no span eliminations (Table 3.3). Figure 6.12 below gives the percentage decrease (increase) in the total design for IRE and MAPR algorithms relative to the cost obtained with shortest-path routing and no span eliminations.

Table 6.9. **The most-improved SCIP results and the total computational time using pure shortest-path (no eliminations), MAPR (rise) and IRE.**

Network	Modular Capacity	Pure Shortest-Path		MAPR (rise)		IRE	
		Min-Cost SCIP Design (Millions)	Comp. Time (seconds)	Min-Cost SCIP Design (Millions)	Comp. Time (seconds)	Min-Cost SCIP Design (Millions)	Comp. Time (seconds)
A	12	1.94	3.63	1.94	0.13	1.43	37.45
B	96	17.84	19.47	17.03	3.69	16.02	14243.5
C	48	12.48	0.08	12.78	0.18	12.05	83.2
D	96	33.70	32.13	33.49	0.87	31.27	3813.58

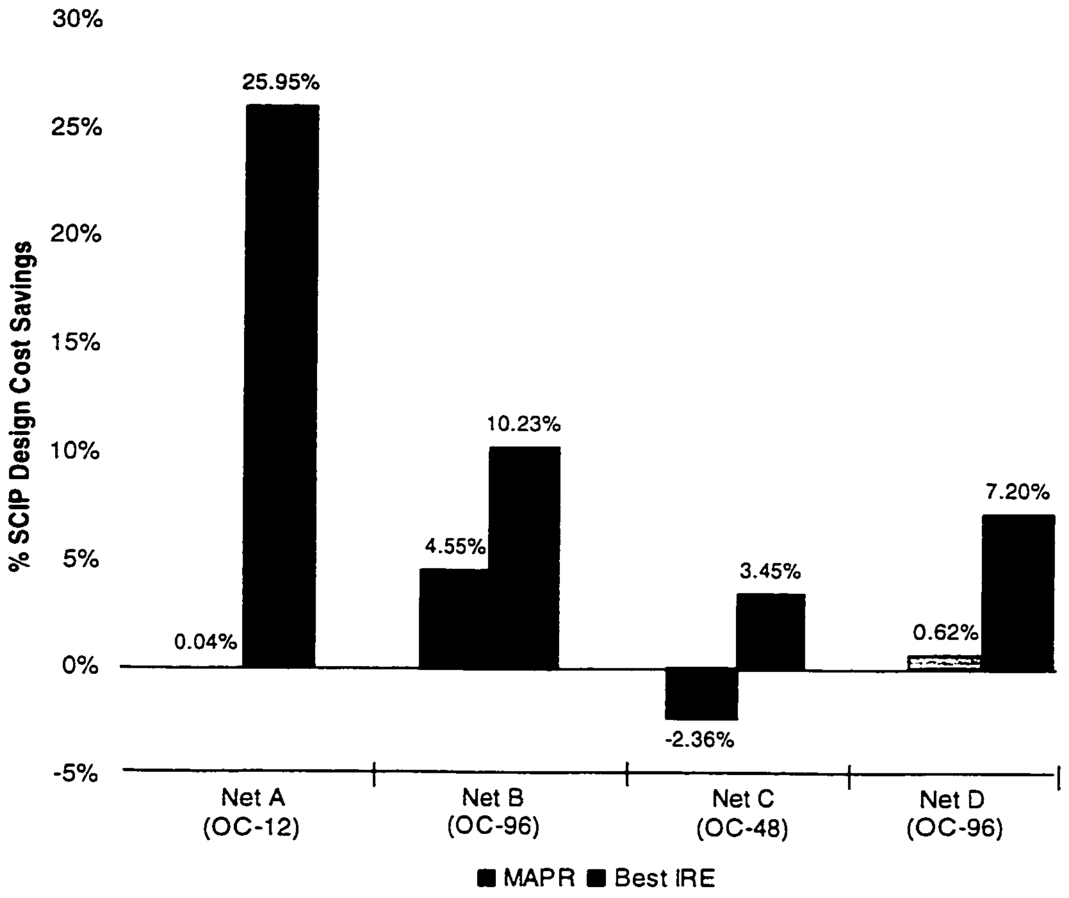


Figure 6.12. **Percentage Total Network Cost Savings for MAPR/SCIP(rise) and IRE/SCIP.**

The SCIP design cost savings reported by IRE range from 3% to 26% as compared to -2 % to 5% reported by MAPR. These results show that the most-improved design found by a run of the IRE algorithm consistently outperformed the MAPR algorithm. The cost increase of 2% by MAPR using rise for Net C was a case where too many eliminations occurred requiring significantly more or larger (longer) rings to cover the remaining sub-graph within which demands were severely detoured. Hence, for this situation, using shortest-path with no span eliminations will be the preferred solution, which implies that MAPR has resulted in zero cost savings*. It can be observed that there are instances where MAPR may not be the suitable approach to use and in hindsight it seems clear why: MAPR results in one specific set of eliminated spans and re-routed flows whereas an IRE generates a progression of increasingly span-eliminated designs to consider.

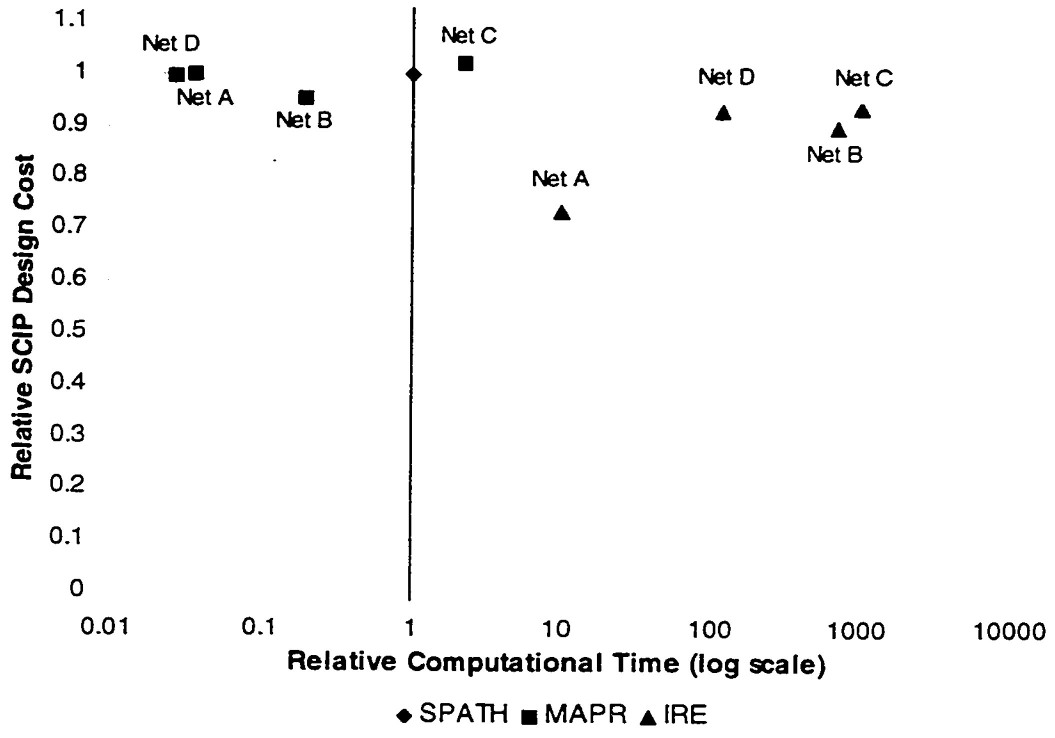


Figure 6.13. **Scatter plot for MAPR and IRE results normalized to corresponding shortest-path and no span elimination results to show the relationship between solution quality and computational time (Nets A-D pooled).**

* The MAPR algorithm alone does not perform trial design using shortest-path with no span eliminations. In practical situation, a planner will perform a baseline design using shortest-path with no span eliminations before attempting to initiate any span elimination strategies. Therefore to reflect practical interest, negative cost savings will be denoted as zero cost savings.

On the other hand, IRE has a considerably longer computational time than MAPR (as shown in Table 6.9) prompting a trade-off in terms of solution quality and computational time. This relationship is shown in the scatter plot in Figure 6.13 for the SCIP design cost and the computational time results obtained using MAPR and IRE normalized to the results using shortest-path and no span eliminations. The *x*-axis (Relative Computational Time) is the computational time results obtained from MAPR and IRE normalized by the respective computational time results using shortest-path and no span eliminations (refer to Table 6.9). The *y*-axis (Relative SCIP Design Cost) is the SCIP design cost obtained from MAPR and IRE normalized by the respective SCIP results for shortest-path routing and no span eliminations. The full run-time for IRE is high (the IRE points are scattered to the right of Figure 6.13) due to its built in breadth-first search strategy, of which each iteration employs an SCIP coverage design. In large networks, there are likely to be many non-essential spans, causing an increase in IRE run-times since the number of SCIP runs rises quadratically with the number of non-essential spans. MAPR, however, requires only a single SCIP run regardless of the size of the network (MAPR points are scattered to the left of Figure 6.13).

6.2.2 Discussions and Conclusions

As presently described, the conclusion must be that MAPR is not reliable as a span elimination heuristic. The increase in design cost obtained for Net C signifies that MAPR has over-done the span elimination. In essence, MAPR is a one-shot span elimination strategy that depends on the aggregation pressure applied to the span using the capacity-sensitive pseudo-cost function to efficiently route demands on the network, leading to a certain number of span eliminations. MAPR has no means of indicating that a pseudo-cost function may have applied too much aggregation pressure, leading to too many span eliminations that result in high coverage design cost. In principle, to solve this problem, *all* possible pseudo-cost functions could be examined with different aggregation pressure to capture the most-improved design cost. However, since there is an infinite number of pseudo-cost curves, and each span in the network may require different aggregation pressure, this is impractical.

From the results in Table 6.9, it can be seen that the advantage of using MAPR is speed whereas the advantage of using IRE is that it generates a succession on the number of span eliminations where the overall min-cost (most-improved) design is captured. Consequently, in Chapter 7, an improvement to MAPR has been developed by allowing a specific number of pseudo-cost functions to be examined, incorporating both of the above properties. This improved strategy is called Iterative MAPR (iMAPR).

At this point, consideration of multi-modularity is also added. Chapters 4 and 5 investigate MAPR and IRE using single-modular design where each resulting design obtained is limited to only a single ring modular capacity. The results summarized in Figure 6.5 suggest a strong dependency between the min-cost design and selection of modular capacity. In addition, design costs are limited since there are situations where using a smaller ring modular capacity is adequate to cover the same demands previously covered by a larger ring modular capacity. There are also situations where it will be cost-effective to use a larger ring that has twice the modular capacity to cover the same demands previously covered by two smaller rings with one half the modular capacity of the larger ring (due to economy-of-scale effects). By implementing multi-modular design, rings can now be selected in terms of cost and modular capacity to cover the working demands in the network, which will in turn result in better and improved cost results relative to single-modular designs. In the chapters to follow, multi-modular SCIP design will be discussed. The IRE algorithm is revisited in Chapter 9 using multi-modular SCIP design.

For the reader's information, the work done in Chapters 3, 4, 5 and 6 was the basis of the related conference paper in [33].

7.0 Ideas for Modification and Improvements for MAPR

The conclusions in Chapter 6 led to the modification and improvement of the MAPR span elimination heuristic. This chapter details the modifications and improvements that have been made.

7.1 Iterative MAPR (iMAPR)

7.1.1 Introduction

In Chapter 6, it was found that MAPR using the rise pseudo-cost function gave better results than MAPR using the impulse pseudo-cost function. This was deemed to be due to the effects of very high aggregation pressure. Applying too much aggregation pressure tends to draw demands to route over longer alternate paths relative to shortest-path routing and MAPR(rise), leading to an increase in average working capacity on each span. Here, slack capacity in the spans is not able to accommodate the increase, resulting in additional rings being required to cover the excess capacities where subsequent SCIP runs will yield higher design costs. In contrast with MAPR(rise), there is a compromise in the level of aggregation pressure applied to the spans, leading to a relatively smaller increase in average working capacity which slack capacity in spans is more likely to be able to accommodate, resulting in better SCIP design costs.

These insights about the relationship between the aggregation pressure applied to the spans and the SCIP design cost led to the idea of progressively applying the aggregation pressure from weak to strong seeking the best-fit aggregation pressure that will result in the most-improved SCIP design. Using rise and impulse alone for general investigations of all networks is not sufficient when searching for and determining the best possible minimum cost.

Iterative MAPR(iMAPR) improves on MAPR by stepping through a family of pseudo-cost functions, allowing the capture of one sub-graph that gives the most-improved SCIP coverage design. Here, there is an iterative strategy where MAPR (which itself is very fast) is executed at each iteration, with a different pseudo-cost curve each time. The ideal would to examine all possible pseudo-cost curves, but doing this would be impractical as there could be an infinite number of possible curves. Therefore, a family of pseudo-cost curves have been selected, and starts with the first iteration using a pseudo-cost curve that has a very light level of aggregation pressure. For each subsequent iteration, the level of aggregation pressure is increased in nearly constant steps until the final iteration is reached where the curve has the highest level of aggregation pressure, much like an impulse pseudo-cost function. The equations and selected pseudo-cost functions are described in Chapter 7.1.2.

Another approach to improving problem fidelity is to utilize SCIP using multi-modular design. The same modular span capacities, as well as network equipment, ring configurations and cost models described in Table 3.1 and Table 3.2, are used. The SCIP formulation for multi-modular capacity design is the same formulation described in Chapter 3.2.2. For multi-modular design, SCIP generates a set of ring candidates *R* containing rings with different modular capacities and searches for the set of ring candidates (that may consist of ring candidates with different modular capacities) that will lead to the min-cost ring coverage design.

Changing single-modular SCIP to multi-modular capacity SCIP has prompted changes to the location of the peak pseudo-cost in the pseudo-cost curve, previously defined in MAPR (Chapter 4.2.2). When single-modular SCIP is used, peak pseudo-cost occurs when the span capacity requirement is at span modular capacity or multiples of span modular capacity. For example, if the current investigation is using a modular capacity of OC-12, peak pseudo-cost occurs when the span capacity requirement is at 0, 12, 24, 36, 48 and so on; or if, the modular capacity is OC-192, peak pseudo-cost occurs at span capacity requirements of 0, 192, 384, 576 and so on. The reason behind this is to have the pseudo-cost for deploying a new span module at its highest. By doing so, the pseudo-cost of placing a demand on a span that has already reached its full modular capacity is at the highest. However, once a new span module has already been 'paid' for (meaning once a demand is placed over a new span module), the cost for placing subsequent demands on that span is relatively lower since the aim is to utilize the span module as much as possible until it reaches its next full modular capacity where a full period is repeated again.

When several different modular capacities are possible in multi-modular SCIP, it would be more economical to use a larger span modular capacity to replace a smaller span modular capacity when there are excess demands placed over the smaller span modular capacity. For example, in the single-modular situation, if there are excess demands being placed over an OC-48 span module, an additional OC-48 span module is required to cover the excess demand; however, in a multi-modular SCIP, it would be more economical to use a single OC-96 span module in place of the two OC-48 span modules since the cost of a single OC-96 is relatively cheaper. Therefore, the pseudo-cost function has been adjusted in a way that will select the best span modular capacity. For example, if the span capacity requirement is in the range of 49 to 96, using a span with a modular capacity of OC-96 would be the most economical. Therefore, the peak pseudo-cost occurs when the span's capacity requirement is at 96. Alternatively, if the span's capacity requirement is within the range of 193 to 204, the most economical combination of span modules would be to use a single OC-192 span module and a single OC-12 span module. Here, the peak pseudo-cost occurs when the span's capacity requirement is at 204. By determining these ranges and the best combination of span modules for those ranges, the positions for the peak pseudo-cost can be determined. So far, it has been determined that the peak pseudo-cost occurs when the span capacity requirement is at 0, 12, 48, 96, 192, 204, 240, 288, 384, and so on; notice that the curves between 0 and 192 are repeated for every period of 192.

There are, however, exceptions. Using the network equipment and fibre cost depicted in Table 3.1, if the span capacity requirement is in the range of 97 to 108, the most economical combination of span modules is, in fact, using a single OC-96 span module and a single OC-12 span module (total cost = $273,063). As described earlier, it is assumed that using a single OC-192 span module is the most economical (total cost = $285,332). Hence, for simplicity's sake, the peak pseudo-cost described earlier has been adopted.

Section 7.1.2 provides more detail about the pseudo-cost equations that are used and the adjustments that are made to reflect the family of selected pseudo-cost curves and the changes to the location of the peak pseudo-cost when multi-modular SCIP is implemented.

7.1.2 iMAPR Pseudo-Cost Function

With the conceptual understanding of the desired pseudo-cost function from the previous section, two conditions are defined that are similar to those adopted by MAPR. These two conditions are used to formulate the general equation for iMAPR pseudo-cost function. The conditions are as follows:

$$c_i(w_i) = \begin{cases} d_i \cdot G, & (w_i \text{ MOD } M(w_i)) = 0; \\ d_i, & (w_i \text{ MOD } M(w_i)) = 1; \end{cases} \qquad \textbf{(7.1)}$$

$c_i(w_i)$	- Pseudo-cost of span i with w_i capacity requirements where $0 \le w_i \le 192$.
d_i	- Geographical distance of span i.
$G = 10$	- An arbitrarily selected constant multiplier set to 10.
$M(w_i)$	- the next modular capacity that is smaller than or equal to w_i where $M(w_i) \in \{0, 12, 48, 96, 192\}$. For example, if $w_i = 13$, then $M(w_i) = 12$, or if $w_i = 96$, then $M(w_i) = 96$.

The conditions shown in Equation 7.1 only apply for $0 \le w_i \le 192$. Later in this chapter, the procedure of calculating pseudo-cost for $w_i > 192$ will be illustrated.

The two conditions in (7.1) are defined as (i) the peak pseudo-cost which occurs when w_i is at 0 or an exact full modular value of 12, 48, 96 or 192 and, (ii) the minimum pseudo-cost occurs when one demand is piled over a new span module or over an exact full module.

A rise-like pseudo-cost function with the capability of varying the aggregation pressure is of interest. Notionally this is related to the rate of ascent of the rise pseudo-cost function. Therefore, a family of exponential functions, parameterized by a strength parameter α to control the aggregation pressure has been utilized. Two general exponential equations were used: a general negative exponential equation to define a set of upper curves, and a general positive exponential equation to define a set of lower curves (Figure 7.1) [*]. The general expressions are shown in Equation 7.2 and Equation 7.3, respectively.

[*] It was later found that it is possible to use one exponential equation to define both the set of upper curves and the set of lower curves, which can further simplify iMAPR pseudo-cost.

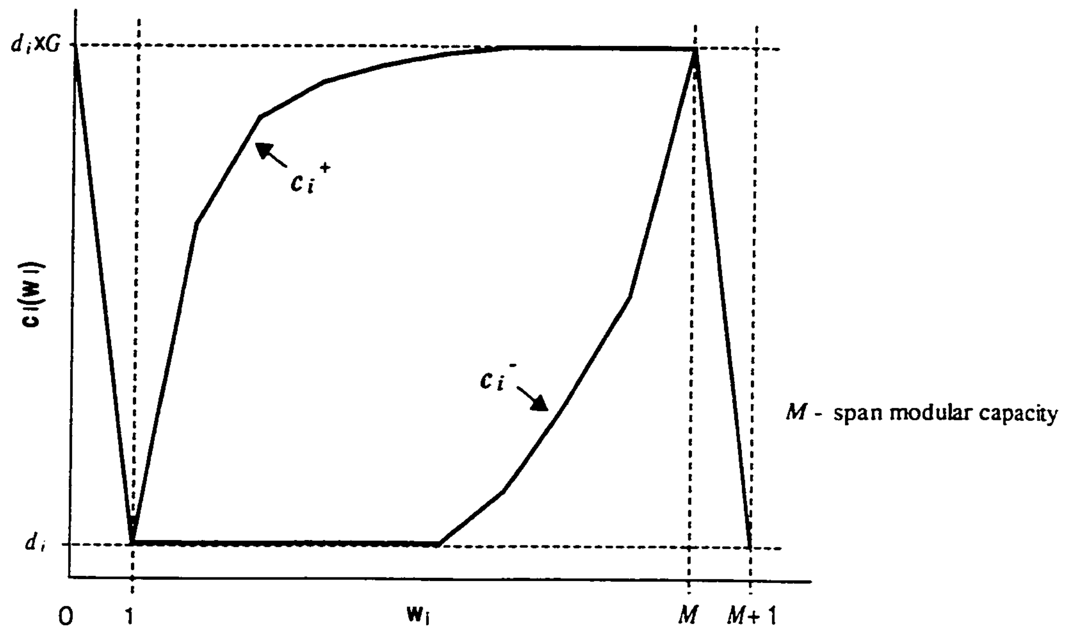


Figure 7.1 Desired c_i^+ and c_i^- exponential curves.

$$c_i^+ = B^+(1 - A^+e^{-\alpha x}) \qquad 0 < \alpha \le 1 \tag{7.2}$$

$$c_i^- = B^-(1 + A^-e^{\alpha x}) \qquad 0 < \alpha \le 1 \tag{7.3}$$

$c_i^+(w_i)$	- Pseudo-cost for the set of upper curves.
$c_i^-(w_i)$	- Pseudo-cost for the set of lower curves.
$x = w_i$ MOD $M(wi)$	- The remainder of w_i divided by $M(w_i)$ where $M(w_i)$ is the next modular capacity that is smaller than or equal to w_i where $M(w_i) \in \{0, 12, 48, 96, 192\}$ and $0 \le w_i \le 192$.
α	- Aggregation pressure coefficient, $0 < \alpha \le 1$.
A^+, B^+, A^-, B^-	- Coefficients.

Note that Equations 7.1, 7.2 and 7.3 take multi-modular capacities into consideration. Next, the steps involved in deriving the general iMAPR pseudo-cost equation are shown.

Step 1: Derive c_i^+ and c_i^- for single modularity.

First, c_i^+ is isolated for single modularity. For $w_i = 0$, c_i^+ is equal to $d_i \cdot G$. For $0 < w_i \leq M$, c_i^+ is at minimum, d_i, when w_i is equal to 1 and c_i^+ is at maximum, $d_i \cdot G$, when w_i is equal to M, where M is the module size. Substituting these values into Equation 7.2 for $x = w_i$, the following equations are obtained for single modularity:

$$d_i = B^+(1 - A^+ e^{-\alpha}) \qquad 0 < \alpha \leq 1 \tag{7.4}$$

$$d_i \cdot G = B^+(1 - A^+ e^{-\alpha M}) \qquad 0 < \alpha \leq 1 \tag{7.5}$$

Using Eq. 7.4 and Eq. 7.5, the solution for A^+ and B^+ is found, thereby deriving c_i^+ for single-modularity (Equation 7.6):

$$A^+ = \frac{G-1}{G \cdot e^{-\alpha} - e^{-\alpha \cdot M}} \qquad B^+ = \frac{d_i}{1 - \left(\frac{G-1}{G \cdot e^{-\alpha} - e^{-\alpha \cdot M}} \right) \cdot e^{-\alpha}}$$

$$c_i^+ = \begin{cases} d_i \cdot G, & w_i = 0 \\ \dfrac{d_i(Ge^{-\alpha} - e^{-\alpha \cdot M} - Ge^{-\alpha \cdot w_i} + e^{-\alpha \cdot w_i})}{e^{-\alpha} - e^{-\alpha \cdot M}}, & 0 < \alpha \leq 1, \; 0 < w_i \leq M \end{cases} \tag{7.6}$$

Similarly, for $w_i = 0$, c_i^- is equal to $d_i \cdot G$. For $0 < w_i \leq M$, c_i^- is at minimum, d_i, when w_i is equal to 1 and c_i^+ is at maximum, $d_i \cdot G$, when w_i is equal to M, where M is the module size. Substituting these values into Equation 7.3 for $x = w_i$, the following equations are obtained for single modularity:

$$d_i = B^-(1 + A^- e^{\alpha}) \qquad 0 < \alpha \leq 1 \tag{7.7}$$

$$d_i \cdot G = B^-(1 + A^- e^{\alpha M}) \qquad 0 < \alpha \leq 1 \tag{7.8}$$

Using Equations 7.7 and 7.8, the solution for A^- and B^- is found and c_i^- is derived for single-modularity (Equation 7.9):

$$A^- = \frac{G-1}{e^{\alpha \cdot M} - G \cdot e^{\alpha}} \qquad B^- = \frac{d_i}{1 + \left(\frac{G-1}{e^{\alpha \cdot M} - G \cdot e^{\alpha}} \right) \cdot e^{\alpha}}$$

$$c_i^- = \begin{cases} d_i \cdot G, & w_i = 0 \\ \dfrac{d_i(e^{\alpha \cdot M} - Ge^{\alpha} + Ge^{\alpha \cdot w_i} - e^{\alpha \cdot w_i})}{e^{\alpha \cdot M} - e^{\alpha}}, & 0 < \alpha \le 1, \quad 0 < w_i \le M \end{cases} \tag{7.9}$$

Step 2: Derive c_i^+ and c_i^- for multi-modularity using c_i^+ and c_i^- for single modularity.

First the locations of the breakpoints in the pseudo-cost curves (where the peak pseudo-cost arises in multi-modularity) are determined. As described in the previous section, the breakpoints are intended to occur when w_i equals 12, 48, 96 and 192 for $0< w_i \le 192$. As such, Δ is introduced as the difference between two subsequent breakpoints. The Δ values are as follows:

$$\Delta = \begin{cases} 12, & 0 < (w_i \text{ MOD } 192) \le 12 \\ 36, & 12 < (w_i \text{ MOD } 192) \le 48 \\ 48, & 48 < (w_i \text{ MOD } 192) \le 96 \\ 96, & 96 < (w_i \text{ MOD } 192) \le 192 \end{cases} \tag{7.10}$$

By replacing M with Δ and w_i with x where x is (w_i MOD $M(w_i)$), c_i^+ and c_i^- are obtained for multi-modularity as shown by Equation 7.11 and Equation 7.12 below.

$$c_i^+ = \begin{cases} d_i \cdot G, & x = 0 \\ \dfrac{d_i(Ge^{-\alpha} - e^{-\alpha \cdot \Delta} - Ge^{-\alpha \cdot x} + e^{-\alpha \cdot x})}{e^{-\alpha} - e^{-\alpha \cdot \Delta}}, & \begin{array}{l} 0 < \alpha \leq 1 \\ 0 < x \leq 192 \end{array} \end{cases} \quad \textbf{(7.11)}$$

$$c_i^- = \begin{cases} d_i \cdot G, & x = 0 \\ \dfrac{d_i(e^{\alpha \cdot \Delta} - Ge^{\alpha} + Ge^{\alpha \cdot x} - e^{\alpha \cdot x})}{e^{\alpha \cdot \Delta} - e^{\alpha}}, & \begin{array}{l} 0 < \alpha \leq 1 \\ 0 < x \leq 192 \end{array} \end{cases} \quad \textbf{(7.12)}$$

where,

$x = w_i$ MOD $M(wi)$, $0 \leq w_i \leq 192$	- The remainder of w_i divided by $M(w_i)$ where $M(w_i)$ is the next modular capacity that is smaller than or equal to w_i where $M(w_i) \in \{0, 12, 48, 96, 192\}$.
$G = 10$	- Constant multiplier.
Δ	- The difference between two subsequent breakpoints, Equation 7.10.
α	- Aggregation pressure coefficient.
d_i	- Geographic distance of span i.

Step 3: Define α for c_i^+ and c_i^- for 21 pseudo-cost curves.

Next, the values for α are defined to determine the strength of aggregation pressure for each upper and lower curve. Here, the criterion is established that each subsequent pseudo-cost curve from the highest aggregation pressure to the lowest should be roughly equally spaced for use in iMAPR. Since 21 different pseudo-cost curves will be examined, α in Equations 7.11 and 7.12 is replaced by $\alpha_\Delta(r)$, where r is the curve number, and Δ is defined by Equation 7.10. The updated equations for c_i^+ and c_i^- are shown in Equation 7.13 and Equation 7.14, respectively.

$$c_i^+ = \begin{cases} d_i \cdot G, & x=0 \\ \dfrac{d_i(Ge^{-\alpha_\Delta(r_1)} - e^{-\alpha_\Delta(r_1)\cdot\Delta} - Ge^{-\alpha_\Delta(r_1)\cdot x} + e^{-\alpha_\Delta(r_1)\cdot x})}{e^{-\alpha_\Delta(r_1)} - e^{-\alpha_\Delta(r_1)\cdot\Delta}}, & 0<\alpha\leq 1,\ 0\leq r_1\leq 10 \\ & 0<x\leq 192 \end{cases} \quad \textbf{(7.13)}$$

$$c_i^- = \begin{cases} d_i \cdot G, & x=0 \\ \dfrac{d_i(e^{\alpha_\Delta(r_2)\cdot\Delta} - Ge^{\alpha_\Delta(r_2)} + Ge^{\alpha_\Delta(r_2)\cdot x} - e^{\alpha_\Delta(r_2)\cdot x})}{e^{\alpha_\Delta(r_2)\cdot\Delta} - e^{\alpha_\Delta(r_2)}}, & 0<\alpha\leq 1,\ 11\leq r_2\leq 20 \\ & 0<x\leq 192 \end{cases} \quad \textbf{(7.14)}$$

where,

$x = w_i$ MOD $M(wi)$, $0 \leq w_i \leq 192$	- The remainder of w_i divided by $M(w_i)$ where $M(w_i)$ is the next modular capacity that is smaller than or equal to w_i where $M(w_i) \in \{0, 12, 48, 96, 192\}$.
$G = 10$	- Constant multiplier.
$\alpha_\Delta(r_1)$, $\alpha_\Delta(r_2)$	- Aggregation pressure coefficient.
	- $\Delta \in \{12, 36, 48, 96\}$, Eq.7.10.
	- r_1 curve number, integer, $0 \leq r_1 \leq 10$
	- r_2 curve number, integer, $11 \leq r_2 \leq 20$
d_i	- Geographic distance of span i.

Using c_i^+ and c_i^- in Equations 7.13 and 7.14 described above, a set of $\alpha_\Delta(r_1)$ and $\alpha_\Delta(r_2)$ values was obtained through trial and error for use in iMAPR. These are listed in Table 7.1. An example of iMAPR pseudo-cost function for a span i with d_i=15 is shown in Figure 7.1. The pseudo-code illustrated in Figure 7.2 is used to calculate the pseudo-cost for all w_i. For w_i > 192, the pseudo-cost curves shown in Figure 7.1 are repeated.

Table 7.1. **Aggregation pressure coefficient, $\alpha_\Delta(r_1)$ and $\alpha_\Delta(r_2)$, for corresponding Δ, r_1 and r_2 values determined through trial and error.**

	Δ / Curve No.	12	36	48	96
r_1	0	1	1	1	1
	1	0.75	0.51	0.51	0.35
	2	0.58	0.32	0.32	0.18
	3	0.46	0.22	0.22	0.11
	4	0.36	0.16	0.16	0.078
	5	0.28	0.12	0.12	0.056
	6	0.22	0.09	0.09	0.042
	7	0.16	0.07	0.065	0.03
	8	0.11	0.05	0.045	0.02
	9	0.06	0.03	0.025	0.01
	10	0.01	0.01	0.007	0.001
r_2	11	0.045	0.01	0.01	0.008
	12	0.095	0.03	0.028	0.018
	13	0.15	0.055	0.048	0.03
	14	0.21	0.08	0.071	0.042
	15	0.265	0.11	0.098	0.059
	16	0.34	0.145	0.135	0.082
	17	0.44	0.2	0.2	0.12
	18	0.55	0.3	0.3	0.19
	19	0.73	0.5	0.5	0.35
	20	1	1	1	1

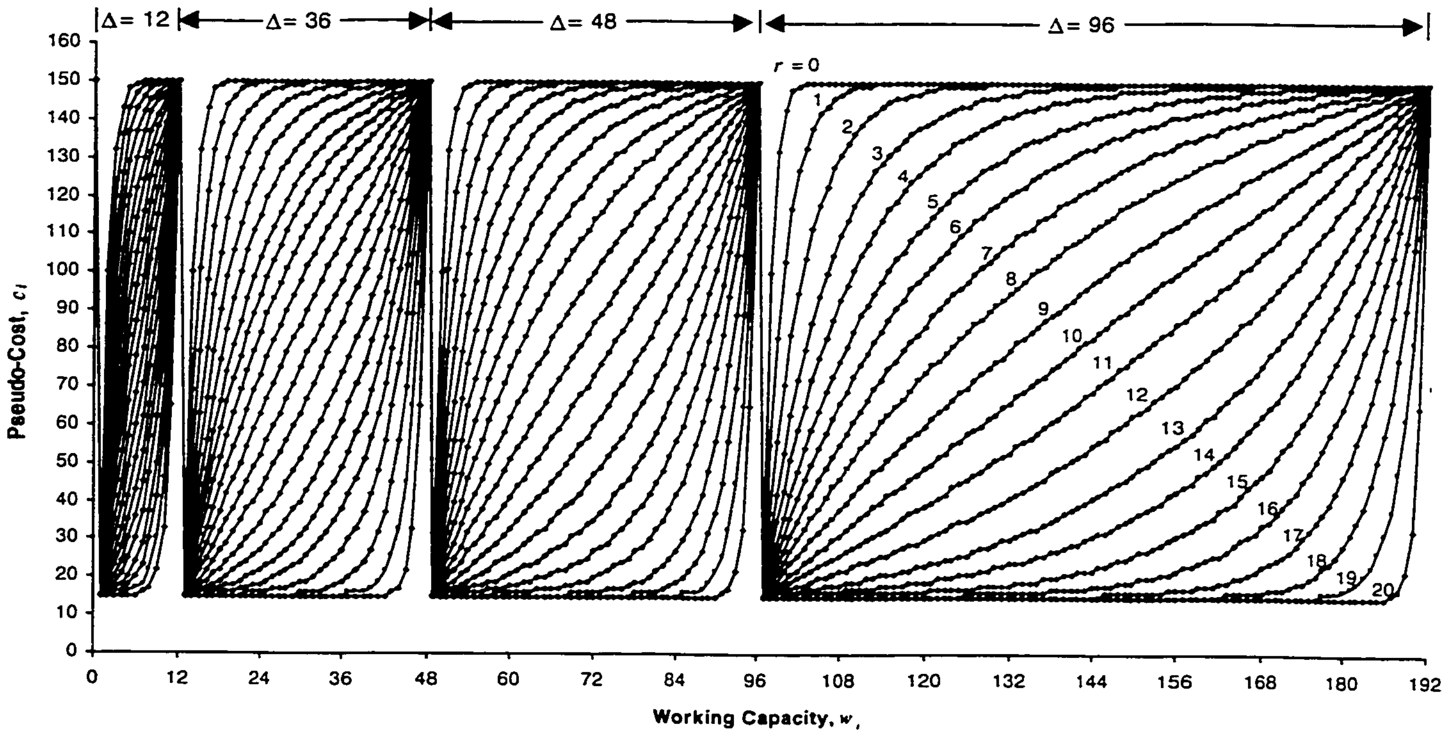


Figure 7.2. **An example of a non-periodical pseudo-cost function for iMAPR with $d_i = 15$ for $0 \le w_i \le 192$.**

```
START cal_pseudo_cost_iMAPR{
    for ($r$ =0; $r \le 20$; $r$++ ){
        if ($0 \le r \le 10$){
            if ($w_i \le 12$){
                UpperPseudoEquation($w_i$, $d_i$, $\Delta$=12)
            }else if ($12 < w_i \le 48$)
                UpperPseudoEquation($w_i$, $d_i$, $\Delta$=36)
            }else if ($48 < w_i \le 96$)
                UpperPseudoEquation($w_i$, $d_i$, $\Delta$=48)
            }else if ($96 < w_i \le 192$)
                UpperPseudoEquation($w_i$, $d_i$, $\Delta$=96)
            }else if ($w_i > 192$)
                $w_i = w_i$ MOD 192
                if ($w_i \le 12$){
                    UpperPseudoEquation($w_i$, $d_i$, $\Delta$=12)
                }else if ($12 < w_i \le 48$)
                    UpperPseudoEquation($w_i$, $d_i$, $\Delta$=36)
                }else if ($48 < w_i \le 96$)
                    UpperPseudoEquation($w_i$, $d_i$, $\Delta$=48)
                }else if ($96 < w_i \le 192$)
                    UpperPseudoEquation($w_i$, $d_i$, $\Delta$=96)
            }
        }
        if ($11 \le r \le 20$){
            if ($w_i \le 12$){
                LowerPseudoEquation($w_i$, $d_i$, $\Delta$=12)
            }else if ($12 < w_i \le 48$)
                LowerPseudoEquation($w_i$, $d_i$, $\Delta$=36)
            }else if ($48 < w_i \le 96$)
                LowerPseudoEquation($w_i$, $d_i$, $\Delta$=48)
            }else if ($96 < w_i \le 192$)
                LowerPseudoEquation($w_i$, $d_i$, $\Delta$=96)
            }else if ($w_i > 192$)
                $w_i = w_i$ MOD 192
                if ($w_i \le 12$){
                    LowerPseudoEquation($w_i$, $d_i$, $\Delta$=12)
                }else if ($12 < w_i \le 48$)
                    LowerPseudoEquation($w_i$, $d_i$, $\Delta$=36)
                }else if ($48 < w_i \le 96$)
                    LowerPseudoEquation($w_i$, $d_i$, $\Delta$=48)
                }else if ($96 < w_i \le 192$)
                    LowerPseudoEquation($w_i$, $d_i$, $\Delta$=96)
            }
        }
    }
}
```

Figure 7.3. Pseudo-code for calculating iMAPR pseudo-cost for all w_i.

7.1.3 Summary of Complete iMAPR Procedure

A full description of the iMAPR process is shown in Figure 7.3 below.

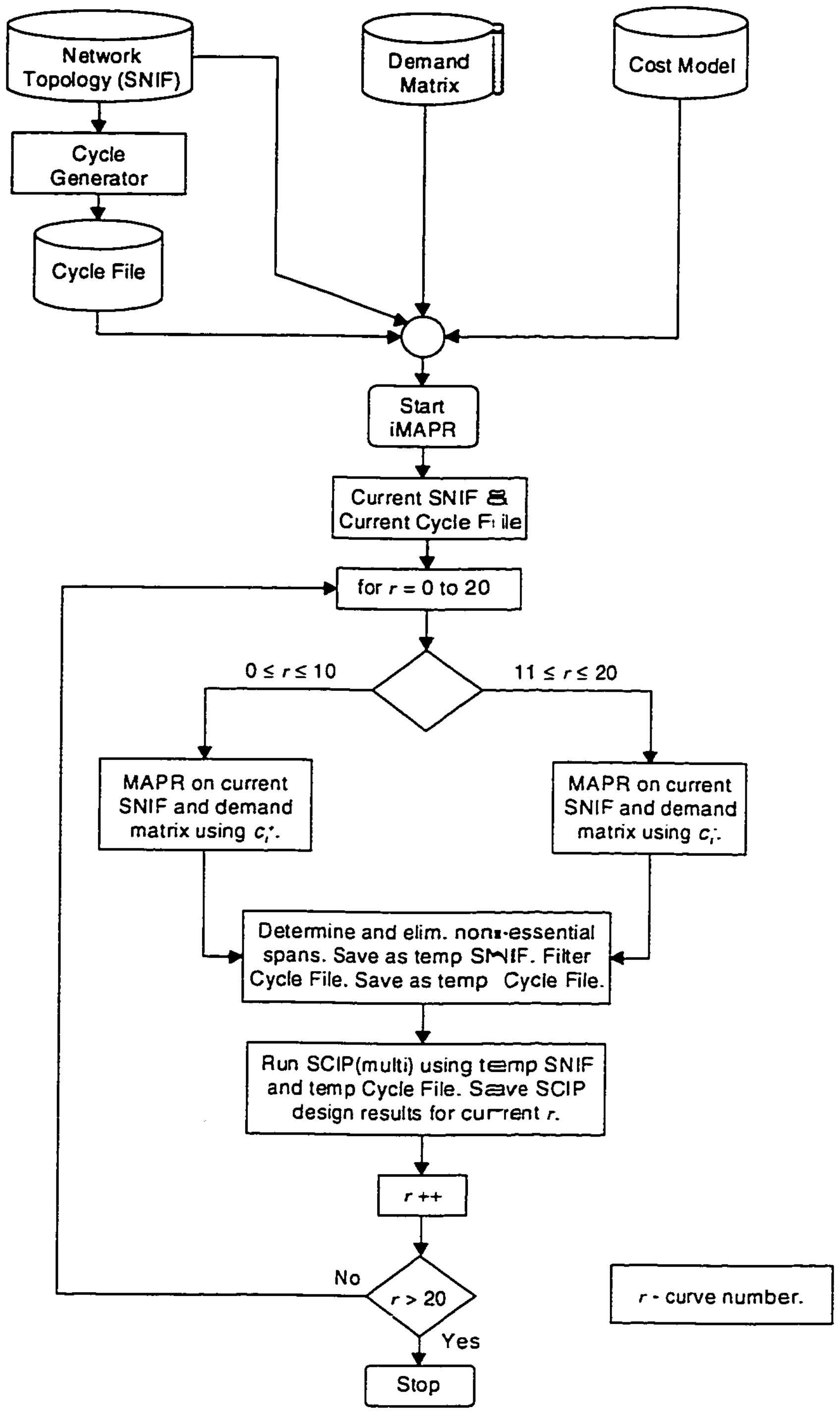

Network Topology (SNIF)
Demand Matrix
Cost Model
Cycle Generator
Cycle File
Start iMAPR
Current SNIF & Current Cycle File
for r = 0 to 20
0 ≤ r ≤ 10
11 ≤ r ≤ 20
MAPR on current SNIF and demand matrix using c_i'.
MAPR on current SNIF and demand matrix using c_i.
Determine and elim. non-essential spans. Save as temp SNIF. Filter Cycle File. Save as temp Cycle File.
Run SCIP(multi) using temp SNIF and temp Cycle File. Save SCIP design results for current r.
r ++
No
r > 20
Yes
Stop
r - curve number.

Figure 7.4. Schematic depiction of iMAPR.

iMAPR begins with four inputs: the SNIF, the demand matrix, the cost model and the cycle file generated by the cycle generator (Chapter 2.3.3). Twenty-one pseudo-cost curves are then stepped through to route the PTP demands. First, the PTP demands are routed using curve number r=0. The $\alpha_\Delta(r)$ for the corresponding r and for $\Delta \in \{12, 38, 48, 96\}$ are shown in Table 7.1. After all the PTP demands have been routed, the non-essential spans with zero working capacity (zero w_i) are determined and eliminated. Similarly, the cycles that use the eliminated non-essential spans are filtered from the cycle file. Using the resulting sub-graph and the filtered cycle file, multi-modular SCIP is executed and the resulting min-cost design is saved. Because SCIP now has four times the number of ring candidates (a set for each modular capacity), the computational time for each SCIP increases. As a practical matter, therefore, and to provide a repeatable basis for comparison, all SCIP run times are limited to 900 sec (an arbitrarily chosen value). Next, r is incremented by one, and the entire iMAPR process is repeated using the full SNIF and the full cycle file until all twenty-one pseudo-curves are stepped through. Here, amongst all the min-cost SCIP designs obtained from the different r curves, the overall least cost design is captured as the most-improved design obtained from using iMAPR.

7.1.4 Discussions and Summary

iMAPR was implemented and investigated as a continuing idea derived from the discussions and conclusions of MAPR in Chapter 6.2.2. Essentially, iMAPR consists of a set of MAPR runs, where each MAPR run uses a different pseudo-cost curve with a systematic variation of aggregation pressure. The idea here is to sweep the pseudo-cost curves and capture the subsequent sub-graph from the MAPR routing and span elimination process that gives the least min-cost (most-improved) ring-coverage design.

The resulting sub-graphs from the routing and elimination process in iMAPR are evaluated using multi-modular SCIP. The computational time for multi-modular SCIP is much higher than that for single-modular SCIP. This tremendous change in computational time prompted the limiting of CPU time for each SCIP run to 900 seconds. Experimental results for iMAPR are deferred to Chapter 9 where they can be presented in a comparative format relative to the next and last two span elimination methods to be considered.

8.0 Post-Inspection and Re-Routing (PIR)

Post-Inspection and Re-Routing (PIR) is a new strategy developed in the second half of this work to resolve the problem of long computational time in IRE. As concluded in Chapter 6, IRE gives good solutions but suffers tremendously in computational time due to the quadratic relationship between the number of SCIP runs and the number of non-essential spans (Equation 5.1). As will be described in this chapter, PIR requires only two SCIP runs, regardless of network size, so computational time required is reduced dramatically. This chapter begins by detailing the motivation, the conceptual idea and the algorithm behind PIR. Later, in Chapter 9, the experimental results obtained using this strategy will be illustrated and compared.

8.1 Introduction

PIR, as the name suggests, is a post-processing span elimination heuristic. The idea behind PIR is similar to that of IRE in the sense that it examines one span elimination at a time seeking the most-improved design cost. However, while IRE uses a greedy approach to choose a non-essential span to eliminate at each step based on an evaluation of cost by SCIP, PIR uses a different cost evaluation process where a complete ring coverage design is not necessary. This process is described later in the chapter. PIR adopts the Breadth-First Search (BFS) strategy to systematically perform the eliminate-detour-evaluate process on every non-essential span. When all the non-essential spans have been examined, it then chooses the sub-graph that most satisfies the given criterion (for IRE, this criterion is least-cost). Using this sub-graph, it then moves down the BFS tree, with one less non-essential span to test, and performs the eliminate-detour-evaluate process again using the remaining non-essential span set. Unlike IRE however, PIR does not necessarily continue progressing through the tree until all the elements in the set of non-essential spans has been eliminated. PIR halts when a special condition is met. This special condition is described later in the chapter.

To aid in understanding PER, the factors that limited the performance of IRE are first illustrated. As described in previous chapters, IRE's computational time suffers due to the large number of SCIP runs. In addition, further investigations of IRE show that there are situations where eliminating non-essential spans does not reduce the SCIP design cost, for instance, where the SCIP design for the network with no eliminations is the preferred most-improved min-cost design. For example, consider the network and span working capacity requirements resulting from shortest-path routing in Figure 8.1. Using IRE with multi-modular SCIP, the following sub-graphs and SCIP designs were obtained as shown in Figure 8.2.

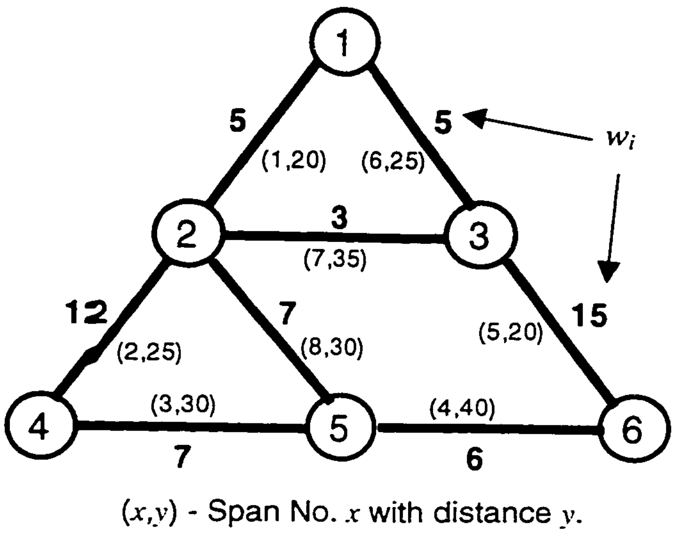
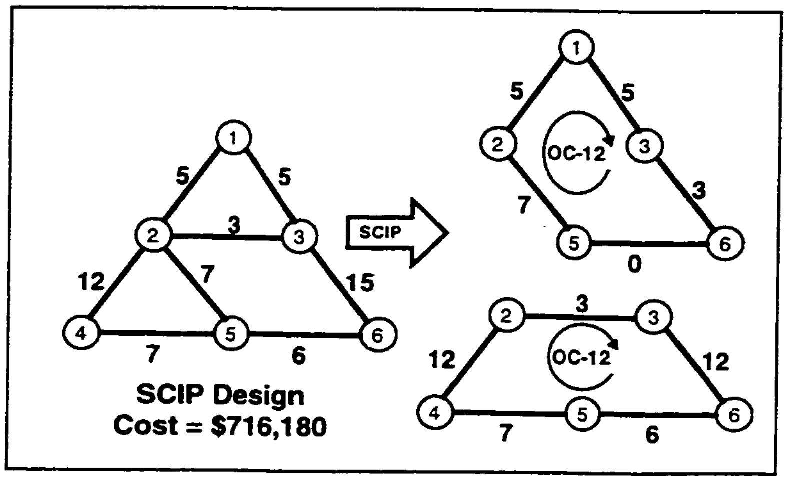
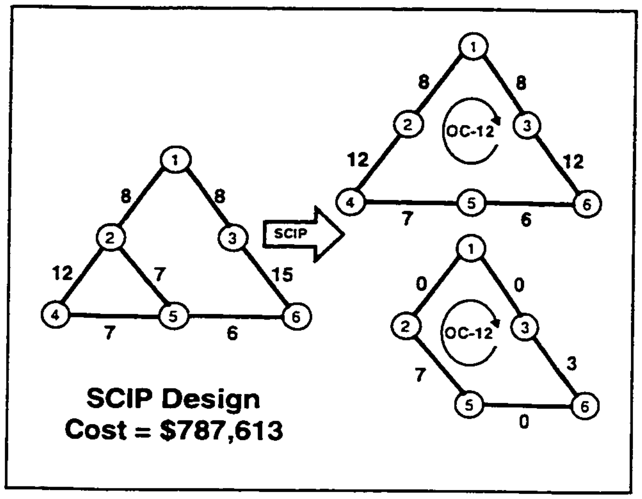


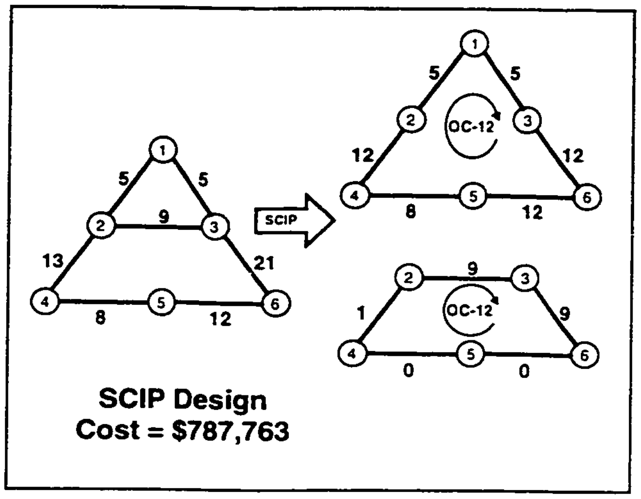
Figure 8.1. **A network example where demands are routed using pure shortest-path.**



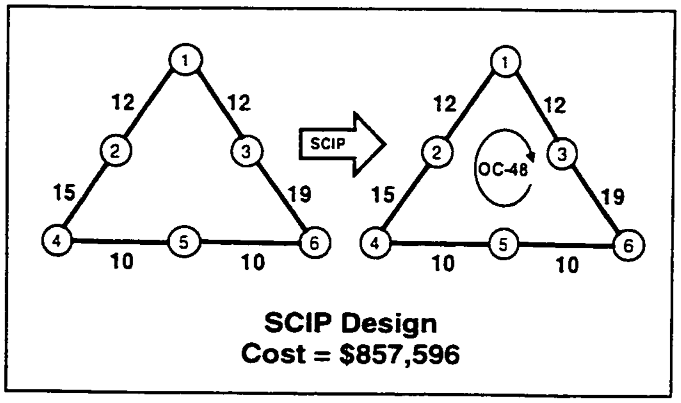
(a) SCIP on the Full Network.



(b) 1st iteration: Test elimination of span (2-3).



(c) 1st iteration: Test elimination of span (2-5).



(d) 2nd iteration: Test elimination of span (2-5).

Figure 8.2. IRE results for the network in Figure 8.1 using multi-modular SCIP. (a) The full network graph with demands routed using pure shortest-path. (b) Span (2-3) is eliminated. (c) Span (2-5) is eliminated. (d) Following the SCIP design in (b), span (2-5) is eliminated.

Clearly, the most-improved SCIP design cost obtained by IRE was the initial graph shown in Figure 8.2a with no span eliminations and a total SCIP design cost of $716,180 (using the same cost model as presented in Table 3.1). In other words this is a case where the resulting SCIP design cost did not benefit from any IRE span elimination. By observing the step-by-step span elimination process used by IRE, it can be seen that, due to the working capacity requirements on span (3-6), the best method of covering the network is to use two OC-12 rings where these rings are restricted to covering span (3-6). The remaining three units of excess working capacity (15 MOD 12) on span (3-6) have forced the coverage from the additional OC-12 ring. Using an OC-48 ring will not be cost-effective here since it was found, in this example, to cost more than using two OC-12 rings. The important role of span (3-6) is realized because it has three more units of capacity than the ring module size, when OC-12 is being considered. Thus, we could say that span (3-6) has a *supermodularity* of three with respect to OC-12. In this example, the existence of supermodular span (3-6) in the network, and because span (3-6) is also an essential span, has restricted the ability of span eliminations in IRE to reduce cost.

Supermodular spans are defined as spans where the accumulated w_i is greater than the largest modular ring capacity placed on the span with respect to the reference rings first placed in a basic SCIP run on the network using shortest-path and no span eliminations. For example, if a span has a working capacity requirement of 52 and is assumed to be covered by an OC-48 ring and an OC-12 ring, this span is a supermodular span with a supermodular capacity of four (52 MOD 48). However, if the span working capacity requirement is 40 and is assumed to be covered by an OC-48 and an OC-12 ring, this span is not a supermodular span since the span has fewer working capacity requirements than the largest modular ring capacity (OC-48) placed on the span. Hence, a span is *not* a supermodular span if it has a working capacity requirement that is less than or equal to the largest modular ring capacity that is placed on the span.

The idea in PIR is to harness the knowledge of supermodular spans. When supermodular spans are determined, the supermodular capacities of these spans are reconfigured by re-routing so as to reduce the number of supermodular spans. To understand the concept of PIR, first it must be determined why a reduced number of supermodular spans may lead to a better cost result from SCIP. Take Figure 8.1 as an example: from the set of rings (Figure 8.2a) obtained from SCIP on the network using shortest-path and no span eliminations, through an evaluation, it is found that span (3-6) is the only supermodular span in the network with three units of supermodular capacity (15 MOD 12). Here, three units of supermodular capacity are re-routed by detouring them using an MAPR(rise)-like routing at the span-level (as described in Figure 2.2). As a result, the three units of extra working capacity are detoured through path (3-2)-(2-5)-(5-6). MAPR(rise)-like re-routing is useful here to ensure that detouring of supermodular capacities will not bump ring capacities over their modular boundaries, triggering additional rings, unless necessary. After the supermodular capacities have been re-routed, the resulting graph is then evaluated again for the number of supermodular spans with respect to the same reference set of rings placed initially on the network with shortest-path and no span eliminations. Figure 8.3 illustrates the resulting network after the supermodular capacities have been re-routed.

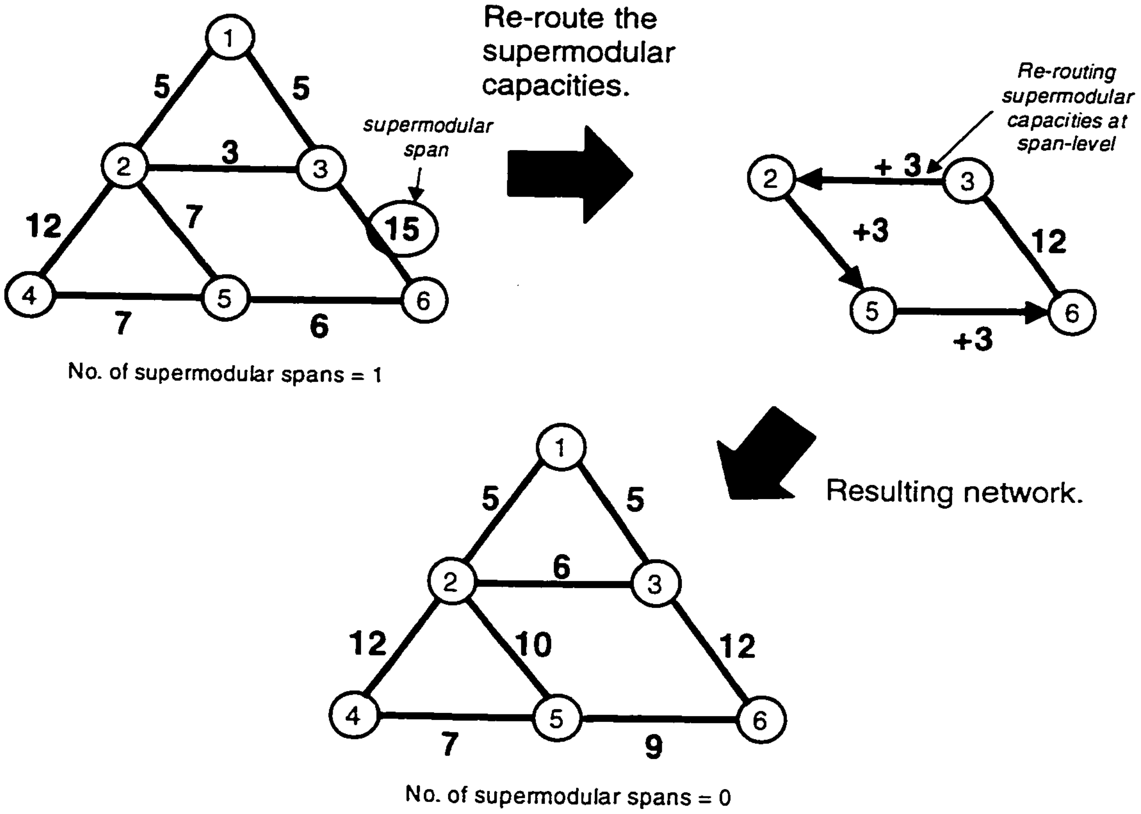

Re-route the supermodular capacities.
supermodular span
No. of supermodular spans = 1
Re-routing supermodular capacities at span-level
+ 3
+3
+3
12
Resulting network.
No. of supermodular spans = 0

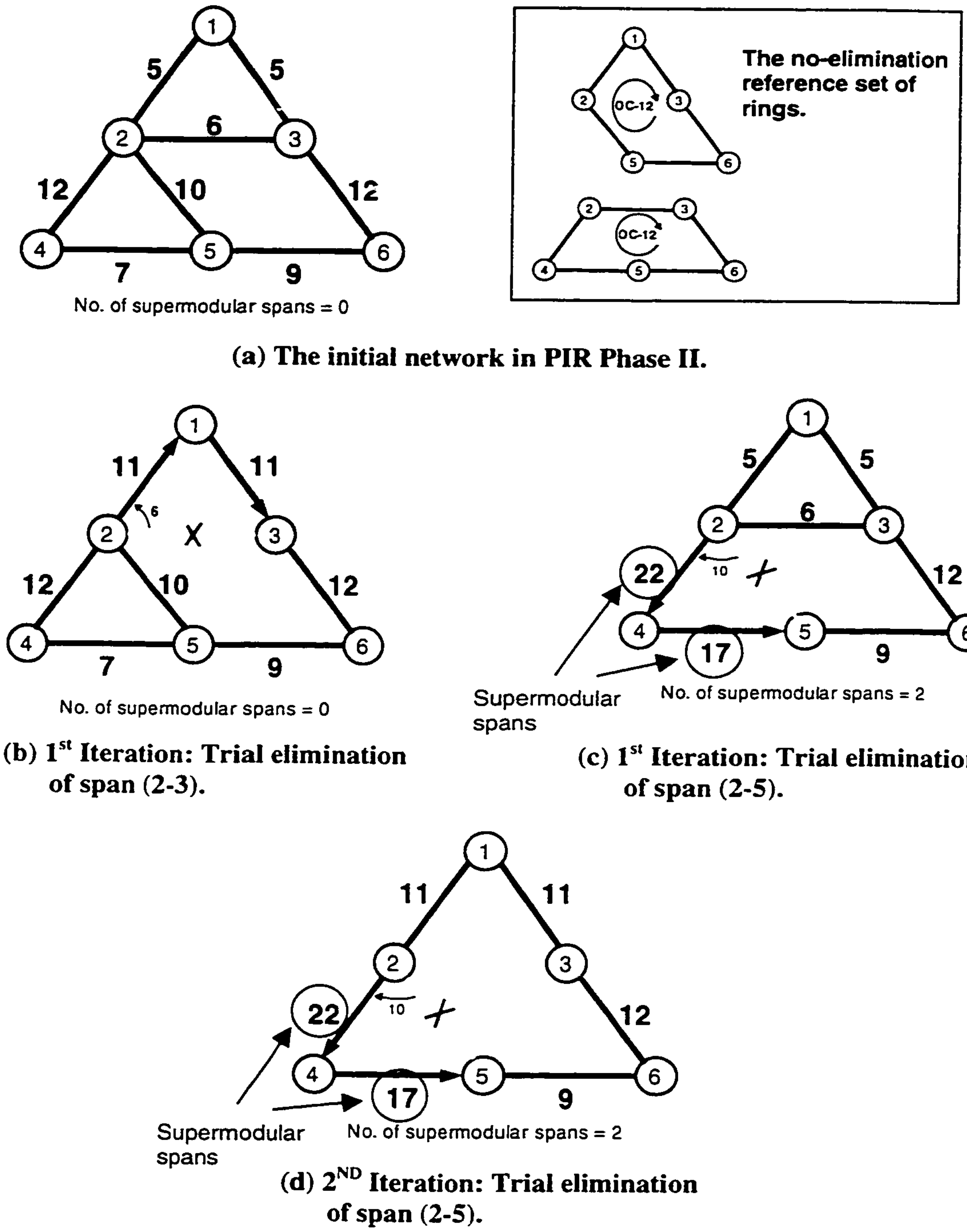
Figure 8.3. **Phase I of PIR where the supermodular capacities of span (3-6) have been re-routed.**

After the supermodular capacity has been re-routed, and since the resulting graph has fewer numbers of supermodular spans than it initially had, the resulting graph in Figure 8.3 is permanently adopted. A brief evaluation shows that there are no subsequent supermodular spans left, therefore this PIR Phase I is complete.

In the second phase, PIR further examines the possibility of reducing the number of supermodular spans by means of eliminating the non-essential spans. This is accomplished by initiating the greedy Breadth-First-Search (BFS) strategy to search for the combination of non-essential span eliminations that permits the lowest number of supermodular spans. This step is similar to IRE except for the polynomial-time evaluation procedure used at the end of each span elimination and re-routing. The evaluation procedure in PIR considers the number of supermodular spans (with respect to the reference set of rings placed on the network earlier using shortest-path and no span eliminations) in the resulting sub-graph after span elimination and re-routing of the affected demands. The fewer the number of supermodular spans, the better the sub-graph is presumed to be for a final SCIP design run. Therefore, at each depth of the BFS tree, a sub-graph with the least number of supermodular spans is kept. If there is more than one sub-graph with the same least number of supermodular spans, one is chosen arbitrarily from amongst them. This PIR Phase II halts when moving further down the BFS tree does not result in an equal or lower number of supermodular spans.

An example of PIR Phase II is shown in Figure 8.4 where the BFS strategy is initiated to find the lowest number of supermodular spans by means of eliminating the non-essential spans and re-routing the affected demands on the network shown in Figure 8.3. The network after PIR Phase I in Figure 8.3 has no supermodular spans. This network is the root of the BFS tree in PIR Phase II. There are two non-essential spans, span (2-3) and span (2-5). At the first depth of the tree, eliminate-reroute of span (2-3) results in zero supermodular spans and eliminate-reroute of span (2-5) results in two supermodular spans. The supermodular spans are determined with respect to the reference rings placed on the network earlier using shortest-path and no span eliminations (Figure 8.2a). The resulting sub-graph due to the elimination-reroute of span (2-3) is kept for this iteration since it gives a lower number of supermodular spans than the other sub-graph, and this number is less than or equal to the number of supermodular spans before the span eliminations process was executed.

Going down to the second depth of the tree, one non-essential span is left. Here, eliminate-reroute of span (2-5) results in two supermodular spans. Consequently, PIR halts here since there are no eliminations of non-essential spans that can result in an equal or lower number of supermodular spans. Note that it is possible that eliminating a span can create a supermodular span but eliminating the next span can remove it. This effect is desired since in total two spans are eliminated without increasing the number of supermodular spans. However, for simplicity's sake, this factor is not taken into consideration at this point.



No. of supermodular spans = 0
The no-elimination reference set of rings.
OC-12
(a) The initial network in PIR Phase II.
No. of supermodular spans = 0
(b) 1st Iteration: Trial elimination of span (2-3).
Supermodular spans
No. of supermodular spans = 2
(c) 1st Iteration: Trial elimination of span (2-5).
Supermodular spans
No. of supermodular spans = 2
(d) 2ND Iteration: Trial elimination of span (2-5).

Figure 8.4. PIR Phase II results following PIR Phase I in Figure 8.3. (a) Resulting network from PIR phase one. (b) Span (2-3) is eliminated. (c) Span (2-5) is eliminated. (d) Following the elimination of span (2-3) in (b), span (2-5) is eliminated.

The most-improved sub-graph found by PIR is the sub-graph that gives the fewest possible supermodular spans (Figure 8.4b). By executing SCIP on this sub-graph (Figure 8.5), it can be seen that the SCIP design cost is $572,864. Comparing this cost result with the most-improved cost result obtained by IRE (Figure 8.2a), a 20% reduction is observed.

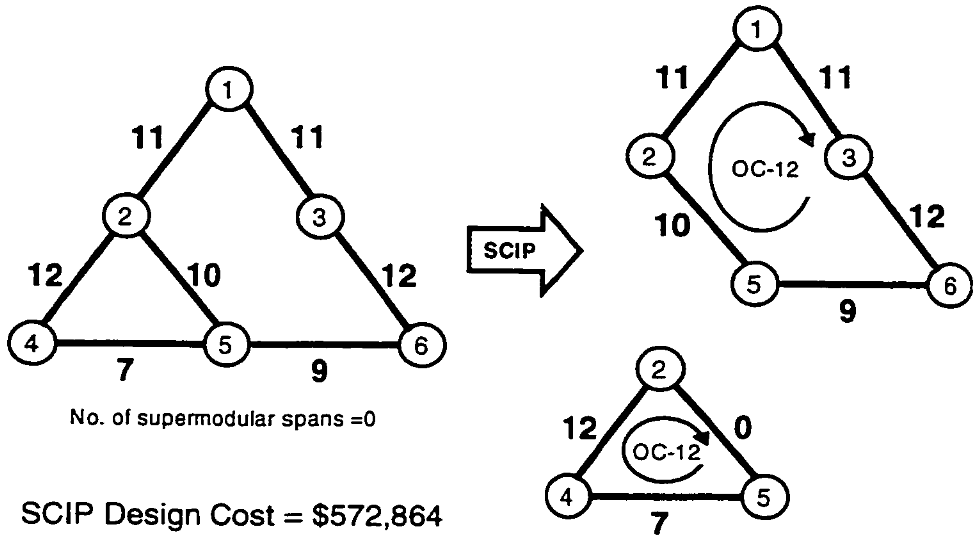
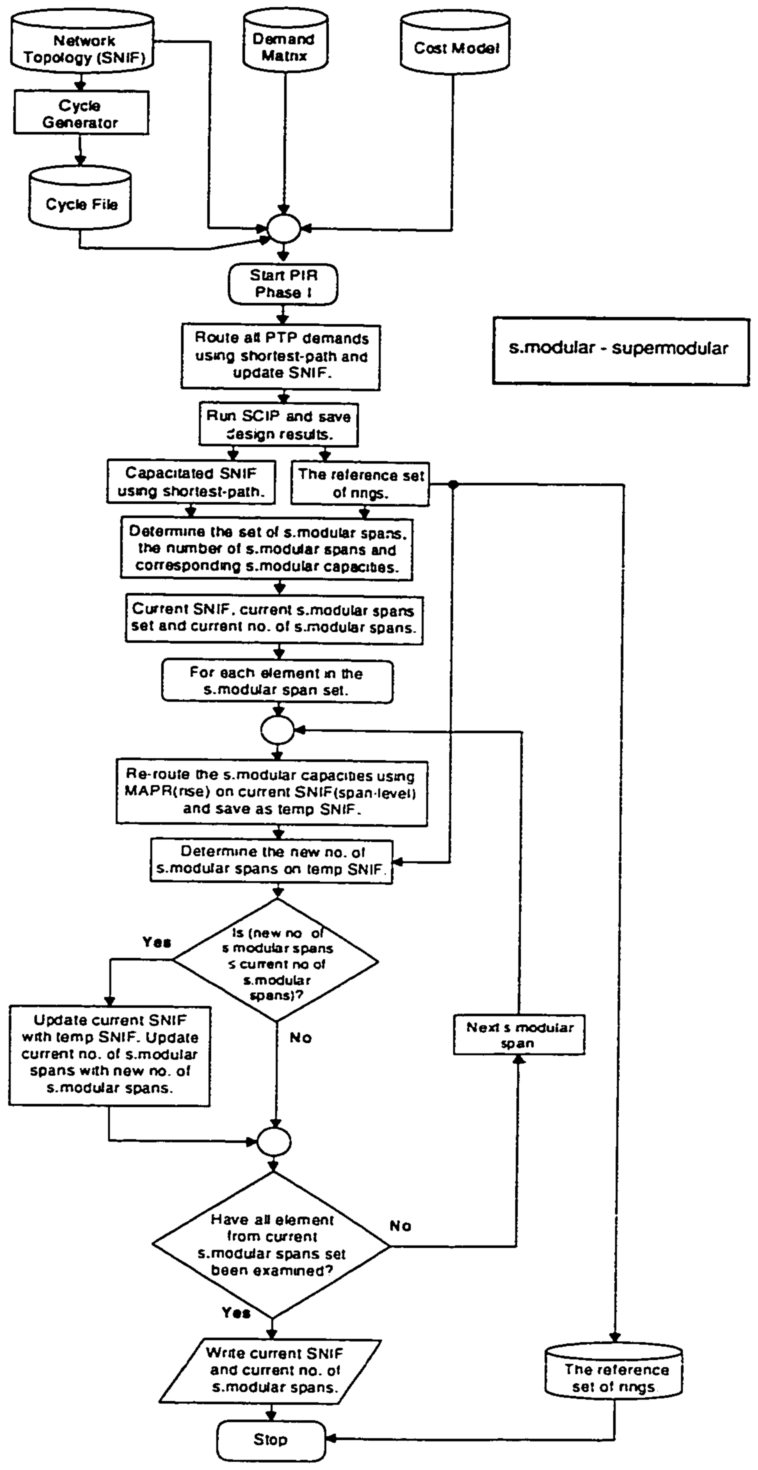


Figure 8.5. SCIP design results on the most-improved sub-graph found by PIR.

The example shown above using PIR shows that reducing the number of supermodular spans by means of re-routing the supermodular capacities and eliminating the non-essential spans does in fact give a better SCIP design cost. Furthermore, the major advantage in PIR is that the overall algorithm uses only two SCIP runs, regardless of the network size, enabling it to give the most-improved solution in a relatively fast computational time. Next, the overall PIR algorithm is described.

8.2 Complete PIR Algorithm

The fully detailed description of the PIR process is shown in Figure 8.6 and Figure 8.7.



Network Topology (SNIF)
Demand Matrix
Cost Model
Cycle Generator
Cycle File
Start PIR Phase I
Route all PTP demands using shortest-path and update SNIF.
s.modular - supermodular
Run SCIP and save design results.
Capacitated SNIF using shortest-path.
The reference set of nngs.
Determine the set of s.modular spans, the number of s.modular spans and corresponding s.modular capacities.
Current SNIF, current s.modular spans set and current no. of s.modular spans.
For each element in the s.modular span set.
Re-route the s.modular capacities using MAPR(rise) on current SNIF(span-level) and save as temp SNIF.
Determine the new no. of s.modular spans on temp SNIF.
Is (new no of s.modular spans ≤ current no of s.modular spans)?
Yes
No
Update current SNIF with temp SNIF. Update current no. of s.modular spans with new no. of s.modular spans.
Next s.modular span
Have all element from current s.modular spans set been examined?
No
Yes
Write current SNIF and current no. of s.modular spans.
The reference set of nngs
Stop

Figure 8.6. **Phase I of PIR showing the processes that are involved in reducing the total number of forcer spans by means of re-routing the supermodular capacities.**

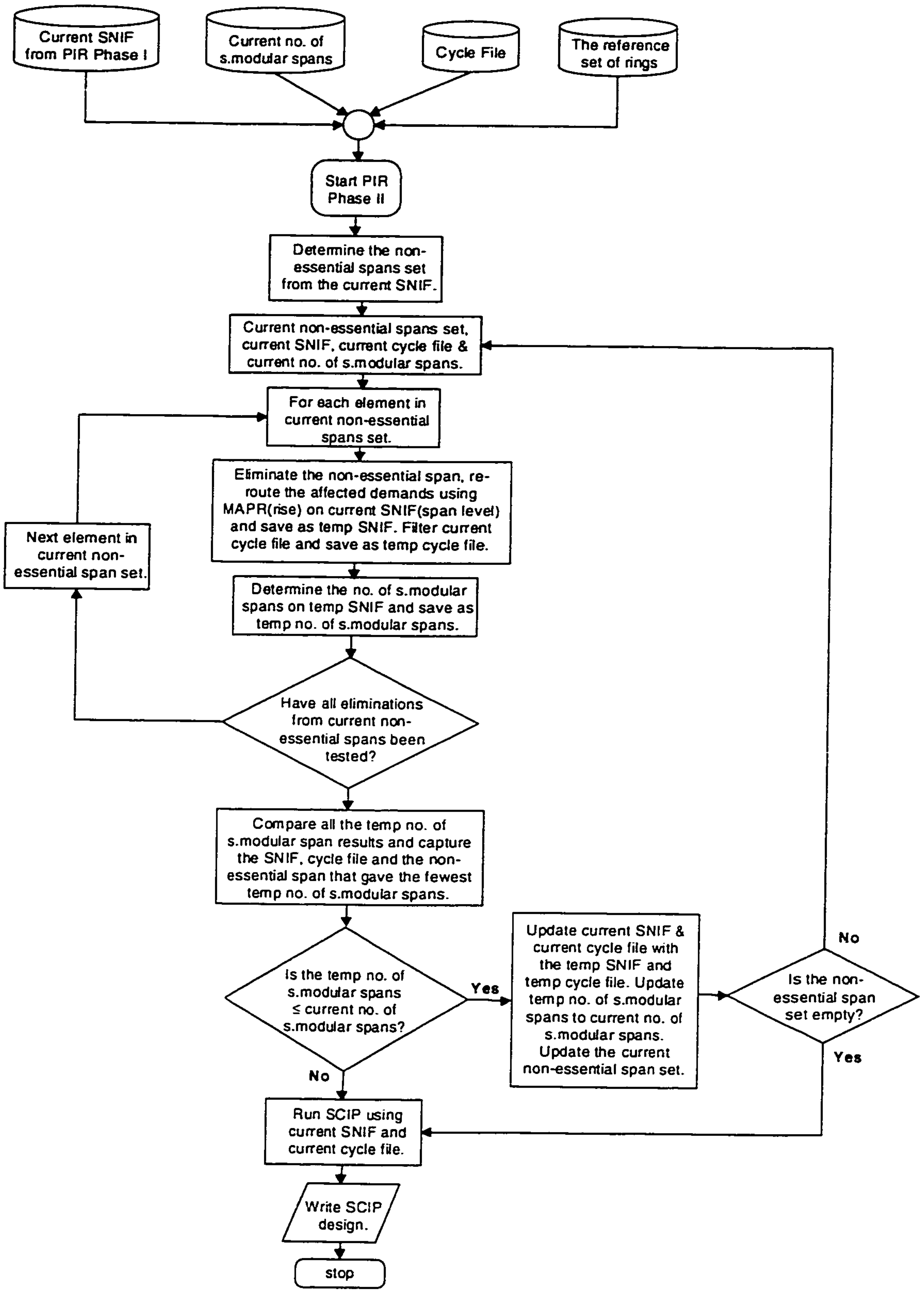

Current SNIF from PIR Phase I
Current no. of s.modular spans
Cycle File
The reference set of rings
Start PIR Phase II
Determine the non-essential spans set from the current SNIF.
Current non-essential spans set, current SNIF, current cycle file & current no. of s.modular spans.
For each element in current non-essential spans set.
Eliminate the non-essential span, re-route the affected demands using MAPR(rise) on current SNIF(span level) and save as temp SNIF. Filter current cycle file and save as temp cycle file.
Next element in current non-essential span set.
Determine the no. of s.modular spans on temp SNIF and save as temp no. of s.modular spans.
Have all eliminations from current non-essential spans been tested?
Compare all the temp no. of s.modular span results and capture the SNIF, cycle file and the non-essential span that gave the fewest temp no. of s.modular spans.
Is the temp no. of s.modular spans ≤ current no. of s.modular spans?
Yes
No
Update current SNIF & current cycle file with the temp SNIF and temp cycle file. Update temp no. of s.modular spans to current no. of s.modular spans. Update the current non-essential span set.
Is the non-essential span set empty?
No
Yes
Run SCIP using current SNIF and current cycle file.
Write SCIP design.
stop

Figure 8.7. **Phase II of PIR: Search for the minimum number of supermodular spans by eliminating non-essential spans and re-routing trials.**

The PIR algorithm has two phases. The objective in Phase I is to reduce the number of supermodular spans by means of re-routing the supermodular capacities. Phase II implements span eliminations using the BFS strategy to search for a set of span eliminations and re-routing that will give the minimum number of supermodular spans.

Phase I begins with four inputs: network SNIF, demand matrix, cycle file and cost model. First, the PTP demands are routed over the network SNIF using pure shortest-path and SCIP is executed on the subsequent capacitated network SNIF to obtain the no-elimination reference set of rings to cover the network graph. With this reference set of rings, the set of supermodular spans and the corresponding supermodular capacities can be determined.

For example, to determine the set of supermodular spans for the network in Figure 8.1, the rings found by SCIP shown in Figure 8.2a are utilized. Using this as the no-elimination reference set of rings, each span in the network is investigated. Specifically, the number and modularity size of the rings that are used to cover each span are considered. Here, all spans in the network in Figure 8.1 are covered by a single OC-12 span module except for span (5-6) and span (3-6), where two OC-12 span modules are used. In detecting supermodular capacities a span that is covered by a single span module is ignored since it has no supermodular capacities and, therefore, is not a supermodular span. For the spans that require multiple overlying modules, its capacity requirements w_i are divided by the largest span modular capacity used to cover this span, and then it is observed to determine whether there are any supermodular capacities; the remainder of w_i divide by the largest span modular capacity is the supermodular capacity. If there are supermodular capacities, this span is supermodular with respect to the no-elimination reference set of rings. As such, span (5-6) is not a supermodular span and span (3-6) is a supermodular span with three supermodular capacities. After the set of supermodular spans, the number of supermodular spans and the supermodular capacities have been determined, a *for* loop is entered with the current SNIF, the current set of supermodular spans, the supermodular capacities, and the current number of supermodular spans.

The first element in the set of supermodular spans and the corresponding supermodular capacities are re-routed at the span-level using an MAPR(rise)-like re-routing. This re-routing is performed at the span-level. The subsequent graph is analyzed and the number of supermodular spans it has is determined with respect to the no-elimination reference set of rings. If the number of supermodular spans is less than or equal to the current number of supermodular spans, the changes to the current SNIF are committed to and the number of supermodular spans with the reduced number of supermodular spans is updated. If not, the process is repeated again with the next element in the set of supermodular spans until all the supermodular spans have been examined. As a result, the network that has the lowest possible number of supermodular spans is found from re-routing the supermodular capacities. Phase II is then entered using the SNIF, which has the smallest number of supermodular spans from Phase I, the number of supermodular spans, and the no-elimination reference set of rings as inputs.

Phase II begins by determining the non-essential spans as per all prior methods. Using this set of non-essential spans, the current network SNIF, the current number of supermodular spans, the current cycle file, and the no-elimination reference set of rings, the BFS iterations begin. For each non-essential span, a non-essential span is eliminated from the network SNIF, the affected demands are re-routed at the span-level using MAPR(rise)-like re-routing, the cycles that use the eliminated non-essential span are filtered, and the number of supermodular spans is found following the elimination and re-routing with respect to the no-elimination reference set of rings. The resulting sub-graph is saved as a temporary SNIF together with the temporary cycle file and temporary number of supermodular spans. The above process is repeated with another non-essential span on the current network SNIF and the current cycle file until all the elements in the non-essential span set have been examined. Then, the trial elimination and re-routing that gives the smallest number of supermodular spans is kept. If there is more than one sub-graph with the same lowest number of supermodular spans, one is arbitrarily chosen.

At each iteration, the number of supermodular spans in the trial case is compared with the current minimum number of supermodular spans. If the number of supermodular spans is less than or equal to the current minimum number of supermodular spans, the number of supermodular spans is determined as the minimum number of supermodular spans. However, if the number of supermodular spans is more than the current minimum number of supermodular spans, the BFS strategy is halted where SCIP is then executed on the current SNIF using the current cycle file. This new SCIP design is compared to the SCIP design obtained earlier in Phase I. The SCIP design that gives the lowest cost design is kept, and will be the most-improved SCIP design obtained using the PIR strategy together with the number of span eliminations and the number of supermodular spans.

8.3 Summary

The PIR strategy was developed in response to the problem of excessive run-time sustained by IRE due to internal SCIP calls. PIR is based on a polynomial time analysis of supermodular capacities as a means of detecting the desired span eliminations. The hypothesis is that a reduced number of supermodular spans will lead to a lower ring network design cost in a single final SCIP coverage run. This is an intuitive motive, which will later be confirmed by experimental results shown in Chapter 9, verifying that PIR is an efficient heuristic.

A major advantage of PIR is that it requires only two SCIP runs regardless of network size. The first SCIP run is part of PIR Phase I to obtain the ring coverage design of the initial network graph where the demands are routed using the pure shortest-path algorithm. The ring-set thus obtained provides the reference modularities against which to later define and detect supermodular spans and capacities used in the PIR strategy. The second SCIP run is performed at the end of PIR Phase II to obtain the final ring coverage design on the span eliminated and the re-routed sub-graph that yields the minimum number of supermodular spans. Among the two SCIP designs, the SCIP design that gives the least-cost will be the most-improved SCIP design found using PIR. Since only two SCIP runs are required, PIR is expected to be much faster than IRE.

9.0 Performance and Results

This chapter reports the experimental results obtained using iMAPR, IRE and PIR. At this stage all methods and results are multi-modular and, in any case where SCIP is used, each SCIP run itself was given at a maximum of 900 seconds to run. These span elimination strategies are now compared in terms of their solution quality and computational time.

9.1 Results of Pure Shortest-Path Routing using Multi-Modular Capacity SCIP - Revised

To have consistent SCIP runtimes for all three span elimination strategies (iMAPR, IRE and PIR), the experiment for a network using pure shortest-path in Chapter 3.3.2 has been revised using SCIP limited to 900 seconds of CPU runtime. Consequently, because of this and the switch to multi-modular designs, the baseline results for SCIP coverage designs employing no span eliminations need to be updated.

Table 9.1. Min-cost results obtained for network using shortest-path routing and SCIP multi-modular capacity design with SCIP's CPU runtime limited to 900 seconds compared to CPU runtime limited to 24-hours.

Network	SCIP Design Cost (million) 900-sec	SCIP Design Cost (million) 24-hour	% Difference
A	1.94	1.94	0.00%
B	17.72	16.73	5.61%
C	11.08	11.08	0.00%
D	27.21	26.85	1.33%

Comparing the results in Table 9.1 and the results obtained in Table 3.4, the difference in design costs between SCIP limited to 24-hour CPU runtime and SCIP limited to 900-sec CPU runtime is in the range of 0% to 6%; the 900-sec limitation has not significantly altered the baseline results against which performance of the various span elimination methods will be assessed.

9.2 Results and Analysis of iMAPR

The same four test networks and demand matrices detailed in Chapter 2.4 have been used to test iMAPR. Here, each of these networks is examined using multi-modular design with four different modular capacities, i.e., OC-12, OC-48, OC-96 and OC-192. The ring coverage designs for all the resulting sub-graphs are generated by SCIP using the network cost model described in Chapter 3.2.3.

Table 9.2 through Table 9.5 show the results obtained from SCIP for Net A, Net B, Net C and Net D, respectively, following the span eliminations using iMAPR. The *Curve Number* column represents the *r* value that was used in the pseudo-cost function (refer to Equations 7.6 and 7.7). The resulting *Number of Span Eliminations* is shown in the next column. The *SCIP Design Cost* column is the min-cost ring coverage design obtained using the multi-modular SCIP of the respective network with the corresponding span eliminations. The overall min-cost result is shown by the (*) in the *SCIP Design Cost* column. Table 9.6 summarizes the most-improved results obtained using iMAPR for each test network.

Table 9.2. iMAPR results for Net A.

Network	Curve Number, *r*	No. of Span Eliminations	SCIP Design Cost (Million)	Comp. Time (seconds)
Net A	0	3	1.93	33.1
	1	3	1.93	
	2	3	1.79	
	3	5	1.72	
	4	6	1.72	
	5	6	1.93	
	6	7	2.00	
	7	6	1.72	
	8	6	1.86	
	9	6	1.86	
	10	6	1.93	
	11	7	1.86	
	12	7	1.93	
	13	8	1.92	
	14	8	2.07	
	15	9	1.93	
	16	9	1.79	
	17	9	2.00	
	18	9	1.65 *	
	19	10	1.86	
	20	10	1.86	

Table 9.3. iMAPR results for Net B.

Network	Curve Number, *r*	No. of Span Eliminations	SCIP Design Cost (Million)	Comp. Time (seconds)
Net B	0	0	17.65	19,438.40
	1	0	16.87	
	2	0	17.82	
	3	0	17.23	
	4	0	17.09	
	5	0	17.66	
	6	0	16.70	
	7	0	16.46	
	8	0	16.42	
	9	0	18.97	
	10	1	16.09 *	
	11	0	16.66	
	12	2	17.15	
	13	2	16.79	
	14	2	17.37	
	15	4	16.13	
	16	2	17.53	
	17	3	18.40	
	18	4	18.76	
	19	6	19.01	
	20	2	19.22	

Table 9.4. iMAPR results for Net C.

Network	Curve Number, *r*	No. of Span Eliminations	SCIP Design Cost (Million)	Comp. Time (seconds)
Net C	0	2	11.15	214.15
	1	2	11.15	
	2	2	11.15	
	3	2	11.15	
	4	2	11.58	
	5	2	11.15	
	6	2	11.15	
	7	2	11.72	
	8	2	11.72	
	9	2	11.58	
	10	2	11.58	
	11	2	11.58	
	12	2	11.58	
	13	2	11.63	
	14	2	11.56	
	15	2	10.99 *	
	16	3	11.99	
	17	4	11.94	
	18	4	11.94	
	19	4	11.94	
	20	4	12.52	

Table 9.5. iMAPR results for Net D.

Network	Curve Number, *r*	No. of Span Eliminations	SCIP Design Cost (Million)	Comp. Time (seconds)
Net D	0	6	29.70	17,347.29
	1	6	29.18	
	2	6	29.02	
	3	6	28.80	
	4	6	29.22	
	5	6	28.69	
	6	6	28.22 *	
	7	6	28.93	
	8	6	28.93	
	9	6	28.40	
	10	6	29.12	
	11	8	30.80	
	12	8	30.26	
	13	8	31.16	
	14	9	32.97	
	15	9	32.90	
	16	9	34.00	
	17	8	32.59	
	18	8	34.42	
	19	8	33.90	
	20	8	35.33	

Table 9.6. **The most-improved SCIP results for each test network using iMAPR.**

Network	iMAPR				% Improvement
	Curve Number, *r*	No. of Span Eliminations	SCIP Design Cost (Million)	Comp. Time (seconds)	
A	18	9	1.65	33.10	14.93%
B	10	1	16.09	19,438.40	9.24%
C	15	2	10.99	214.15	0.78%
D	6	6	28.22	17,347.29	-3.73%

As expected, the results in Table 9.2 to 9.5 show that as the pseudo-cost curves are swept from the lowest aggregation pressure (r=0) to the highest (r=20), the number of span eliminations increases. However, the corresponding SCIP design costs do not appear to follow any specific pattern; in general, they vary with r. As for the computational time, in Table 9.6, it can be seen that Net B and Net D require a relatively longer computational times as comparea to Net A and Net C where relatively short computational times were observed. The *% improvement* shows the percentage improvement obtained from the most-improved SCIP design results over the corresponding design using shortest-path and no span eliminations (Table 9.1).

9.3 Results of IRE using Multi-Modular SCIP

A similar experimental setup was used to investigate IRE, this time, however, embedding multi-modular SCIP. Figure 9.1 to Figure 9.4 show the results of using IRE(multi) on Net A, Net B, Net C and Net D, respectively. The eliminated spans for each IRE iteration number are displayed adjacent to each point shown on the graphs. The non-essential span set found for each network is the same as the non-essential span set shown in Table 6.1.

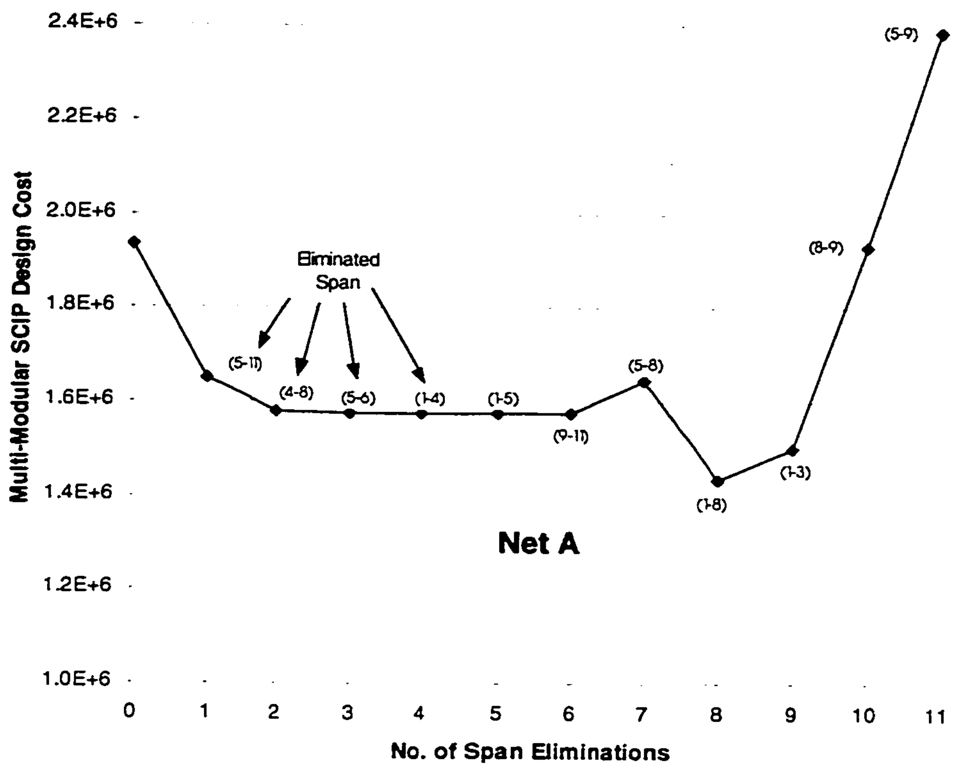


Figure 9.1. SCIP results for Net A using IRE(multi).

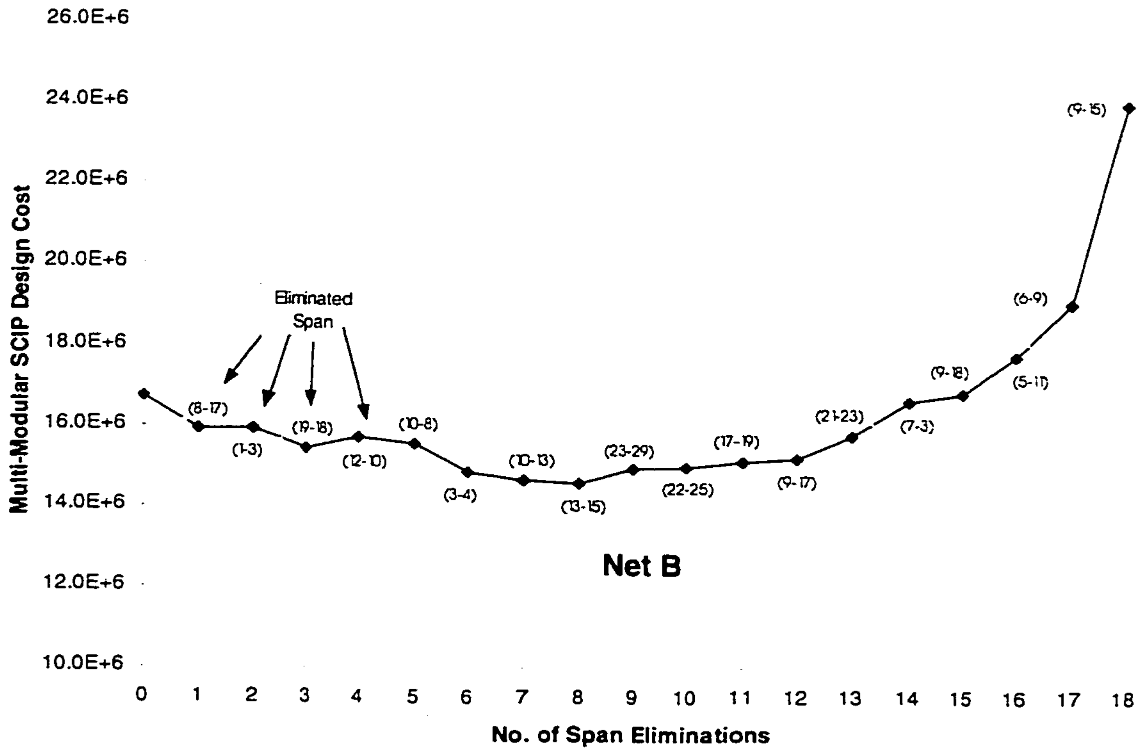


Figure 9.2. SCIP results for Net B using IRE(multi).

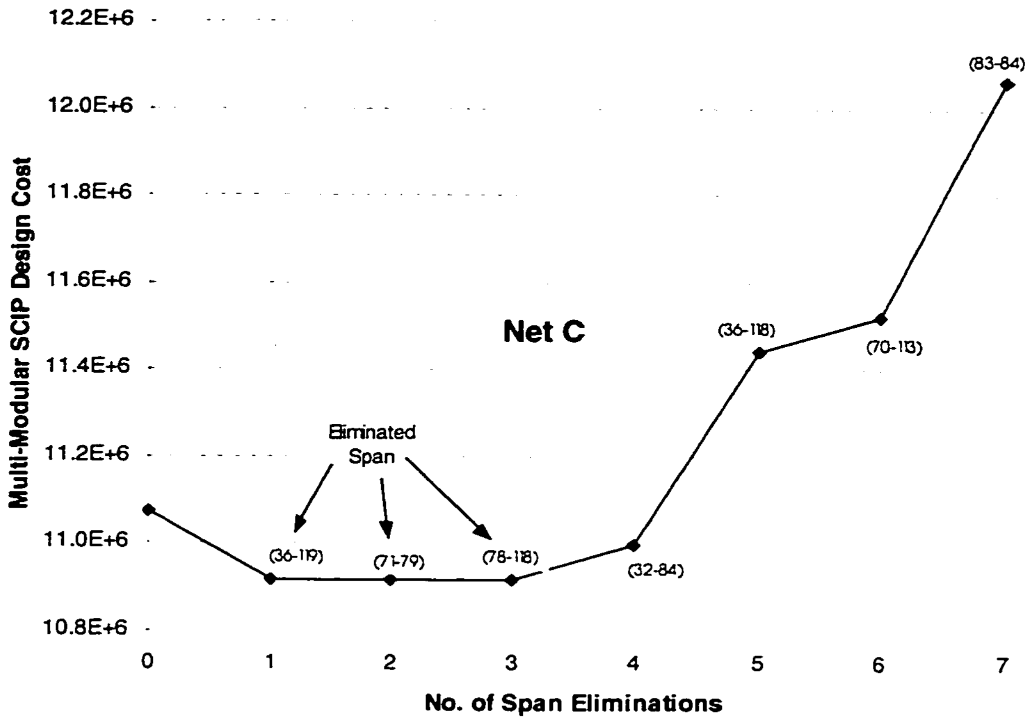


Figure 9.3. SCIP results for Net C using IRE(multi).

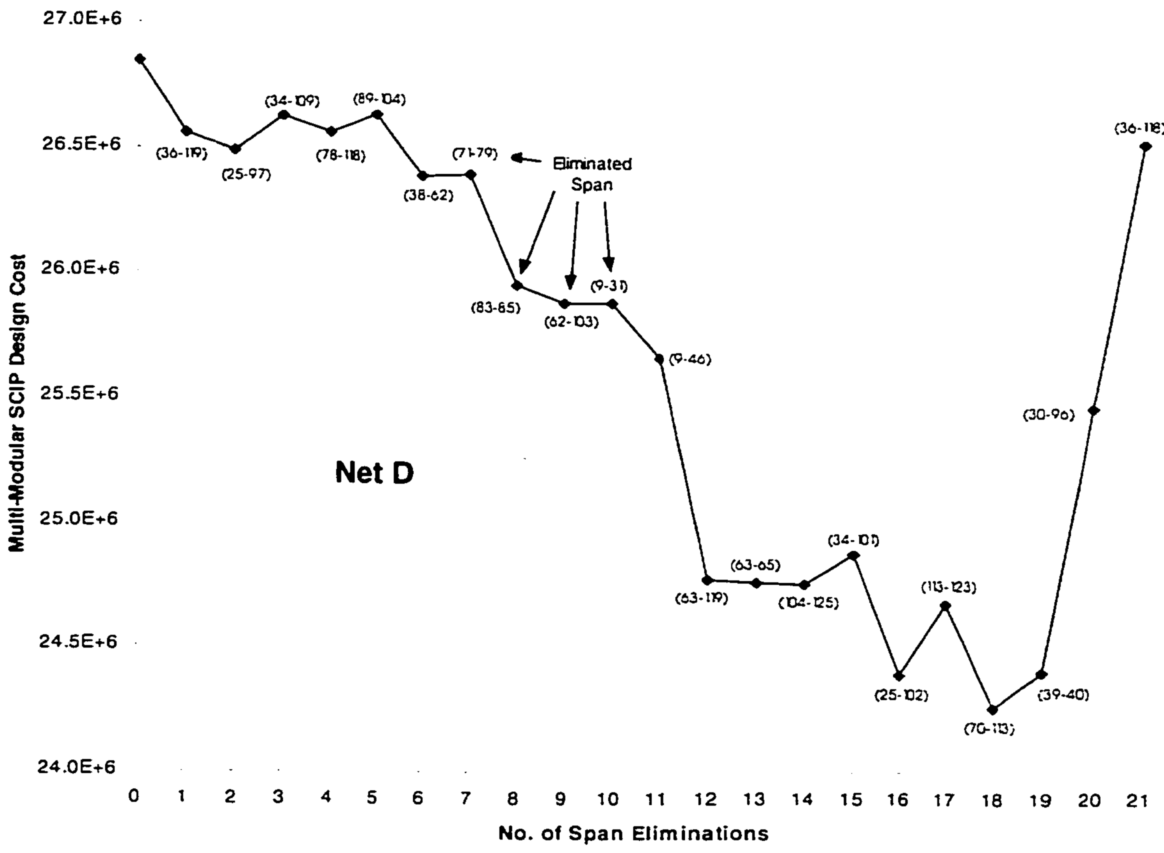


Figure 9.4. SCIP results for Net D using IRE(multi).

Examining the figures above, it can be seen that a similar pattern occurs for all the IRE(multi) curves resembling the curves obtained for IRE(single) in Chapter 6.1.2. The first falling section of the curves occurs when too many rings emerge in the design forced by the need to cover all spans. The lightly loaded rings that result are reduced by eliminating spans and detouring the affected demands, sending them over longer alternate paths via other rings. Thus, cost decreases as more spans are eliminated and ring utilization increases. A minimum value (most-improved design) is reached when the rings and spans achieve the best possible utilization. When yet more spans are removed, demands are diverted over longer and longer paths, which eventually drives up the capacity requirements until the ring capacity is exceeded and additional rings are triggered. Table 9.7 below summarizes the most-improved results obtained from using IRE(multi). The % *improvement* shows the percentage improvement obtained from the most-improved SCIP design results over the corresponding design using shortest-path and no span eliminations (Table 9.1).

Table 9.7. The most-improved SCIP design results using IRE multi-modular design.

Network	No. of Span Elim.	Span Elim. Sequence	SCIP Design Cost (Millions)	Comp. Time (seconds)	% Improvement
A	8	(5-11)-(4-8)-(5-6)-(1-4)-(1-5)-(9-11)-(5-8)-(1-8)	1.43	75.80	25.91%
B	8	(8-17)-(1-3)-(19-18)-(12-10)-(10-8)-(3-4)-(10-13)-(13-15)	14.54	145,538.70	17.95%
C	3	(36-119)-(71-79)-(78-118)	10.92	296.27	1.43%
D	18	(36-119)-(25-97)-(34-109)-(78-118)-(89-104)-(38-62)-(71-79)-(83-85)-(62-103)-(9-31)-(9-46)-(63-119)-(63-65)-(-104-125)-(34-101)-(25-102)-(113-123)-(70-113)	24.26	152,503.60	10.85%

9.4 Results using PIR

Table 9.8 summarizes the most-improved SCIP design found by PIR for each respective test network. The *No. of Span Elim* (Number of Span Eliminations) column indicates the number of spans that are removed from the network. PIR reduces the total number of supermodular spans in the network by first re-routing the excess capacities in the supermodular spans, and then second by means of eliminating the non-essential spans from the network and detouring the affected demands at the span-level. PIR halts when subsequent elimination of the remaining non-essential spans has resulted in an increase in the number of supermodular spans. The minimum number of supermodular spans is shown in Table 9.8 below in the *Remaining No. of Supermodular Spans* column. The *% improvement* shows the percentage improvement obtained from the most-improved SCIP design results over the corresponding design using shortest-path and no span eliminations (Table 9.1).

Table 9.8. The most-improved SCIP design obtained using PIR.

Network	No. of Span Elim	Eliminated Span	Remaining No. of Supermodular Spans	SCIP Design Cost (Million)	Comp. Time (seconds)	% Improvement
A	6	(1-3)-(1-4)-(4-8)-(5-8)-(5-11)-(9-11)	0	1.58	19.7	18.55%
B	11	(3-4)-(12-10)-(10-8)-(13-15)-(9-15)-(9-17)-(17-19)-(19-18)-(22-25)-(23-29)-(21-23)	0	15.88	1204.26	10.42%
C	3	(32-84)-(36-89)-(36-118)	6	10.91	74.82	1.46%
D	0	-	10	26.85	2304.84	0.00%

Note that the most-improved SCIP design results for Net D has zero number of span eliminations and using PIR shows no improvement in design cost over SCIP design using shortest-path and no elimination. This observation is further discussed later in this chapter.

9.5 Comparison

This section compares the performance of iMAPR, IRE and PIR using multi-modular design. Table 9.9 below shows the most-improved SCIP design from iMAPR, IRE and PIR over the series for each test network. These results were compared against those from multi-modular SCIP using pure shortest-path and no span eliminations included in Table 9.9. Figure 9.5 below gives the percentage decrease (increase) in the total design for the iMAPR, IRE and PIR strategies relative to the cost obtained with shortest-path routing.

Table 9.9. The most-improved multi-modular SCIP design cost and the required computational time using pure shortest-path, iMAPR, IRE and PIR.

Network	Multi-Modular Shortest-Path SCIP		iMAPR		IRE		PIR	
	SCIP Design Cost (Million)	Comp. Time (seconds)	SCIP Design Cost (Million)	Comp. Time (seconds)	SCIP Design Cost (Million)	Comp. Time (seconds)	SCIP Design Cost (Million)	Comp. Time (seconds)
A	1.94	9.7	1.65	33.10	1.43	75.80	1.58	19.7
B	17.72	900.14	16.09	19,438.40	14.54	145,538.70	15.88	1204.26
C	11.08	39.91	10.99	214.15	10.92	296.27	10.91	74.82
D	27.21	900.59	28.22	17,347.29	24.26	152,503.60	27.21	2304.84

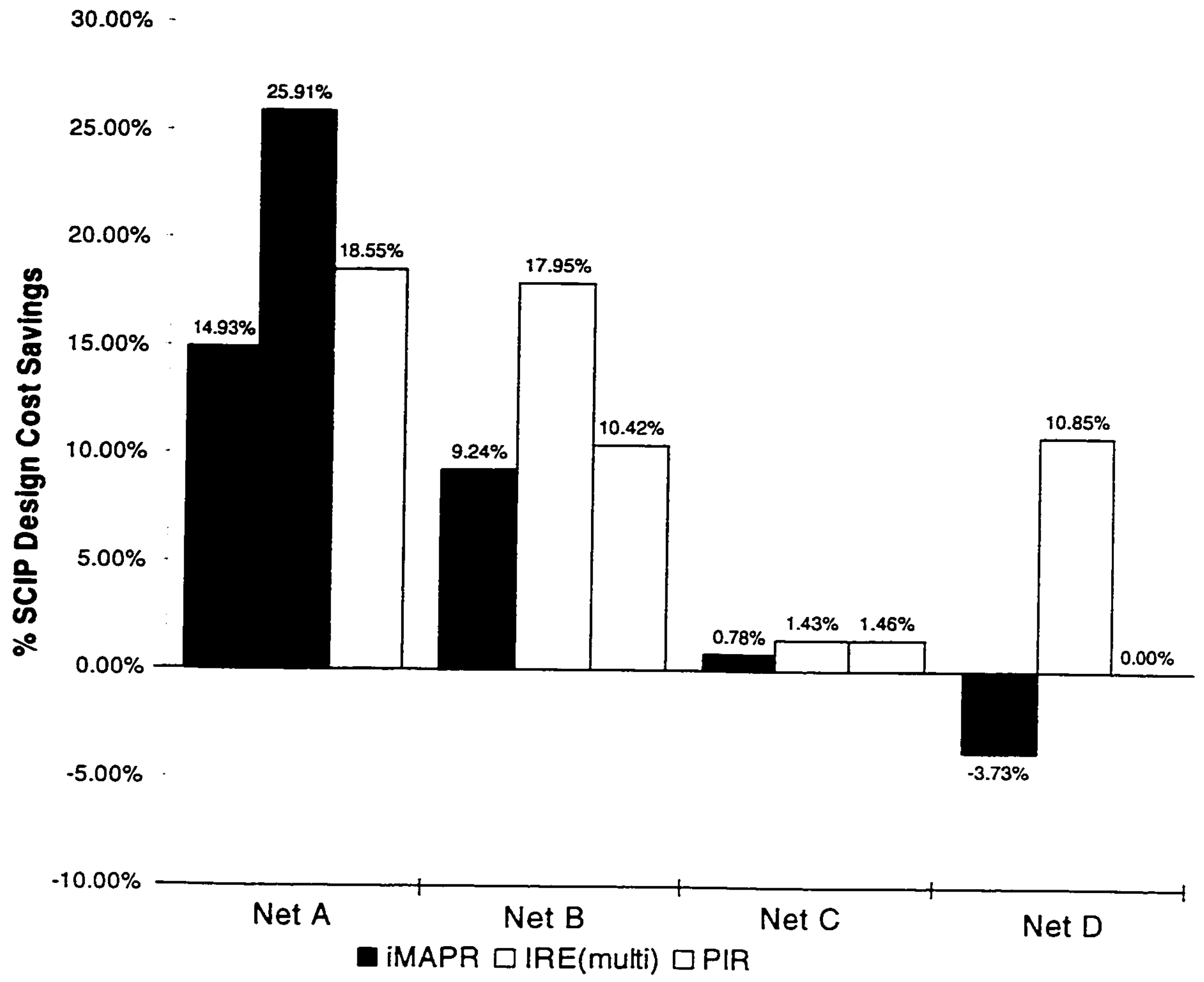


Figure 9.5. Percentage Total Network Cost Savings for iMAPR, IRE(multi) and PIR relative to multi-modular SCIP reference design.

In Figure 9.5, the total network cost savings reported by iMAPR range from -4% to 15%. The negative cost savings reported my iMAPR on Net D implies that the ring design using shortest-path and no elimination is the preferred solution [*]. PIR performs slightly better with cost savings ranging from 0% to 19%, where the 0% signifies that reducing the supermodular spans in PIR did not have an effect on reducing the SCIP design cost with respect to the SCIP design using pure shortest-path and no span eliminations. The best cost savings came from using IRE, where the reported cost savings range from 1% to 26%. However, amongst the span elimination heuristics

[*] The iMAPR algorithm alone does not perform a trial design using shortest-path and no elimination. In practical situation, a planner will perform a baseline design using shortest-path with no span eliminations before attempting to initiate any span elimination strategies. Therefore to reflect practical interest, negative cost savings will be denoted as zero cost savings.

examined, PIR requires the least amount of CPU computational time followed by iMAPR. IRE, on the other hand, requires a large amount of computational time. The relationship between solution quality and computational time is depicted in Figure 9.6. The *x*-axis (Relative Computational Time) is the computational time results obtained from iMAPR, IRE and PIR normalized by the respective computational time results using shortest-path and no span eliminations (refer to Table 9.9). The *y*-axis (Relative SCIP Design Cost) is the SCIP design cost obtained from iMAPR, IRE and PIR normalized by the respective SCIP results for shortest-path routing and no span eliminations.

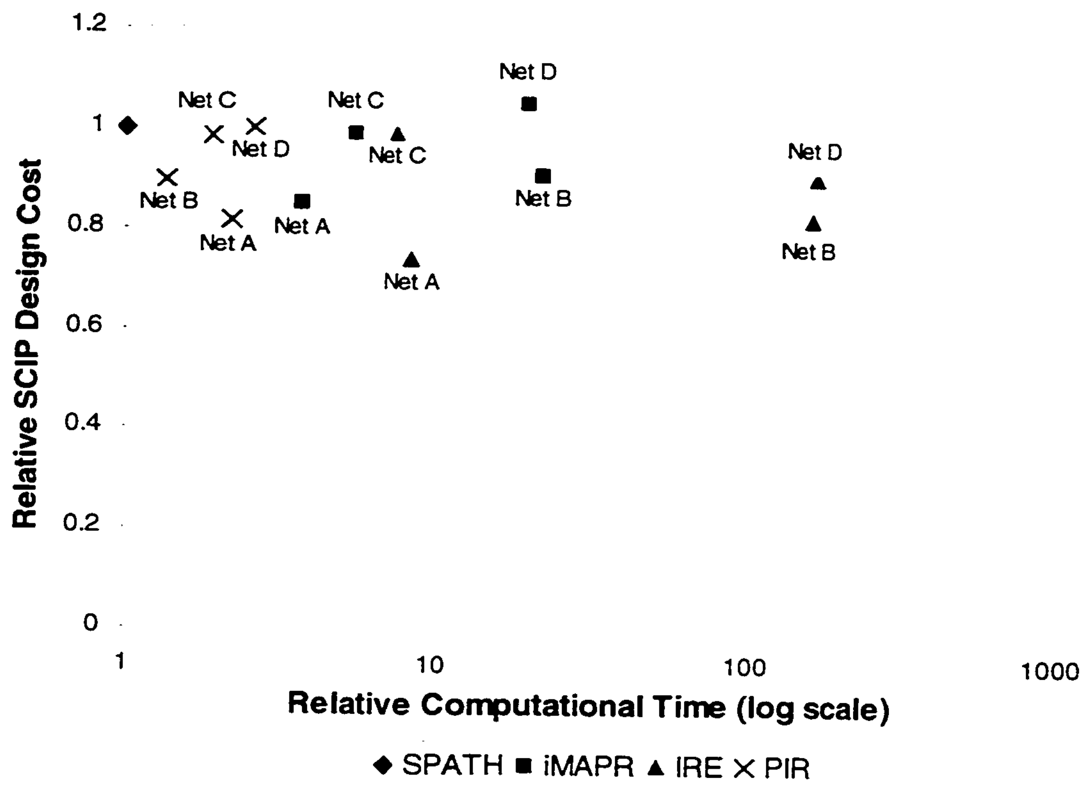


Figure 9.6. **Scatter plot for iMAPR, IRE and PIR results normalized to corresponding shortest-path and no span elimination results to show the relationship between solution quality and computational time.**

The scatter plot in Figure 9.6 shows that IRE has a slow computational runtime (IRE points to the right of the figure) but has the best solution quality (IRE points are scattered lower that other heuristics). PIR, on the other hand, has the fastest computational runtime providing an average solution quality (PIR points are scattered to the left of Figure 9.6 and are vertically in-between

IRE and iMAPR points). iMAPR requires a moderate computational runtime and give an average solution quality (iMAPR points are scattered horizontally between PIR and IRE points and are vertically higher than other heuristics).

9.6 Concluding Discussion of Results

This chapter explored and compared the performance of iMAPR, IRE and PIR. The results illustrated in Table 9.9 and Figure 9.5 show a clear trade-off between the solution quality and the total computational-time. Although IRE still gives the highest solution quality (between 1% and 26% improvement), it requires a considerably longer computational time than iMAPR and PIR (as shown in Table 9.9). The full runtime for IRE is high because of its built-in BFS strategy where it employs SCIP as a fundamental measurement for selecting the span elimination that lead to the least min-cost SCIP design at every iteration.

iMAPR, however, demands a relatively shorter computational time since, regardless of the size of the investigated network, it requires a constant total of 21 SCIP runs. But the solution quality found is compromised (between 0% to 15%) relative to IRE. The zero cost savings by iMAPR for Net D signifies that there is a case where iMAPR is unable to give a better and improved cost solution relative to its counterpart results using pure shortest-path. Further suggestions to improve iMAPR heuristic are given later in Chapter 11.1.

Using PIR, requiring only a fraction of iMAPR's total computational time, gives a better solution quality than iMAPR (between 0% to 19%). For example, in the investigations of Net B, the cost savings reported by iMAPR are 9% with a computational time of 353.3 minutes; PIR only requires 20.1 minutes, which is a mere 6% of the computational time required by iMAPR, and gives a total cost savings of 10%. As such, PIR gives better solutions employing less computational time. Overall, the results obtained by PIR generally outperformed iMAPR both in terms of solution quality and computational time.

When comparing PIR against IRE, three out of the four investigations, Net A, Net B and Net D, show that IRE gives better solutions than PIR. For Net C, PIR gives a slightly better solution than IRE, 1.46% for PIR as opposed to 1.43% by IRE. However, the 0% cost savings reported by PIR on Net D signify that there are situations where the current PIR strategy is unable to produce a solution quality as good as IRE's, and reducing the number of supermodular spans did not result in a positive cost reduction with respect to the SCIP design obtained from shortest-path and no eliminations. Some further improvements on the PIR strategy will be described later in Chapter 11.1.

In terms of total computational time, PIR has a strong advantage. PIR requires only two SCIP runs regardless of the size of the network. A large portion of the PIR processes and methodologies are based on a straight-forward process of determining the supermodular spans. The built-in BFS strategy searches for the sub-graph, amongst other sub-graphs, that has the minimum number of supermodular spans at every BFS iteration. Here the measurement used is a function call that searches for the number of supermodular spans on the remaining sub-graph as opposed to the complete SCIP run implemented by IRE. In addition, PIR does not need to examine and trial-eliminate all non-essential spans; it halts when subsequent elimination of the remaining non-essential spans results in an increase in the number of supermodular spans.

In all the experiments performed in this chapter SCIP is limited in CPU time to 900 seconds. The rationale was a necessary practical measure to permit the development of a complete set of comparative results in a reasonable time while retaining a fair and defined basis amongst the methods in the way that they use SCIP. When multi-modular SCIP is employed, there are four times the number of ring candidates that the IP has to investigate relative to a single-modular SCIP. Indeed, the ideal would be to remove the constraint of computational time and let SCIP finish investigating all the possible SCIP designs. But for investigations of large and complex networks, such as Net B and Net D, it might take weeks or even months for a single SCIP run to be completed. In addition, the improvements generated by SCIP with high computational runtime are relatively small. Comparing SCIP designs with a 24-hour limit runtime obtained in Table 3.4 with SCIP designs for a 900-second limit runtime obtained in Table 9.1, it can be seen that the gap between the cost results ranges from 1% to 6%. Hence, the benefit of limiting SCIP to a shorter runtime out-weighs the benefit of the cost improvements from SCIP with a long computational run-time.

A qualitative summary comparison of each span elimination heuristic investigated in this chapter is illustrated in Table 9.10. The *Category* row displays the approach that the span elimination heuristic used as described in Chapter 2.1. The *Quality of Solution* and *Comp. Time* rows compare the cost and runtime results obtained for each of the strategies. The *No. of SCIP Runs* shows the total number of SCIP runs necessary to complete the corresponding strategy. Here, k is the number of non-essential spans found in the given network topology. The *Re-routing Method* illustrates the strategy used to re-route the affected demands due to the elimination of non-essential spans. The *Span Elimination Technique* displays the core mechanism used by each heuristic to systematically eliminate the non-essential spans and obtain the most-improved SCIP design.

Table 9.10. Qualitative Comparison of the Span Elimination Heuristics.

	Pure Graph Cover	**Span Elimination Heuristics**		
		iMAPR	**IRE**	**PIR**
Category	-	Pre-Processor	Post-Processor	Post-Processor
Quality of Solution	Existing state of the art	Medium	Good	Medium
Comp. Time	Fast	Moderate	Slow	Fast
No. of SCIP Runs	1	21	$k(k+1)/2$	2
Re-routing Method	-	-	MAPR(rise), Path-Level	MAPR(rise), Span-Level
Span Elimination Technique	-	Applying Aggregation Pressure on Spans	BFS using SCIP	BFS using Supermodular Spans

10.0 Correlation between SCIP and RingBuilder

10.1 The Issue

This chapter investigates the use of SCIP as the evaluation tool for span elimination heuristics. As described in Chapter 2, SCIP involves certain idealizations. For example, SCIP assumes that every node in the network has an ADM on every ring at its site and that demands may transit between rings at any location. In addition, a 'side-effect' of using the IP described in Chapter 3 has limited SCIP to use only BLSRs for its investigations. As such, a sub-study was warranted to see how well SCIP correlates to full-blown RingBuilder* results. The requirement is not that SCIP absolute design costs be as good as RingBuilder, but only that high SCIP costs correlate to high RingBuilder costs. As long as this is true, SCIP can be used confidently as a surrogate for the complete ring design process in a *comparative* study such as this. From what follows, it can be concluded that SCIP is useful as a precisely defined and repeatable reference model for the ring coverage design phase through which the effectiveness of the span elimination strategies investigated previously can be interpreted.

The method used to evaluate the usage of SCIP is by completing a second run using RingBuilder v2.0 on all the experiments that were executed in Chapters 3.3, 6 and 9 using SCIP. RingBuilder is a full-blown ring coverage design heuristic that is used commercially for SONET ring-based network planning. It uses a greedy heuristic method that selects from the modular ring candidate system for the network design to achieve a near-optimal placement of ring systems on the network. A more thorough methodology and optimization strategy taken by RingBuilder is described in [15,30]. RingBuilder uses the same inputs as SCIP, i.e., the capacitated SNIF and the cost model. It has a built-in cycle finding module that searches exhaustively for all the cycles in a given network graph similar to the cycle generator used in this work. In addition to these inputs, RingBuilder requires a routing file that consists of all the route information taken by each PTP demand. All the proposed span elimination algorithms (single-modular and multi-modular) proposed in this work generate a routing file except for PIR. Hence, results using PIR are not included in this investigation. Additional information on the inputs required by RingBuilder are described in [30]-[32].

* RingBuilder is a group of software programs developed by TR*Labs* for synthesizing ring-based network designs using BLSRs and UPSRs. RingBuilder heuristic finds the best ring system to achieve a near-optimal placement of ring systems on the network.

Two data sets were obtained, one a set of SCIP design results and the other a set of RingBuilder results. Only SCIP results obtained from completed SCIP runs are included in the data set, not SCIP results that are found strictly due to the time limitations, e.g. the results that are marked by (*) in the *Comp. Time* column in Table 3.5. Similarly, other designs that resulted from an incomplete SCIP run are not included in the data set. The aim is to determine how well SCIP results and RingBuilder results correlate statistically over a large number of trial runs.

The method used to measure the relationship between the two ranges of data is measuring the correlation coefficient of the two data sets that are scaled to be independent of each other. Eq.10.1 below measures the correlation coefficient of data set *X* and data set *Y*.

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \cdot \sigma_Y} \qquad \textbf{(10.1)}$$

where,

$$\mathrm{cov}(X,Y) = \frac{1}{a}\sum (X_i - \mu_X)(Y_i - \mu_Y)$$

$$\sigma_X{}^2 = \frac{1}{a}\sum (X_i - \mu_X)^2$$

$$\sigma_Y{}^2 = \frac{1}{a}\sum (Y_i - \mu_Y)^2$$

a	- No. of samples.
μ_X	- Mean of data set *X*.
μ_Y	- Mean of data set *Y*.
X_i	- Element *i* of data set *X*.
Y_i	- Element *i* of data set *Y*.

The population correlation calculation returns the covariance of two data sets *X* and *Y*, *cov(X,Y)*, divided by the product of their standard deviations, σ_X and σ_Y. This covariance of the two data sets returns the average of the product of deviations of data points from their respective means. As such, this correlation coefficient determines whether two ranges of data move together - i.e., whether large values of one set are associated with large values of the other (positive correlation), whether small values of one set are associated with large values of the other (negative correlation), or whether values in both sets are unrelated (correlation near zero). For the measurement to be reliable, a relatively large pool of data is required. Here, a total of 183 data samples (a = 183) has been accumulated.

10.2 Results and Discussions

Using Equation 10.1, it was found that the correlation coefficient between the two data sets with a=183 is 0.97. In addition, for visual inspection, Figure 10.5 below shows the cost results obtained from SCIP and RingBuilder plotted against the sample number.

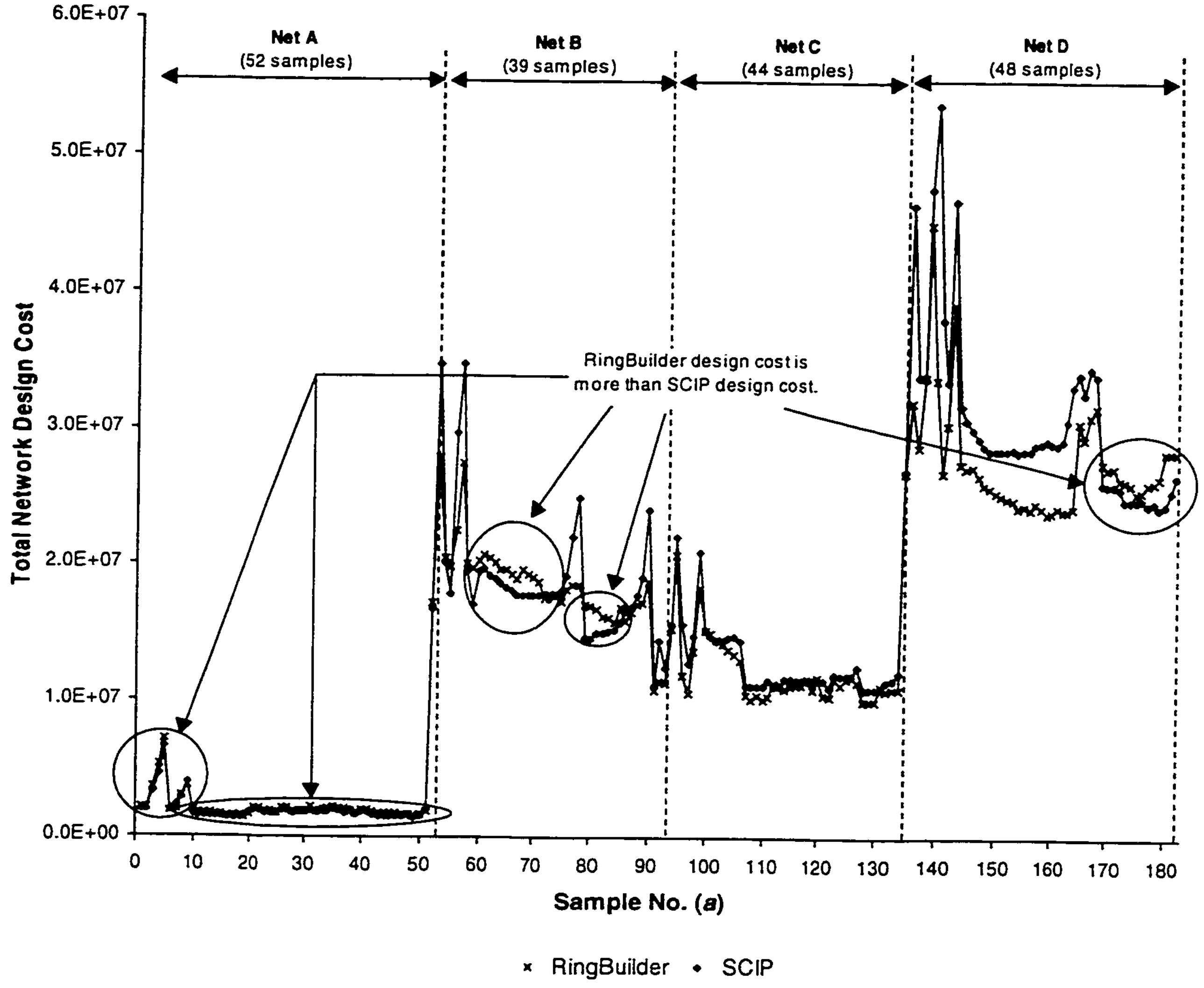


Figure 10.1. The total network cost obtained using RingBuilder v2.0 and SCIP.

Figure 10.1 shows a minimal discrepancy between the cost resulting from SCIP and the cost resulting from RingBuilder. In most of the samples, the large values of one set do associate with the large values of the other set and vice versa. Hence, using Eq.10.1, the correlation coefficient between the two data sets was found to be 0.97. This value suggests a very strong correlation that supports the claim that SCIP results can be well applied as a surrogate for the full-blown RingBuilder designs in purely relative or comparative design studies.

With accurate cost modelling, the absolute dollars obtained by RingBuilder results are almost always lower than those obtained by the SCIP results, since RingBuilder takes the cost for all the required rings and the ring-to-ring transitions into consideration, including the effects of glass-throughs where ADMs are replaced with the relatively cheaper repeaters. SCIP only takes into consideration the cost of all required rings, ignoring the ring-to-ring transition cost and the effects of glass-throughs. However, the cost reported by RingBuilder may not always be less than the cost reported by SCIP as indicated in Figure 10.1. These situations occur for the designs on Net A, Net B and Net D when the total cost of the ring-to-ring transitions is high due to the relatively high average nodal degree of the networks.

11.0 Summary of Thesis

This thesis explored the feasibility and the performance of using span eliminations to advance the art of ring network design beyond the pure coverage design principle. Currently, there are good solutions for optimized multi-ring network design where the design problem is posed as a form of graph-covering problem. The capacity requirements of each span are generally determined by routing demands over the shortest-path from end-to-end. Intuitively, this can lead to one or more low-utilized spans. Performing optimization in ring design on the resulting fiber graph, and approaching this optimization problem as a graph-covering problem, may impose unnecessary lower bounds on the network cost when the min-cost ring set found exists strictly to protect every capacitated span in the fiber graph, including the low-utilization spans. In this case, one or more rings may effectively be required due to strict coverage requirement, but serve little demand.

By solving the min-cost multi-ring network design problem from the topological perspective of the network, one can recognize and remove those key spans in the existing fiber graph at the point where it may be more effective not to route any demands at all, thereby avoiding the requirement for ring coverage on those spans. The motivation behind the present work was, therefore, to explore methods to systematically analyze the fiber graph for opportunities to remove certain spans so that subsequent coverage algorithms on the resulting sub-graph would give better ring network design costs.

In a relatively small network, it is sometimes easy by manual inspection to spot situations where span eliminations will improve the overall network cost. However, this is not the case for large networks of practical interest. Even without the aspect of span eliminations, the design of multi-ring networks is an exceptionally complex combinatorial optimization problem [16]-[20] and is known to be NP-Complete [16,17]. Therefore, the span elimination problem has been approached by proposing heuristic procedures, based on insights into the underlying nature of the ring-network design problem, for finding improved ring network designs. There is no formal proof that the results are optimal. Nevertheless, experimental results show that the proposed heuristic methods offer worthwhile improvements in design cost, and that some may be efficient tools to consider for production use in systems such as RingBuilder.

The work in span elimination was begun by determining the standard coverage algorithm to be implemented. The Span Coverage Integer Program (SCIP) [14] was adopted. Described in

Chapter 3, SCIP is a pure (ideal) capacitated span coverage integer program which finds the strictly min-cost subset of rings that are required to cover all the non-zero span working capacities in the given capacitated network. Several idealizations have been recognized, that is, this IP assumes that every node has an ADM and that demands may transit between rings at any node. In addition, due to its IP formulation (constraint set 3.2), SCIP is only suitable for shared-protection rings limiting the investigations to SONET BLSRs only. In spite of the drawbacks, SCIP is precisely defined and repeatable, leading to results being easily reproduced and verified. To validate SCIP, a comparison between the SCIP results and the RingBuilder was performed, and the results are given in Chapter 10. It has been shown that there is a minimal discrepancy between the costs resulting from SCIP and the costs resulting from RingBuilder with the correlation coefficient found to be 0.97, strengthening the claim that SCIP results can be well-applied as a surrogate for the full-blown RingBuilder designs in purely relative or comparative design studies. It is important to note, however, that the focus of this study is in the usage of SCIP as a quick surrogate for a full-blown ring network design in relative cost comparison contexts. In accuracy-sensitive situations, for example if an absolute and accurate cost minimization of a real design is to be deployed, RingBuilder or other full-scale design development tools may be used.

In Chapter 4, a pre-processor span elimination heuristic called Modular Aggregating Pre-Routing (MAPR) is proposed. Unlike the conventional ring design shortest-path algorithm, MAPR routes demands based on the least-cost path determined by the pseudo-cost of each span. The motivation is to route demands in a manner that tends to aggregate flows on spans, leaving totals that are well-suited for later modular ring coverage design. The pseudo-cost is determined by the span's geographical distance and its current capacity requirements with the idea that once a span has some demand on it, it should look more attractive (cheaper) to subsequent least cost routing determinations, but that it should be cancelled as soon as a module boundary arises. Two pseudo-cost functions were tested, namely rise and impulse (Eq.4.1 and Eq.4.2). Implementing MAPR is relatively simple as it only requires a single iteration where PTP demands are routed using rise or impulse; any non-essential spans thereafter with zero working capacity are eliminated, and SCIP is executed on the resulting sub-graph.

Chapter 5 introduced a post-processor span elimination heuristic called Iterated Routing and Elimination (IRE). IRE uses a strategy that involves a series of single span elimination trial runs with ring coverage design process calls for each one to search for a set of span eliminations that permits (ideally) the lowest cost coverage design. In IRE, the non-essential spans are identified, after which IRE begins. IRE can be described at two logical levels: one is a low-level iteration at which a single non-essential span is eliminated, the affected demand matrix is routed in a MAPR-like way at the path-level (specifically using rise pseudo-cost function) on the remaining sub-graph, and a ring coverage design trial using SCIP is obtained; and, above this, a BFS is in effect to develop a sequence of accumulating eliminations from the set of non-essential spans. Consequently, IRE requires a high number of SCIP runs which are quadratically dependant upon the number of non-essential spans.

In Chapters 4 and 5, the investigations were limited to using a single-modular design (OC-12, OC-48, OC-96 and OC-192) with the cost model for the network equipment following a certain economy-of-scale (Table 3.1). The experimental results obtained for MAPR showed a strong relationship between resulting SCIP design costs and the permitted modular capacity. Using a small modular capacity may not be adequate for accommodating the increase in the $\overline{w}_i$ due to the span eliminations at which additional rings are prompted to cover the excess capacities, increasing the overall network costs. On the other hand using large modular capacity increases the network equipment cost. Hence, there exists a best-fit modular capacity where the overall minimum SCIP results are obtained. Comparing the MAPR results using rise and impulse, it can be seen that MAPR(rise) showed cost improvements for 11 out of 16 results whereas MAPR(impulse) only showed cost improvements for 7 out of 16 results when compared against designs using pure shortest-path without span eliminations. It appears that impulse introduces an extreme level of aggregation pressure at which demands are severely detoured, and there is not enough slack capacity in the spans to accommodate the increase in w_i. Therefore, although there are more span eliminations (since more demands are attracted towards the 'cheaper' used spans, more spans will remain unused), the design cost may still be higher. As such, impulse may show a better design cost when a large modular capacity is permitted. The evaluation of impulse led the research to focus on MAPR(rise) where there is a compromise in the level of aggregation pressure.

As for IRE results, it was discovered that a similar pattern occurs in all the IRE curves obtained from different test networks. The first falling section of the curves shows that cost decreases as more spans are eliminated so that the lightly loaded rings that are forced upon the design to cover relatively lightly loaded spans are either removed or become more efficiently loaded. This continues until a minimum value (most-improved design) is reached where the spans and rings have achieved the best possible utilization. Further span eliminations eventually drive up the spans' capacity requirements until the ring capacity is exceeded and additional rings are triggered.

Comparing both MAPR(rise) and IRE results relative to the designs using pure shortest-path and no span eliminations, it was found that there is a trade-off between solution quality and computational time. The total network cost savings with respect to shortest-path and no elimination SCIP design reported by IRE range from 3% to 26% as compared to 0% to 5% reported by MAPR(rise), showing that IRE consistently out-performed MAPR. The zero cost savings by MAPR(rise) shows that the shortest-path with no elimination SCIP is the preferred design, and that there are instances where MAPR may not be the suitable approach to use. However, IRE requires a considerably longer computational time since it employs the breadth-first search strategy, at which each low-level iteration employs a full SCIP coverage design, causing an increase in the runtimes when the network has a large number of non-essential spans (The number of SCIP runs increases quadratically with the number of non-essential spans, Equation 5.1).

In conclusion, it was found that MAPR in general is not a reliable span elimination heuristic, essentially because it produces a one-shot set of span eliminations that depends on the aggregation pressure applied on the span using the capacity-sensitive pseudo-cost function to efficiently route demands on the network, leading to a certain number of span eliminations. As such, MAPR does not have any indication whether it has over-done the span eliminations or not. IRE, on the other hand, has the highest solution quality but requires a substantial amount of runtime to complete.

Chapter 7 proposed to improve the MAPR primarily by giving it a progressive sweep-like aspect over a range of aggregation pressures. This was done by using 21 different pseudo-cost curves with different aggregation pressures and capturing the SCIP design results from each. This was called Iterative MAPR (iMAPR) where MAPR is executed at each iteration, each time using a different pseudo-cost curve. Here, relatively more complex pseudo-cost functions have been developed (Eq.7.6 and E1.7.7). The SCIP design that gives the overall least-cost will be the most-improved SCIP design found by iMAPR.

Further motivated to try some wholly new approach that can give good solutions with a fast runtime, Chapter 8 presented Post Inspection and Re-Routing (PIR). PIR is a post-processor span elimination heuristic which utilizes the knowledge of supermodular spans in a baseline SCIP design to reduce the total network design cost. A supermodular span is a span where the w_i is more than the current largest span modular capacity used to cover the span with respect to the reference rings first placed in a basic SCIP run on the network using shortest-path and no span eliminations. The remainder of w_i divided by the largest span modular capacity is called the supermodular capacity. PIR is based on the hypothesis that a reduced number of supermodular spans will lead to lower cost ring network coverage designs.

PIR has two phases. Phase I reduces the number of supermodular spans by means of re-routing the supermodular capacities at the span-level. Phase II then implements span elimination using the BFS strategy to search for the sub-graph that gives the fewest supermodular spans. This is a purely polynomial time re-routing test and does not involve any SCIP runs for evaluating each trial-elimination. Iteration stops upon the discovery of a sub-graph with the fewest supermodular spans. A final SCIP design run follows. Hence, only two SCIP runs are required: the first is to obtain the ring coverage design of the initial fiber graph with no elimination using pure shortest-path, and the set of rings is then used as a reference to determine supermodular spans; and, the second SCIP is executed on the sub-graph that gave the least possible number of supermodular spans. The two SCIP designs are then compared where the overall least cost SCIP design becomes the most-improved SCIP design found using PIR. With only two SCIP runs, PIR does not have the problem of high computational time.

Chapter 9 was the main results chapter for the final half of the thesis. It compares the performance of iMAPR, IRE(multi) and PIR using multi-modular SCIP. In multi-modular designs, rings are selected in terms of the cost and modular capacity that can best be implemented to cover the w_i of the network. However, since four different modular capacities are being investigated, SCIP now has four times the number of ring candidates that it has to compute. This change has a tremendous impact on computational time since the increase in the number of ring candidates increases the number of possible designs in SCIP (Eq.3.5). Here, the total CPU time for each SCIP run was limited to 900 seconds. This was a necessary practical measure to permit the development of a complete set of comparative results in a reasonable time while retaining at least a fair and defined basis amongst the methods in the way that they use SCIP. It was found that when comparing the design cost obtained using a 24-hour limited CPU runtime and a 900-seconds limited CPU runtime, the gap between the two results was between 0% to 6%. It was then fair to conclude that the benefit of limiting SCIP to a shorter runtime out-weighs the benefit of the cost improvements given by SCIP with a long computational runtime.

Comparing the solutions obtained by iMAPR, IRE(multi) and PIR relative to the SCIP designs obtained for the network using pure shortest-path and no eliminations, it was found that IRE(multi) gave the highest solution quality with cost savings ranging from 1% to 26%. PIR came second with cost savings ranging from 0% to 18% and then iMAPR with 0% to 15%. The zero cost savings by iMAPR and PIR reported for Net D shows that zero elimination and using shortest-path for Net D is the preferred design, indicating that there are situations where the current iMAPR and PIR strategies are unable to produce solutions that are as good as IRE. iMAPR was unable to find the aggregation pressure to use a better and a positive cost reduction relative to its counterpart results using pure shortest-path.

In terms of total computational time, PIR has the fastest runtime, iMAPR is next, and IRE(multi) requires the longest runtime. These results are mainly based on the total number of SCIP runs that each heuristic employs. PIR implements only two SCIP runs where iMAPR requires 21 SCIP runs, and the SCIP runs for IRE are based on $k(k+1)/2$ (where k is the total number of non-essential spans).

Of all the span elimination heuristics, it seems clear that PIR gave the best compromise between solution quality and computational time. PIR gave an average solution quality, but the most appealing aspect of PIR is that it requires only two SCIP runs and the remainder of its algorithm is based on a polynomial time analysis of supermodular capacities as a means to detect the desired span eliminations. In this regard, it is capable of giving a relatively fast span elimination solution regardless of the network size. On the other hand, IRE offers good solution quality and gives positive cost savings while the other heuristics may not. With up to a reported cost saving of 26%, IRE has shown considerable cost benefits and is an appealing algorithm by giving good span elimination solutions. However, due to its reliance on SCIP and the quadratic relationship between its total number of SCIP runs and the number of non-essential spans, it can and does exhibit relatively long computational runtimes for large test network models. iMAPR has been shown to give average solution quality employing a moderate runtime. Further research in iMAPR is required to improve its effectiveness in terms of solution quality.

11.1 Suggestions for Further Research

11.1.1 Test Case: The impact of using a different non-essential spans set on SCIP design cost (Using IRE heuristic on Net B).

Throughout the experiments in Chapter 6 and Chapter 9, only a single set of non-essential spans is investigated for each test network (as shown in Figures 6.1 to 6.4). These non-essential span sets are determined by arbitrarily selecting one of the test networks' minimal bi-connected sub-graph, determined by inspection (details are outlined in section 2.3.1). However, it is realized that there can be more than one MBG. As a result, different non-essential spans set can be obtained for the test networks. Therefore, this section is a test case of the impact that using a different non-essential span set will have on the SCIP design cost. Specifically, the test case to be performed here investigates Net B using a different set of non-essential spans and IRE multi-modular span elimination heuristic.

To begin this test case, it is realized that from [35], that odd degree nodes can cause rings to overlap. This can be avoided by having even degree nodes. Thus, the selected MBG for Net B in Figure 6.2 may be improved by increasing the "evenness" of the node degrees. This is performed manually (by hand) and the improved MBG is shown in Figure 11.1 with the new set of non-essential spans shown in Table 11.1. Set 1 in Table 11.1 is the set of non-essential spans previously defined from Figure 6.2 and the new set of non-essential spans is called Set 2 in Table 11.1. When this two sets are compared, the MBG for Set 1 has 14 even degree nodes where as the new Set 2 has 20 even degree nodes, an increase by 6. Both of these sets have 18 non-essential spans.

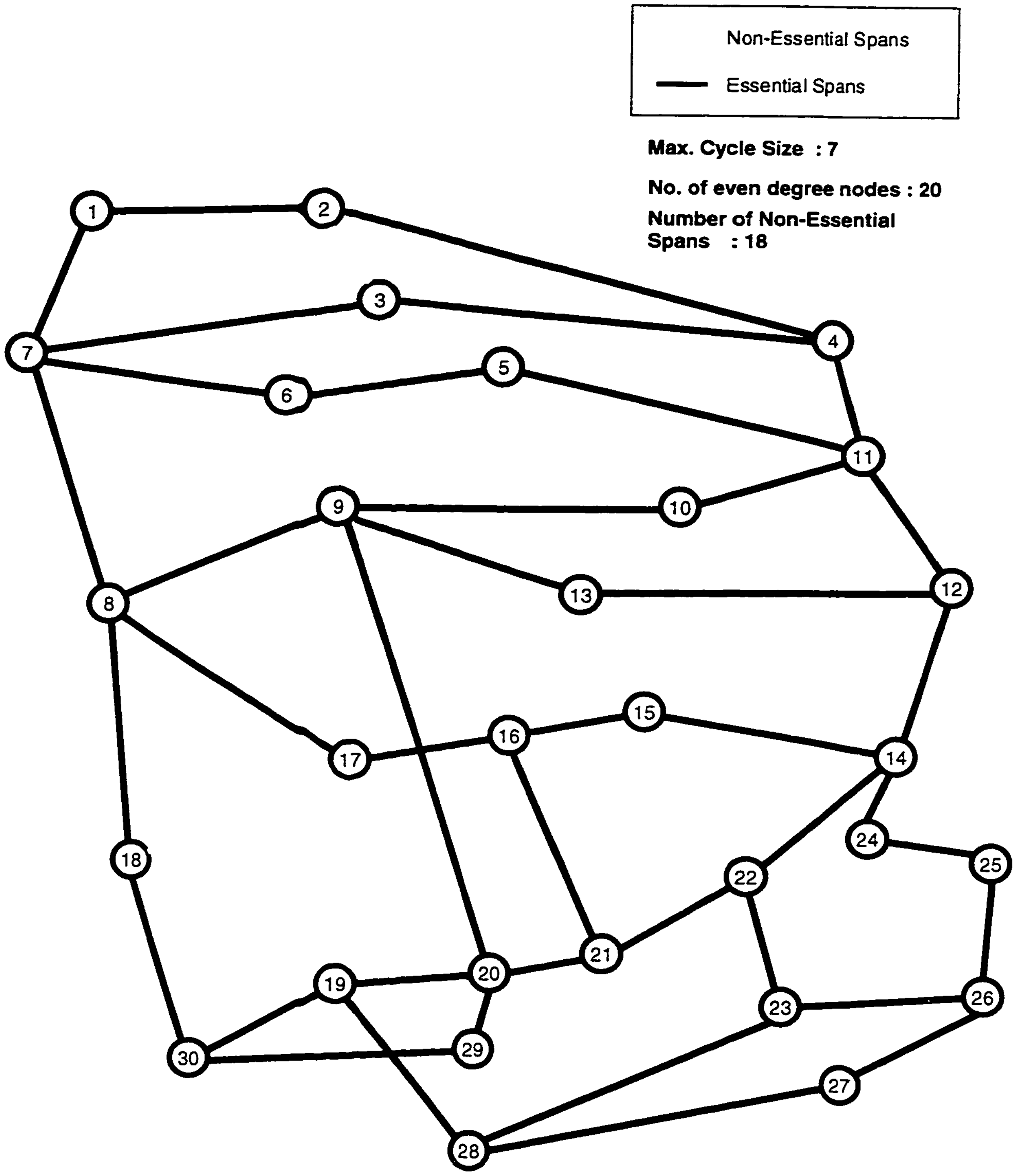


Figure 11.1. The new minimal bi-connected sub-graph and non-essential spans for Net B.

Table 11.1. **The new set of non-essential spans for Net B.**

Network	B	
Set	1	2
Max. Cycle Size	7	7
No. of Non-Essential Spans, *k*	18	18
Non-Essential Spans	(1-3)	(1-3)
	(3-4)	(2-3)
	(7-3)	(3-5)
	(6-9)	(5-10)
	(5-11)	(6-9)
	(12-10)	(8-10)
	(10-8)	(10-12)
	(10-13)	(10-13)
	(13-15)	(13-15)
	(9-15)	(9-15)
	(9-17)	(9-17)
	(8-17)	(9-18)
	(17-19)	(17-18)
	(19-18)	(17-19)
	(22-25)	(18-19)
	(9-18)	(21-23)
	(23-29)	(22-25)
	(21-23)	(23-29)

A similar experimental setup, used to investigate IRE(multi) in Chapter 9, was used here to test the impact of Set 2, the new non-essential spans set. Figure 11.2 shows the IRE(multi) results and the specific combination of span eliminations that were found during each IRE iteration are displayed adjacent to each point as shown in Figure 11.2. The IRE(multi) curve obtained using Set 1 non-essential spans are also included in Figure 11.2 for comparison. The solution quality and the computational time obtained from IRE(multi) using both Set 1 and Set 2 non-essential spans are shown in Table 11.2.

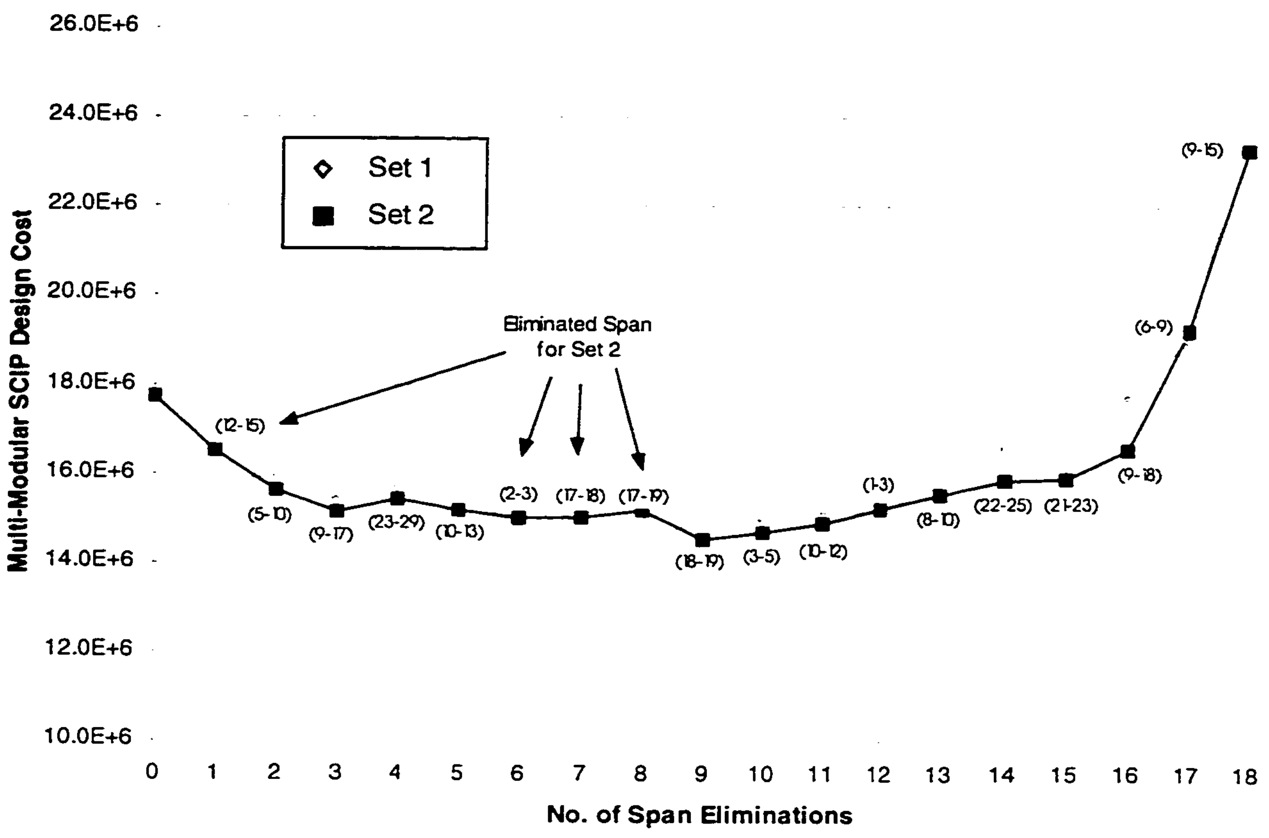


Figure 11.2. SCIP results for Net B using IRE(multi) with Set 1 and Set 2 non-essential spans.

Table 11.2. The most-improved SCIP design results using IRE(multi) with Set 1 and Set 2 non-essential spans.

Network	Non-Essential Spans Set	No. of Span Elim.	Span Elim. Sequence	SCIP Design Cost (Millions)	Comp. Time (seconds)	% Improvement
B	Set 1	8	(8-17)-(1-3)-(19-18)-(12-10)-(10-8)-(3-4)-(10-13)-(13-15)	14.54	145,538.70	17.95%
	Set 2	9	(12-15)-(5-10)-(9-17)-(23-29)-(10-13)-(2-3)-(17-18)-(17-19)-(18-19)	14.50	144,535.08	18.19%

Observing Figure 11.2, the IRE curve obtained using Set 2 exhibits similar pattern resembling the curves obtained for IRE(single) in Chapter 6.1.2. In fact, this curve is almost similar to the IRE curve obtained using Set 1. Table 11.2 shows that the most-improved design using Set 2 occurs when nine spans were eliminated giving a 18.19% improvement over design using shortest-path and no eliminations. When compared these results with the results using Set 1, Table 11.2 shows that Set 2 has a completely different *Span Elim. Sequence* and giving results slightly better than Set 1, a mere 0.24% improvement (Set 1 gave 17.95% improvement over SCIP design using shortest-path and no eliminations). In addition, IRE using Set 2 exhibits a slightly faster runtime than IRE using Set 1.

The results from this test case shows that different design solutions can be obtained using different set of non-essential spans. In this test case, by improving the 'evenness' of the selected MBG for Net B, a slightly better and improved design can be obtained. At least in this one trial case, there is no evidence of great dependence on the choice of MBG. Further investigation in the impact of using different non-essential spans set is required to further outline the criteria for selecting the MBG of the test network that will lead to a better-improved SCIP design.

11.1.2 Future Improvements for iMAPR and PIR heuristics

This work presented a first attempt to formulate a systematic strategy to offer a solution to the topological level of cost optimization in designing ring-based transport networks based on span eliminations. From the understanding of the work presented, it is accepted that the PIR heuristic gave a good compromise between the solution quality and computational time. We also realized that there are situations where using PIR and iMAPR have failed to achieve a certain level of positive cost savings. Further research in PIR and iMAPR is required to improve their effectiveness. The following recommendations summarize several proposed modifications in PIR and iMAPR to date that may improve their productivity in terms of solutions quality.

PIR

i. During Phase I of PIR, supermodular spans are found and supermodular capacities are re-routed where a decision is later made on whether to permanently adopt the resulting graph based on the resulting number of supermodular spans. No re-routing priority is given to whether the supermodular spans are essential spans or non-essential spans. It is conceptually more efficient if the re-routing priority is given to supermodular spans that are also essential spans. Since the non-essentials spans may be eliminated anyway in Phase II of PIR, detouring the supermodular capacities of the supermodular spans that are also non-essential spans in Phase I serves no practical advantage in reducing the number of supermodular spans. It is more efficient if the supermodular capacities of the supermodular spans which are also essential spans are given priority where they are first arranged in descending order of supermodular capacity size and the element with the largest supermodular capacities are re-routed first.

ii. Detouring the supermodular capacities (in Phase I) or the affected demands (in Phase II) at the span-level is relatively inefficient. Span-level re-routing may cause the same unit of demand to be routed over using the same span more than once, falsely increasing the overall average network working capacity and causing a subsequent rise in the cost of ring design. Modifications can be made here by utilizing a path-level re-routing algorithm, where a new set of spans from source to destination is determined to route the given single unit of demand.

iMAPR

i. The pseudo-cost function used in iMAPR (Equations 7.13, 7.14 and Figure 7.1) for multi-modular design implies that the *range* of the pseudo-cost for placing a demand on a span is the same regardless of the cost of the span module. For example, a span has w_i less than 12 and d_i of 15 (refer to Figure 11.1). Therefore, it is assumed that an OC-12 span modular will be placed on this span. Here the range of c_i for this w_i is $15 \leq c_i \leq 150$, which is the same for a span with $12 \leq w_i \leq 48$ where an OC-48 span module is assumed. However, since OC-48 is more expensive than OC-12, demands might first be placed on the OC-48 span module to efficiently utilize the OC-48 span module before choosing to place them on the OC-12 span module. Therefore, the absolute scaling on the range of pseudo-cost must take into consideration the cost of the span module that the span is currently assumed for. This may or may not follow the same cost rule, i.e. two times the cost for four times the capacity. An example of the proposed changes to the iMAPR pseudo-cost using the 2:4 cost rule is shown in Figure 11.1.

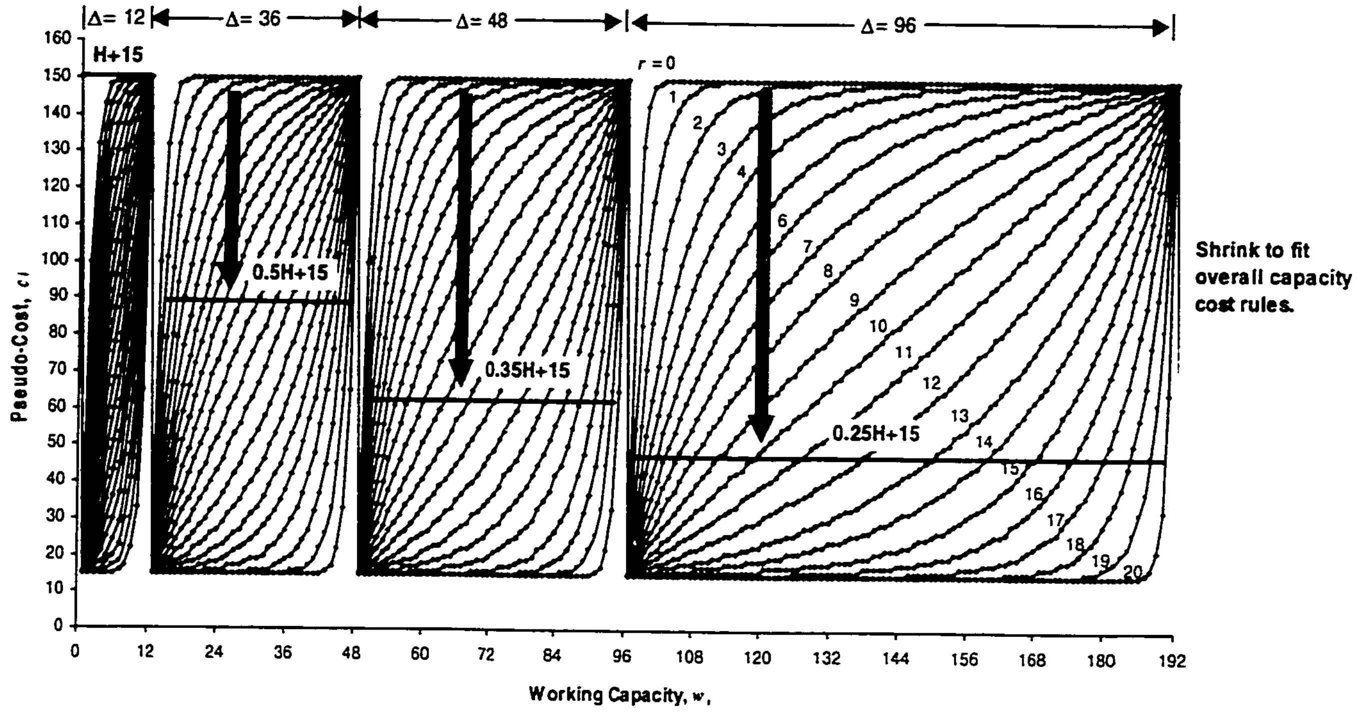


Figure 11.3. The proposed changes to the iMAPR pseudo-cost function with d_i=15.

Bibliography

[1] R. K Butler and D. R. Polson, "Wave-Division Multiplexing in the Sprint Long Distance Network," *IEEE Comm. Mag.*, vol.36, pp. 52-55, Feb. 1998.

[2] J. P. Ryan, "WDM: North American Deployment Trends," *IEEE Comm. Mag.*, vol.36, pp.40-44, Feb. 1998.

[3] E. Lowe, "Current European WDM Deployment Trends," *IEEE Comm. Mag.*, vol.36, pp.46-50, Feb. 1998.

[4] A. Jajszczyk, "What is the Future of Telecommunications Networking?," *IEEE Comm. Mag.*, vol.37, no. 6, pp.12-20, June 1999.

[5] A. G. Mallis, "Reconstructing Transmission Networks Using ATM and DWDM," *IEEE Comm. Mag.*, vol.37, no. 6, pp.140-145, June 1999.

[6] D. Y. Al-Salameh, M. T. Fatehu, W. J. Gartner, S. Lumish, B. L. Nelson, and K. K. Raychaudhuri, "Optical Networking," *Bell Labs Technical Journal*, January - March 1998.

[7] Nortel Networks, "Nortel Networks Introduces World's Fastest, Highest Capacity, Most Reliable Internet Technology," *Nortel Network Press Releases*, May 4, 1999.

[8] Nortel Networks, "MCI WorldCom to Trial World's Highest-Capacity Next-Generation Optical Solution from Nortel Networks," *Nortel Network Press Releases*, May 5, 1999.

[9] M. W. Maeda, "Management and Control of Transparent Optical Networks," *IEEE JSAC*, 16(7), pp. 1005-1023, 1998.

[10] Y. Miyao and H. Saito, "Optimal Design and Evaluation of Survivable WDM Transport Networks," *IEEE JSAC*, 16(7), pp. 1190-1198, 1998.

[11] PMC-Sierra Inc., "Network Survivability Using Automated Protection Switching (APS) Over SONET/SDH Point-to-Point & Ring Networks: Application Note," *PMC-Sierra Protection Switching PM5312*, Issue 3, Feb., 1998.

[12] W. D. Grover, D. Y. Li and M. H. MacGregor, "On Optimized Design of 'Express' Routes in Mesh-Survivable Broadband Networks," *1996 Symposium on Planning and Design of Broadband Networks*, 1996.

[13] E. Traupman, P. O'Connell and J. Minnis, "The Evolution of the Existing Carrier Infrastructure," *IEEE Comm. Mag.*, vol 37, no. 6, pp.134-139, June 1999.

[14] D. Johnson, N. Hayman and P. Veitch, "The Evolution of a Reliable Transport Network," *IEEE Comm. Mag.*, vol 37, no. 8, pp.52-57, August 1999.

[15] G. D. Morley and W. D. Grover, "Current Issues in Design of Optimized Ring-Based Optical Networks," *Canadian Conference of Elec. & Comp. Eng.*, pp. 220-225, May 9-12, 1999.

[16] W. D. Grover, J. B. Slevinsky and M. H. MacGregor, " Optimized Design of Ring-Based Survivable Networks," *Can. J. Elec. & Comp. Eng.*, vol. 20, no. 3, 1995.

[17] D. Morley, "Optimized Ring Network Design", *EE681 Survivable Network Guest Speaker* Presentation Handout, Nov. 13, 1997.

[18] B. Doshi and P. Harshavardhana, " Broadband Network Infrastructure of the Future: Roles of Network Design Tools in Technology Deployment Strategies," *IEEE Comm. Mag.*, vol.39, pp.60-71, May 1998.

[19] O. J. Wasem, T. H. Wu and R. H. Cardwell, " Survivable SONET Networks – Design Methodology," *IEEE JSAC*, vol. 12, no. 1, pp. 205-212, Jan. 1994.

[20] L. M. Gardner, I. H. Sudborough and I. G. Tollis, "Net Solver: A Software Tool for the Design of Survivable Networks," *IEEE GLOBECOMM*, vol. 1, pp. 39-44, Nov 1995.

[21] C. H. Papadimitriou and K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity, Englewood Cliffs, NJ: Prentice Hall, pp.456-462, 1982.

[22] Nortel Networks, "Introduction to SONET Networking," *SONET 101 Tutorial Handbook*, Oct 30, 1996.

[23] T. Flanagan, "Fiber Network Survivability," *IEEE Comm. Mag.*, pp.46-53, June 1990.

[24] C. H. Papadimitriou and K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity, Englewood Cliffs, NJ: Prentice Hall, pp.128-129, 1982.

[25] Johnson, D. B., " Finding All the Elementary Circuits of a Directed Graph," *SIAM J. Comput.*, 4(1975), 77-84.

[26] Reingold E., Nievergelt J., Deo N., Combinatorial Algorithms : Theory and Practice, Prentice-Hall, Section 8.3, pp. 324-353, 1977.

[27] Kennington J.L., Nair V.S.S., Rahman M.H, "Optimization Based Algorithms for Finding Minimal Cost Ring Covers in Survivable Networks," *Technical Report 97 - CSE - 12*, Southern Methodist University, Dallas, Texas, August 1997, pp. 1-26.

[28] R. Fourer, D. M. Gay and B. W. Kernighan, AMPL: A Modeling Language for Mathematical Programming, AT&T Inc., 1993.

[29] "CPLEX Manual," CPLEX Optimization Corp., 1995.

[30] G. D. Morley and W. D. Grover, "A Comparative Survey of Method for Automated Design of Ring-based Transport Networks," TR*Labs* Technical Report TR-97-04, Jan. 28, 1998.

[31] "Misc. Reference Manual Pages for RingBuilder v2.0," TR*Labs* Network System Manuals Library, Feb. 9, 1999.

[32] "RingBuilder^{TM} SONET SHR Design Tools: User's Guide," TR*Labs*, Issue 2.0, 1997.

[33] C. Y. Lee, W. D. Grover and G. D. Morley, " Heuristic Methods for the 'Span Elimination' Problem in Ring-Based Transport Network Design," *Canadian Conference of Elec. & Comp. Eng.*, pp.232-237, May 9-12, 1999.

[34] Morley, G. D. and Grover, W. D, "Comparison of Mathematical Programming Approaches to Optical Ring Network Design," to appear in *Canadian Conference on Broadband Research 1999*, Ottawa, Nov 1999.

[35] Grover, W. D, Rings ... Part 3: Towards Algorithmic Synthesis of Near-Optimal Multi-SHR Networks, *EE681 Class Notes*, TR*Labs*, pp.55 (SHR-15), Fall 1997.

Appendix A

Test Network Standard Network Interface Files (SNIF)

In this appendix, the network topologies for Net A, Net B, Net C and Net D are listed. These topology files are arranged in SNIF format. The SNIF file starts with four description lines detailing Date, File Name, Network and Program. The positions of the network nodes are then listed in the second portion of the SNIF file. The third portion lists all the spans, which are arranged in 6 columns. These columns are span ID, the end nodes, distance, working capacity and the spare capacity of the span.

Network: Net A

Date: February 7, 1997
File Name: bellcore.snif
Network: Bellcore (Net A)
Program:

Node	**Xcoord**	**Ycoord**
1	400	600
2	500	500
3	100	600
4	100	400
5	200	550
6	600	500
7	400	470
8	500	300
9	200	200
10	300	100
11	400	200

Span	**NodeA**	**NodeB**	**Distance**	**Working**	**Spare**
1	1	2	20	0	0
2	1	3	30	0	0
3	1	4	75	0	0
4	1	5	35	0	0
5	1	6	50	0	0
6	1	8	60	0	0
7	2	3	20	0	0
8	3	5	40	0	0
9	4	5	35	0	0
10	4	8	70	0	0
11	4	9	25	0	0
12	5	6	55	0	0
13	5	7	45	0	0
14	5	8	40	0	0
15	5	9	45	0	0
16	5	11	50	0	0
17	6	8	40	0	0
18	7	8	15	0	0
19	8	9	60	0	0
20	8	11	30	0	0
21	9	10	18	0	0
22	9	11	30	0	0
23	10	11	20	0	0

Network: Net B

Date: June 23, 1994
File Name: btrl.snif
Network: British Telecom Study Network (Net B)
Program: None

Node	Xcoord	Ycoord
1	27662	58733
2	36839	59453
3	40193	51952
4	59028	51473
5	45300	48681
6	36839	45327
7	26544	44810
8	31358	32213
9	39315	39740
10	53040	41815
11	60304	44847
12	64071	37623
13	50965	35670
14	62094	28058
15	53441	30961
16	48094	28886
17	40193	25933
18	32545	19089
19	40272	16914
20	48094	18512
21	51893	19353
22	57272	23060
23	57512	17315
24	61578	24480
25	65811	23221
26	65650	17716
27	59652	14602
28	47375	10850
29	47454	15240
30	33568	12448

Span	NodeA	NodeB	Distance	Working	Spare
0	1	2	100	0	0
1	1	3	100	0	0
2	1	7	100	0	0
3	2	4	100	0	0
4	2	3	100	0	0
5	4	11	100	0	0
6	3	5	100	0	0
7	3	4	100	0	0
8	7	6	100	0	0
9	7	8	100	0	0
10	7	3	100	0	0
11	6	5	100	0	0
12	6	9	100	0	0
13	5	11	100	0	0
14	10	5	100	0	0
15	11	10	100	0	0

16	11	12	100	0	0
17	12	10	100	0	0
18	12	13	100	0	0
19	12	14	100	0	0
20	10	8	100	0	0
21	10	13	100	0	0
22	9	13	100	0	0
23	13	15	100	0	0
24	9	15	100	0	0
25	9	17	100	0	0
26	8	18	100	0	0
27	8	17	100	0	0
28	17	18	100	0	0
29	17	16	100	0	0
30	17	19	100	0	0
31	16	15	100	0	0
32	16	21	100	0	0
33	14	24	100	0	0
34	14	22	100	0	0
35	14	15	100	0	0
36	24	25	100	0	0
37	19	20	100	0	0
38	19	28	100	0	0
39	19	30	100	0	0
40	19	18	100	0	0
41	30	18	100	0	0
42	30	29	100	0	0
43	27	28	100	0	0
44	27	26	100	0	0
45	28	23	100	0	0
46	22	25	100	0	0
47	26	23	100	0	0
48	29	20	100	0	0
49	20	21	100	0	0
50	9	18	100	0	0
51	9	20	100	0	0
52	9	10	100	0	0
53	9	8	100	0	0
54	23	29	100	0	0
55	21	22	100	0	0
56	25	26	100	0	0
57	22	23	100	0	0
58	21	23	100	0	0

Network: Net C

Date: Sept 09, 1997
File Name: mci3.4.snif
Network: MCI Telecom Network (Net C)
Program: None

Node	Xcoord	Ycoord
2	167.751	43.9024
3	193.225	67.4797
4	187.805	46.0705
5	175.881	55.2846
7	169.106	19.7832
11	195.122	7.31707
13	18.9702	49.8645
14	80.2168	85.0949
15	15.4472	7.31707
18	146.07	64.2276
29	154.743	19.5122
31	164.499	63.9566
32	126.287	37.3984
33	141.734	28.7263
36	66.1247	75.0678
38	54.7425	18.9702
41	39.5664	61.5176
51	124.932	23.0352
54	131.978	48.5095
59	117.344	93.4959
62	15.7182	18.6992
63	47.4255	106.504
64	63.9566	100.271
65	51.7615	85.6369
70	104.878	66.3957
71	117.886	78.0488
75	97.019	21.9512
77	78.3198	35.7724
78	106.233	85.3659
79	78.0488	71.0027
80	186.992	19.7832
83	122.764	60.4336
84	111.653	37.3984
85	136.856	59.3496
86	41.4634	36.0434
89	70.1897	43.9024
99	58.8076	60.9756
103	39.8374	6.77507
104	39.8374	48.2385
105	23.3062	31.7073
107	57.4526	39.5664
113	94.5799	66.9377
115	162.331	7.31707
118	79.6748	109.756
119	42.5474	75.8808
120	124.932	5.42005
121	92.4119	36.3144
123	104.336	49.3225

124	73.4417	21.4092
125	30.0	45.0

Span	**NodeA**	**NodeB**	**Distance**	**Working**	**Spare**
2	2	5	5	0	0
3	2	7	20	0	0
4	2	29	281	0	0
5	3	4	41	0	0
6	3	5	46	0	0
7	3	11	125	0	0
8	3	31	209	0	0
9	4	80	10	0	0
12	7	80	5	0	0
22	11	115	224	0	0
24	13	119	36	0	0
25	13	125	36	0	0
28	14	36	29	0	0
29	14	118	1	0	0
30	15	62	12	0	0
31	15	103	30	0	0
34	18	31	75	0	0
35	18	85	175	0	0
56	29	33	113	0	0
60	32	33	20	0	0
61	32	54	25	0	0
62	32	84	305	0	0
68	36	89	148	0	0
69	36	118	31	0	0
70	36	119	7	0	0
72	38	62	185	0	0
73	38	124	69	0	0
80	41	99	123	0	0
81	41	104	27	0	0
89	51	75	144	0	0
90	51	120	65	0	0
94	54	85	10	0	0
99	59	71	63	0	0
100	59	118	25	0	0
101	62	77	152	0	0
102	38	103	30	0	0
103	62	105	304	0	0
104	63	64	8	0	0
105	65	119	2	0	0
106	63	118	19	0	0
107	63	119	36	0	0
108	64	65	1	0	0
110	70	83	112	0	0
111	70	113	10	0	0
112	70	123	14	0	0
113	71	78	43	0	0
114	71	79	76	0	0
119	75	124	125	0	0
122	77	79	361	0	0
123	78	79	76	0	0
124	78	118	88	0	0
125	79	113	71	0	0
127	83	84	12	0	0

128	83	85	207	0	0
129	84	121	140	0	0
130	86	105	200	0	0
131	86	107	104	0	0
133	89	104	139	0	0
134	89	107	132	0	0
141	99	119	16	0	0
145	113	121	6	0	0
146	113	123	14	0	0
147	115	120	65	0	0
148	104	125	7	0	0
149	84	51	140	0	0

Network: Net D

Date: Sept 09, 1997
File Name: mci5new.snif
Network: MCI Telecom Network (Net D)
Program: None

Node	Xcoord	Ycoord
2	167.751	43.9024
3	193.225	67.4797
4	187.805	46.0705
5	175.881	55.2846
6	136.314	131.978
7	169.106	19.7832
8	191.599	95.122
9	149.051	94.5799
10	18.1572	99.187
11	195.122	7.31707
13	18.9702	49.8645
14	80.2168	85.0949
15	15.4472	7.31707
18	146.07	64.2276
22	102.71	135.501
23	9.21409	139.024
24	32.7913	158.537
25	139.566	173.984
27	162.602	132.249
28	23.0352	129.81
29	154.743	19.5122
30	7.04607	78.3198
31	164.499	63.9566
32	126.287	37.3984
33	141.734	28.7263
34	68.8347	176.152
36	66.1247	75.0678
37	61.7886	142.005
38	54.7425	18.9702
39	149.051	117.886
40	169.106	102.439
41	39.5664	61.5176
42	164.77	121.68
43	23.0	148.0
45	120.596	131.707
46	162.602	80.7588
48	153.659	172.629
49	81.5718	127.642
51	124.932	23.0352
52	134.417	109.214
53	46.6125	127.642
54	131.978	48.5095
55	22.4932	142.818
56	180.759	80.4878
59	117.344	93.4959
60	148.78	104.878
61	63.6856	128.184

62	15.7182	18.6992
63	47.4255	106.504
64	63.9566	100.271
65	51.7615	85.6369
66	88.8889	189.431
68	21.9512	155.556
70	104.878	66.3957
71	117.886	78.0488
72	119.241	113.008
74	110.569	150.678
75	97.019	21.9512
77	78.3198	35.7724
78	106.233	85.3659
79	78.0488	71.0027
80	186.992	19.7832
81	71.2737	189.702
83	122.764	60.4336
84	111.653	37.3984
85	136.856	59.3496
86	41.4634	36.0434
87	153.93	151.491
88	192.683	81.3008
89	70.1897	43.9024
90	149.593	132.249
91	33.3333	173.442
93	140.108	188.618
95	123.848	162.331
96	28.7263	82.9268
97	121.68	173.984
98	46.3415	165.583
99	58.8076	60.9756
100	106.775	188.618
101	107.317	173.713
102	123.306	189.16
103	39.8374	6.77507
104	39.8374	48.2385
105	23.3062	31.7073
107	57.4526	39.5664
108	102.981	113.008
109	52.3035	176.152
111	177.778	133.333
113	94.5799	66.9377
115	162.331	7.31707
116	189.431	149.051
117	5.14905	95.122
118	79.6748	109.756
119	42.5474	75.8808
120	124.932	5.42005
121	92.4119	36.3144
122	21.4092	116.531
123	104.336	49.3225
124	73.4417	21.4092
125	30.0	45.0
126	140.0	22.0

Span	NodeA	NodeB	Distance	Working	Spare
2	2	5	5	0	0
3	2	7	20	0	0
4	2	29	281	0	0
5	3	4	41	0	0
6	3	5	46	0	0
7	3	11	125	0	0
8	3	31	209	0	0
9	4	80	10	0	0
10	6	90	79	0	0
11	6	45	80	0	0
12	7	80	5	0	0
13	8	9	168	0	0
15	8	88	296	0	0
17	9	31	214	0	0
18	9	46	16	0	0
19	9	108	143	0	0
20	10	96	91	0	0
21	10	117	36	0	0
22	11	115	224	0	0
24	13	104	36	0	0
25	13	125	36	0	0
28	14	36	29	0	0
29	14	118	1	0	0
30	15	62	12	0	0
31	15	103	30	0	0
34	18	31	75	0	0
35	18	85	175	0	0
39	22	45	12	0	0
40	22	49	128	0	0
41	23	28	263	0	0
42	23	55	459	0	0
43	23	122	18	0	0
44	24	68	572	0	0
45	24	98	397	0	0
46	25	48	15	0	0
47	25	93	114	0	0
48	25	95	38	0	0
49	25	97	307	0	0
50	25	102	35	0	0
52	27	90	594	0	0
53	27	111	317	0	0
54	28	53	322	0	0
56	29	33	113	0	0
57	30	62	118	0	0
58	30	96	79	0	0
59	30	117	64	0	0
60	32	33	20	0	0
61	32	54	25	0	0
62	32	84	305	0	0
63	34	66	23	0	0
64	34	81	25	0	0
65	34	101	32	0	0
66	34	109	53	0	0
68	36	89	148	0	0
69	36	118	31	0	0
70	36	119	7	0	0

71	37	53	256	0	0
72	38	62	185	0	0
73	38	124	69	0	0
74	39	40	48	0	0
75	39	42	28	0	0
76	39	49	210	0	0
77	39	60	411	0	0
78	39	90	243	0	0
79	40	42	20	0	0
80	41	99	123	0	0
81	41	104	27	0	0
83	43	68	290	0	0
84	43	55	150	0	0
85	46	56	358	0	0
86	48	87	64	0	0
87	49	61	224	0	0
89	51	75	144	0	0
90	51	120	65	0	0
91	52	60	180	0	0
92	52	72	72	0	0
93	53	61	149	0	0
94	54	85	10	0	0
95	55	68	407	0	0
96	56	88	67	0	0
99	59	71	63	0	0
100	59	118	25	0	0
101	62	77	152	0	0
102	62	103	30	0	0
103	62	105	304	0	0
104	63	64	8	0	0
105	63	65	2	0	0
106	63	118	19	0	0
107	63	119	36	0	0
108	64	65	1	0	0
109	66	100	11	0	0
110	70	83	112	0	0
111	70	113	10	0	0
112	70	123	14	0	0
113	71	78	43	0	0
114	71	79	76	0	0
115	72	108	205	0	0
116	74	87	80	0	0
117	74	95	51	0	0
118	74	109	348	0	0
119	75	124	125	0	0
122	77	79	361	0	0
123	78	79	76	0	0
124	78	118	88	0	0
125	79	113	71	0	0
127	83	84	12	0	0
128	83	85	207	0	0
129	84	121	140	0	0
130	86	105	200	0	0
131	86	107	104	0	0
132	87	116	401	0	0
133	89	104	139	0	0
134	89	107	132	0	0

136	91	98	725	0	0
137	91	109	79	0	0
138	93	102	98	0	0
139	96	122	155	0	0
140	97	101	70	0	0
141	99	119	16	0	0
142	100	101	38	0	0
145	113	121	6	0	0
146	113	123	14	0	0
147	115	120	65	0	0
148	104	125	7	0	0
149	37	115	53	0	0
150	81	91	75	0	0
151	111	116	168	0	0
152	115	126	208	0	0
153	126	33	150	0	0
154	102	97	50	0	0
155	103	38	50	0	0
156	46	31	50	0	0
157	9	40	50	0	0
158	119	65	50	0	0
159	125	105	50	0	0
160	121	123	50	0	0

Appendix B

Span Coverage Integer Program (SCIP): Implementation Validation

In this appendix, the Span Coverage Integer Program (SCIP) formulated in Chapter 3 of this study, SCIP(Chee Yoon), is verified against the original SCIP developed by Morley in [15], SCIP(Morley). Both of these IPs employ the cost model given in Table 3.1. Table B.1 in the appendix shows the test samples and the corresponding SCIP design cost obtained for each IP. Figure B.1 illustrates the graphical depiction of the SCIP Design Cost obtained from each IP vs. Sample No.

In conclusion, Figure B.1 shows a minor difference between the two IPs. The difference lies in the cost of fiber where SCIP(Morley) uses \$10 per unit distance per fiber where as SCIP(Chee Yoon) uses \$10 per unit distance per span module (regardless of the fiber counts in the span module). In terms of the IP itself, both SCIP(Chee Yoon) and SCIP(Morley) are equivalent as shown in Table B.2 where the cost of fiber in SCIP(Morley) is modified to use \$2.5 per unit distance per span module.

Table B.1. Cost results obtained from SCIP(Chee Yoon) and SCIP(Morley).

Sample No.	SCIP(Chee Yoon)	SCIP(Morley)
1	1934871	1961511
2	11076129	11325549
3	1935871	1965511
4	3290098	3316438
5	4648570	4674910
6	6571416	6597756
7	14448650	14994800
8	12483840	12699542
9	15600400	15801970
10	22037754	22239324
11	20113240	20533240
12	17840240	18104240
13	20615904	20831904
14	46359908	49336268
15	33852020	34828550
16	33698130	34330980
17	47573362	48198112

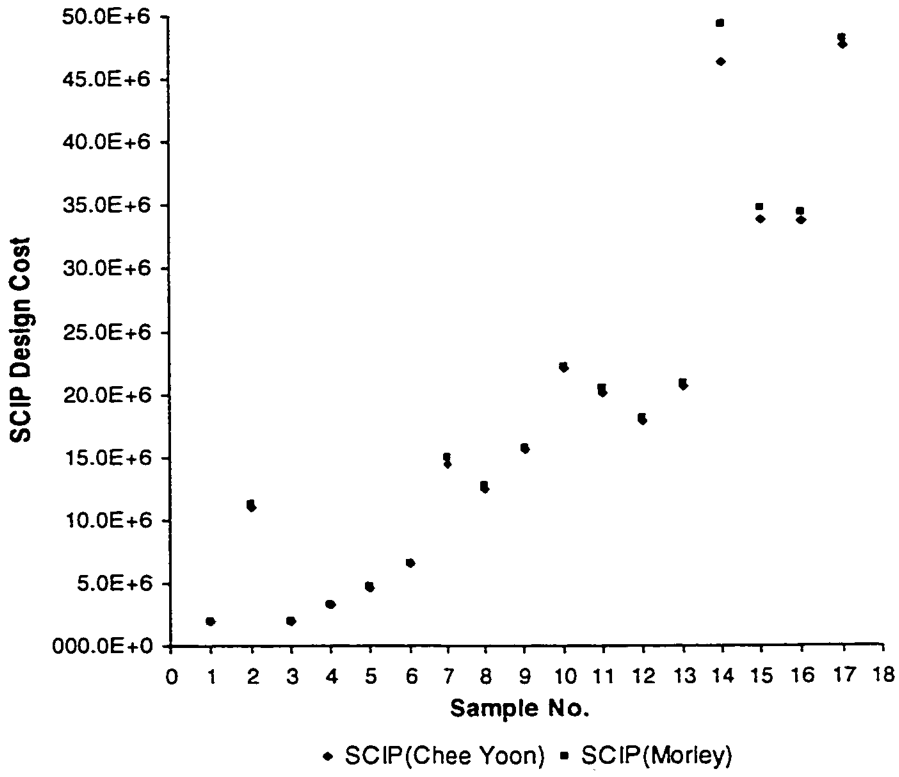


Figure B.1. SCIP Design Cost vs. Sample No. to verify SCIP(Chee Yoon) against SCIP(Morley).

Table B.2. Cost results obtained from SCIP(Chee Yoon) and SCIP(Morley).

Sample No.	SCIP(Chee Yoon)	SCIP(Morley)
1	1934871	1934871
2	11076129	11076129
3	1935871	1935871
4	3290098	3290098
5	4648570	4648570
6	6571416	6571416
7	14448650	14448650
8	12483840	12483840
9	15600400	15600400
10	22037754	22037754
11	20113240	20113240
12	17840240	17840240
13	20615904	20615904
14	46359908	46359908
15	33852020	33852020
16	33698130	33698130
17	47573362	47573362

Appendix C

AMPL Model File for SCIP

This appendix describes the AMPL model file that describes the IP formulations for SCIP (Chapter 3). An exact copy of the AMPL model file is presented together with PERL script used to generate the AMPL data file. An example of the AMPL data file is also presented.

AMPL Model File

```
# AMPL Model for BLSR Network Design
# This AMPL model finds the min-cost set of
# rings (with a given module size) that covers
# the working capacity assigned to each span
# in the network.
#
# By: C.Y. Lee, April 15, 1999.
#

set CYCLE; # cycles
set SPAN; # spans
set CYCLE_SPAN within {CYCLE,SPAN};

param moduleSize{CYCLE} >= 0; # the working capacity of each
ring.
param CycleCost{CYCLE} >= 0;
param working{SPAN} >= 0;

var CycleCopies{CYCLE} integer >=0, <=10;

minimize total_modules:
sum{j in CYCLE} CycleCopies[j]*CycleCost[j];

subject to span_coverage {j in SPAN}:
 sum {(i,j) in CYCLE_SPAN} moduleSize[i] * CycleCopies[i] >=
working[j];
```

PERL Script: Used to generate AMPL Data File

```
#! /usr/local/bin/perl

require 5.004;
use Snif;
use Cycle;
use Text::Wrap qw (wrap $columns);

# Process command line arguments

while ( defined ($myarg = shift) ) {
    if ($myarg =~ /^-c$/i) {
        $cycleFileName = shift;
    } elsif ($myarg =~ /^-n$/i) {
        $snifFileName = shift;
    } elsif ($myarg =~ /^-m$/i) {
        $modSize = shift;
        die "Module Size must be a positive integer!" unless $modSize =~
/^\d+$/;
        die "Module Size must be a positive integer!" unless $modSize > 0;
    } elsif ($myarg =~ /^-z$/i) {
        $maxCycle = shift;
        die "Max cycle size must be a positive integer!:" unless $maxCycle =~
/^\d+$/;
        die "Max cycle size must be at least 3!:" unless $maxCycle >= 3;
```

```
     } elsif ($myarg =~ /^-w$/i) {
         $columns = shift;
         die "display width must be a positive integer!:" unless $columns =~
/^\d+$/;
     }
}

die "Usage: $0 -c cycleFile -n snifFile -m moduleSize -w displayWidth (-z
maxCycleSize)"
    unless (defined $cycleFileName) and (defined $snifFileName) and (defined
$modSize);

# Read in snif file
SNIFReadNetFile ($snifFileName, \$nnodes, \$nspans, \@nodes, \@node2tag,
\@tag2node, \@spans, \@span2tag, \@tag2span);

# Read in cycle file
CYCLEReadCycleFile ($cycleFileName, \@cycles, \@tag2span);

# Define or limit maxCycle to the number of network nodes
$maxCycle = $nnodes if (! defined $maxCycle) or ($maxCycle > $nnodes) ;


# Generate AMPL datafile (printed to STDOUT)

print "param moduleSize :=\n";
$ci = 1;
foreach $i (3 .. $maxCycle) {
    foreach $j (0 .. $#{$cycles[$i]}) {
        print "$ci 12\n";
        $ci++;
    }
}
foreach $i (3 .. $maxCycle) {
    foreach $j (0 .. $#{$cycles[$i]}) {
        print "$ci 48\n";
        $ci++;
    }
}
foreach $i (3 .. $maxCycle) {
    foreach $j (0 .. $#{$cycles[$i]}) {
        print "$ci 96\n";
        $ci++;
    }
}
foreach $i (3 .. $maxCycle) {
    foreach $j (0 .. $#{$cycles[$i]}) {
        print "$ci 192\n";
        $ci++;
    }
}
print "\;\n";

print "set SPAN :=\n";
$line = "";
foreach $i (0 .. $nspans) {
    $line .= "$span2tag[$i] ";
}
print wrap("","",$line);
print "\;\n\n";

print "param working :=\n";
foreach $i (0 .. $nspans) {
    print "$span2tag[$i] $spans[$i]->{WORKING}\n";
}

print "\;\n";
print "set CYCLE :=\n";
$line = "";
$ci = 1;
# do once for each span modular size
foreach $l (1 .. 4){
   foreach $i (3 .. $maxCycle) {
      foreach $j (0 .. $#{$cycles[$i]}) {
          $line .= "$ci ";
          $ci++;
      }
   }
}
print wrap("", "", $line);
print "\;\n\n";

$ci = 1;
print "set CYCLE_SPAN :=\n";
# do once for each span modular size
foreach $l (1 .. 4){
  foreach $i (3 .. $maxCycle) {
      foreach $j (0 .. $#{$cycles[$i]}) {
          $line = "($ci,*) ";
          foreach $k (0 .. $#{$cycles[$i][$j]}) {
              $line .= "$span2tag[$cycles[$i][$j][$k]] ";
          }
          print wrap ("","",$line);
          print "\n";
          $ci++;
      }
  }
}
print "\;\n";



print "param CycleCost :=\n";
$ci = 1;
$n = 0;
$fibercost = 10;
foreach $i (3 .. $maxCycle) {
    foreach $j (0 .. $#{$cycles[$i]}) {

        $cycleADM = $#{$cycles[$i][$j]} + 1;

       foreach $k (0 .. $#{$cycles[$i][$j]}) {
           $m = $spans[$cycles[$i][$j][$k]];
           $n = $n + $m->{DISTANCE};
       }

       $cycleDISTANCE = $n ;

        $n = 0;

        $cycleCost = ($cycleADM * 71333) + ($cycleDISTANCE * 4 * $fibercost);
        print "$ci $cycleCost\n";
        $ci++;
    }
}
foreach $i (3 .. $maxCycle) {
        foreach $j (0 .. $#{$cycles[$i]}) {

            $cycleADM = $#{$cycles[$i][$j]} + 1;

           foreach $k (0 .. $#{$cycles[$i][$j]}) {
               $m = $spans[$cycles[$i][$j][$k]];
               $n = $n + $m->{DISTANCE};
           }

           $cycleDISTANCE = $n ;

            $n = 0;

            $cycleCost = ($cycleADM * 142666) + ($cycleDISTANCE * 4 * $fibercost);
            print "$ci $cycleCost\n";
            $ci++;
        }
    }
    foreach $i (3 .. $maxCycle) {
        foreach $j (0 .. $#{$cycles[$i]}) {

            $cycleADM = $#{$cycles[$i][$j]} + 1;

           foreach $k (0 .. $#{$cycles[$i][$j]}) {
               $m = $spans[$cycles[$i][$j][$k]];
               $n = $n + $m->{DISTANCE};
           }

           $cycleDISTANCE = $n ;

            $n = 0;

            $cycleCost = ($cycleADM * 201730) + ($cycleDISTANCE * 4 * $fibercost);
            print "$ci $cycleCost\n";
            $ci++;
        }
    }
    foreach $i (3 .. $maxCycle) {
        foreach $j (0 .. $#{$cycles[$i]}) {

            $cycleADM = $#{$cycles[$i][$j]} + 1;

           foreach $k (0 .. $#{$cycles[$i][$j]}) {
               $m = $spans[$cycles[$i][$j][$k]];
               $n = $n + $m->{DISTANCE};
           }

           $cycleDISTANCE = $n ;

            $n = 0;

            $cycleCost = ($cycleADM * 285332) + ($cycleDISTANCE * 4 * $fibercost);
            print "$ci $cycleCost\n";
            $ci++;
        }
    }
    print "\;\n";
```

Example of AMPL data file for Net C using multi-modular SCIP

```
param moduleSize :=
1 12
2 12
3 12
4 12
5 12
6 12
7 12
8 12
9 12
10 12
11 12
12 12
13 12
14 12
15 12
16 12
17 12
18 12
19 12
20 12
21 12
22 12
23 12
24 12
25 12
26 12
27 12
28 12
29 12
30 12
31 12
32 12
33 12
34 12
35 12
36 12
37 12
38 12
39 12
40 12
41 12
42 12
43 12
44 12
45 12
46 12
47 12
48 12
49 12
50 12
51 12
52 12
53 12
54 12
55 12
56 12
57 12
58 12
59 12
60 12
61 12
62 12
63 12
64 48
65 48
66 48
67 48
```

```
68 48
69 48
70 48
71 48
72 48
73 48
74 48
75 48
76 48
77 48
78 48
79 48
80 48
81 48
82 48
83 48
84 48
85 48
86 48
87 48
88 48
89 48
90 48
91 48
92 48
93 48
94 48
95 48
96 48
97 48
98 48
99 48
100 48
101 48
102 48
103 48
104 48
105 48
106 48
107 48
108 48
109 48
110 48
111 48
112 48
113 48
114 48
115 48
116 48
117 48
118 48
119 48
120 48
121 48
122 48
123 48
124 48
125 48
126 48
127 96
128 96
129 96
130 96
131 96
132 96
133 96
134 96
135 96
136 96
137 96
138 96
```

```
139 96
140 96
141 96
142 96
143 96
144 96
145 96
146 96
147 96
148 96
149 96
150 96
151 96
152 96
153 96
154 96
155 96
156 96
157 96
158 96
159 96
160 96
161 96
162 96
163 96
164 96
165 96
166 96
167 96
168 96
169 96
170 96
171 96
172 96
173 96
174 96
175 96
176 96
177 96
178 96
179 96
180 96
181 96
182 96
183 96
184 96
185 96
186 96
187 96
188 96
189 96
190 192
191 192
192 192
193 192
194 192
195 192
196 192
197 192
198 192
199 192
200 192
201 192
202 192
203 192
204 192
205 192
206 192
207 192
208 192
209 192
```

```
210 192
211 192
212 192
213 192
214 192
215 192
216 192
217 192
218 192
219 192
220 192
221 192
222 192
223 192
224 192
225 192
226 192
227 192
228 192
229 192
230 192
231 192
232 192
233 192
234 192
235 192
236 192
237 192
238 192
239 192
240 192
241 192
242 192
243 192
244 192
245 192
246 192
247 192
248 192
249 192
250 192
251 192
252 192
;
set SPAN :=
2 3 4 5 6 7 8 9 12 22 24 25 28 29 30 31 34 35 56 60 61 62 68 69 70 72 73 80 81
89 90 94 99 100 101 102 103 104 105 106 107 108 110 111 112 113 114 119 122 123
124 125 127 128 129 130 131 133 134 141 145 146 147 148 149  ;

param working :=
2 17
3 9
4 16
5 7
6 19
7 23
8 11
9 7
12 6
22 22
24 22
25 0
28 38
29 52
30 6
31 0
34 21
35 21
56 18
60 18
61 21
```

```
62 0
68 12
69 0
70 36
72 22
73 24
80 20
81 18
89 23
90 28
94 21
99 14
100 14
101 29
102 0
103 10
104 0
105 20
106 29
107 12
108 16
110 51
111 45
112 8
113 2
114 17
119 27
122 29
123 57
124 53
125 65
127 27
128 35
129 2
130 18
131 19
133 8
134 19
141 26
145 11
146 17
147 28
148 0
149 25

;
set CYCLE :=
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55
56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81
82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105
106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145
146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165
166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185
186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205
206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225
226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245
246 247 248 249 250 251 252 ;

set CYCLE_SPAN :=
(1,*) 30 102 31
(2,*) 113 123 114
(3,*) 111 146 112
(4,*) 28 69 29
(5,*) 104 108 105
(6,*) 24 148 25
(7,*) 99 113 124 100
(8,*) 69 106 107 70
(9,*) 61 94 128 127 62
(10,*) 99 114 123 124 100
```

```
(11,*) 110 127 129 145 111
(12,*) 28 70 107 106 29
(13,*) 2 6 5 9 12 3
(14,*) 110 127 129 145 146 112
(15,*) 68 133 81 80 141 70
(16,*) 61 94 128 110 111 145 129 62
(17,*) 68 133 81 80 141 107 106 69
(18,*) 61 94 128 110 112 146 145 129 62
(19,*) 28 68 133 81 80 141 107 106 29
(20,*) 72 101 122 125 145 129 149 89 119 73
(21,*) 7 22 147 90 149 127 128 35 34 8
(22,*) 2 6 8 34 35 94 61 60 56 4
(23,*) 68 134 131 130 103 101 122 123 124 69
(24,*) 2 6 7 22 147 90 149 62 60 56 4
(25,*) 7 22 147 90 149 62 61 94 35 34 8
(26,*) 2 6 8 34 35 128 127 62 60 56 4
(27,*) 72 101 122 125 111 110 127 149 89 119 73
(28,*) 28 68 134 131 130 103 101 122 123 124 29
(29,*) 68 134 131 130 103 101 122 114 99 100 69
(30,*) 68 134 131 130 103 101 122 114 113 124 69
(31,*) 3 12 9 5 8 34 35 94 61 60 56 4
(32,*) 72 101 122 125 146 112 110 127 149 89 119 73
(33,*) 68 134 131 130 103 101 122 123 124 106 107 70
(34,*) 28 68 134 131 130 103 101 122 114 99 100 29
(35,*) 28 68 134 131 130 103 101 122 114 113 124 29
(36,*) 68 134 131 130 103 101 122 123 113 99 100 69
(37,*) 7 22 147 90 149 129 145 111 110 128 35 34 8
(38,*) 3 12 9 5 7 22 147 90 149 62 60 56 4
(39,*) 3 12 9 5 8 34 35 128 127 62 60 56 4
(40,*) 28 68 134 131 130 103 101 122 123 113 99 100 29
(41,*) 68 134 131 130 103 101 122 114 99 100 106 107 70
(42,*) 68 134 131 130 103 101 122 114 113 124 106 107 70
(43,*) 2 6 7 22 147 90 149 127 128 94 61 60 56 4
(44,*) 7 22 147 90 149 129 145 146 112 110 128 35 34 8
(45,*) 2 6 8 34 35 128 110 111 145 129 62 60 56 4
(46,*) 69 124 123 122 101 103 130 131 134 133 81 80 141 70
(47,*) 80 141 107 106 124 123 122 101 103 130 131 134 133 81
(48,*) 68 134 131 130 103 101 122 123 113 99 100 106 107 70
(49,*) 61 94 128 110 111 125 122 101 72 73 119 89 149 62
(50,*) 2 6 8 34 35 128 110 112 146 145 129 62 60 56 4
(51,*) 28 70 141 80 81 133 134 131 130 103 101 122 123 124 29
(52,*) 69 100 99 114 122 101 103 130 131 134 133 81 80 141 70
(53,*) 69 124 113 114 122 101 103 130 131 134 133 81 80 141 70
(54,*) 80 141 107 106 100 99 114 122 101 103 130 131 134 133 81
(55,*) 80 141 107 106 124 113 114 122 101 103 130 131 134 133 81
(56,*) 61 94 128 110 112 146 125 122 101 72 73 119 89 149 62
(57,*) 3 12 9 5 7 22 147 90 149 127 128 94 61 60 56 4
(58,*) 3 12 9 5 8 34 35 128 110 111 145 129 62 60 56 4
(59,*) 28 70 141 80 81 133 134 131 130 103 101 122 114 99 100 29
(60,*) 28 70 141 80 81 133 134 131 130 103 101 122 114 113 124 29
(61,*) 68 134 131 130 103 72 73 119 89 149 129 145 125 123 124 69
(62,*) 69 100 99 113 123 122 101 103 130 131 134 133 81 80 141 70
(63,*) 80 141 107 106 100 99 113 123 122 101 103 130 131 134 133 81
(64,*) 30 102 31
(65,*) 113 123 114
(66,*) 111 146 112
(67,*) 28 69 29
(68,*) 104 108 105
(69,*) 24 148 25
(70,*) 99 113 124 100
(71,*) 69 106 107 70
(72,*) 61 94 128 127 62
(73,*) 99 114 123 124 100
(74,*) 110 127 129 145 111
(75,*) 28 70 107 106 29
(76,*) 2 6 5 9 12 3
(77,*) 110 127 129 145 146 112
(78,*) 68 133 81 80 141 70
(79,*) 61 94 128 110 111 145 129 62
(80,*) 68 133 81 80 141 107 106 69
(81,*) 61 94 128 110 112 146 145 129 62
```

```
(82,*) 28 68 133 81 80 141 107 106 29
(83,*) 72 101 122 125 145 129 149 89 119 73
(84,*) 7 22 147 90 149 127 128 35 34 8
(85,*) 2 6 8 34 35 94 61 60 56 4
(86,*) 68 134 131 130 103 101 122 123 124 69
(87,*) 2 6 7 22 147 90 149 62 60 56 4
(88,*) 7 22 147 90 149 62 61 94 35 34 8
(89,*) 2 6 8 34 35 128 127 62 60 56 4
(90,*) 72 101 122 125 111 110 127 149 89 119 73
(91,*) 28 68 134 131 130 103 101 122 123 124 29
(92,*) 68 134 131 130 103 101 122 114 99 100 69
(93,*) 68 134 131 130 103 101 122 114 113 124 69
(94,*) 3 12 9 5 8 34 35 94 61 60 56 4
(95,*) 72 101 122 125 146 112 110 127 149 89 119 73
(96,*) 68 134 131 130 103 101 122 123 124 106 107 70
(97,*) 28 68 134 131 130 103 101 122 114 99 100 29
(98,*) 28 68 134 131 130 103 101 122 114 113 124 29
(99,*) 68 134 131 130 103 101 122 123 113 99 100 69
(100,*) 7 22 147 90 149 129 145 111 110 128 35 34 8
(101,*) 3 12 9 5 7 22 147 90 149 62 60 56 4
(102,*) 3 12 9 5 8 34 35 128 127 62 60 56 4
(103,*) 28 68 134 131 130 103 101 122 123 113 99 100 29
(104,*) 68 134 131 130 103 101 122 114 99 100 106 107 70
(105,*) 68 134 131 130 103 101 122 114 113 124 106 107 70
(106,*) 2 6 7 22 147 90 149 127 128 94 61 60 56 4
(107,*) 7 22 147 90 149 129 145 146 112 110 128 35 34 8
(108,*) 2 6 8 34 35 128 110 111 145 129 62 60 56 4
(109,*) 69 124 123 122 101 103 130 131 134 133 81 80 141 70
(110,*) 80 141 107 106 124 123 122 101 103 130 131 134 133 81
(111,*) 68 134 131 130 103 101 122 123 113 99 100 106 107 70
(112,*) 61 94 128 110 111 125 122 101 72 73 119 89 149 62
(113,*) 2 6 8 34 35 128 110 112 146 145 129 62 60 56 4
(114,*) 28 70 141 80 81 133 134 131 130 103 101 122 123 124 29
(115,*) 69 100 99 114 122 101 103 130 131 134 133 81 80 141 70
(116,*) 69 124 113 114 122 101 103 130 131 134 133 81 80 141 70
(117,*) 80 141 107 106 100 99 114 122 101 103 130 131 134 133 81
(118,*) 80 141 107 106 124 113 114 122 101 103 130 131 134 133 81
(119,*) 61 94 128 110 112 146 125 122 101 72 73 119 89 149 62
(120,*) 3 12 9 5 7 22 147 90 149 127 128 94 61 60 56 4
(121,*) 3 12 9 5 8 34 35 128 110 111 145 129 62 60 56 4
(122,*) 28 70 141 80 81 133 134 131 130 103 101 122 114 99 100 29
(123,*) 28 70 141 80 81 133 134 131 130 103 101 122 114 113 124 29
(124,*) 68 134 131 130 103 72 73 119 89 149 129 145 125 123 124 69
(125,*) 69 100 99 113 123 122 101 103 130 131 134 133 81 80 141 70
(126,*) 80 141 107 106 100 99 113 123 122 101 103 130 131 134 133 81
(127,*) 30 102 31
(128,*) 113 123 114
(129,*) 111 146 112
(130,*) 28 69 29
(131,*) 104 108 105
(132,*) 24 148 25
(133,*) 99 113 124 100
(134,*) 69 106 107 70
(135,*) 61 94 128 127 62
(136,*) 99 114 123 124 100
(137,*) 110 127 129 145 111
(138,*) 28 70 107 106 29
(139,*) 2 6 5 9 12 3
(140,*) 110 127 129 145 146 112
(141,*) 68 133 81 80 141 70
(142,*) 61 94 128 110 111 145 129 62
(143,*) 68 133 81 80 141 107 106 69
(144,*) 61 94 128 110 112 146 145 129 62
(145,*) 28 68 133 81 80 141 107 106 29
(146,*) 72 101 122 125 145 129 149 89 119 73
(147,*) 7 22 147 90 149 127 128 35 34 8
(148,*) 2 6 8 34 35 94 61 60 56 4
(149,*) 68 134 131 130 103 101 122 123 124 69
(150,*) 2 6 7 22 147 90 149 62 60 56 4
(151,*) 7 22 147 90 149 62 61 94 35 34 8
(152,*) 2 6 8 34 35 128 127 62 60 56 4
```

```
(153,*) 72 101 122 125 111 110 127 149 89 119 73
(154,*) 28 68 134 131 130 103 101 122 123 124 29
(155,*) 68 134 131 130 103 101 122 114 99 100 69
(156,*) 68 134 131 130 103 101 122 114 113 124 69
(157,*) 3 12 9 5 8 34 35 94 61 60 56 4
(158,*) 72 101 122 125 146 112 110 127 149 89 119 73
(159,*) 68 134 131 130 103 101 122 123 124 106 107 70
(160,*) 28 68 134 131 130 103 101 122 114 99 100 29
(161,*) 28 68 134 131 130 103 101 122 114 113 124 29
(162,*) 68 134 131 130 103 101 122 123 113 99 100 69
(163,*) 7 22 147 90 149 129 145 111 110 128 35 34 8
(164,*) 3 12 9 5 7 22 147 90 149 62 60 56 4
(165,*) 3 12 9 5 8 34 35 128 127 62 60 56 4
(166,*) 28 68 134 131 130 103 101 122 123 113 99 100 29
(167,*) 68 134 131 130 103 101 122 114 99 100 106 107 70
(168,*) 68 134 131 130 103 101 122 114 113 124 106 107 70
(169,*) 2 6 7 22 147 90 149 127 128 94 61 60 56 4
(170,*) 7 22 147 90 149 129 145 146 112 110 128 35 34 8
(171,*) 2 6 8 34 35 128 110 111 145 129 62 60 56 4
(172,*) 69 124 123 122 101 103 130 131 134 133 81 80 141 70
(173,*) 80 141 107 106 124 123 122 101 103 130 131 134 133 81
(174,*) 68 134 131 130 103 101 122 123 113 99 100 106 107 70
(175,*) 61 94 128 110 111 125 122 101 72 73 119 89 149 62
(176,*) 2 6 8 34 35 128 110 112 146 145 129 62 60 56 4
(177,*) 28 70 141 80 81 133 134 131 130 103 101 122 123 124 29
(178,*) 69 100 99 114 122 101 103 130 131 134 133 81 80 141 70
(179,*) 69 124 113 114 122 101 103 130 131 134 133 81 80 141 70
(180,*) 80 141 107 106 100 99 114 122 101 103 130 131 134 133 81
(181,*) 80 141 107 106 124 113 114 122 101 103 130 131 134 133 81
(182,*) 61 94 128 110 112 146 125 122 101 72 73 119 89 149 62
(183,*) 3 12 9 5 7 22 147 90 149 127 128 94 61 60 56 4
(184,*) 3 12 9 5 8 34 35 128 110 111 145 129 62 60 56 4
(185,*) 28 70 141 80 81 133 134 131 130 103 101 122 114 99 100 29
(186,*) 28 70 141 80 81 133 134 131 130 103 101 122 114 113 124 29
(187,*) 68 134 131 130 103 72 73 119 89 149 129 145 125 123 124 69
(188,*) 69 100 99 113 123 122 101 103 130 131 134 133 81 80 141 70
(189,*) 80 141 107 106 100 99 113 123 122 101 103 130 131 134 133 81
(190,*) 30 102 31
(191,*) 113 123 114
(192,*) 111 146 112
(193,*) 28 69 29
(194,*) 104 108 105
(195,*) 24 148 25
(196,*) 99 113 124 100
(197,*) 69 106 107 70
(198,*) 61 94 128 127 62
(199,*) 99 114 123 124 100
(200,*) 110 127 129 145 111
(201,*) 28 70 107 106 29
(202,*) 2 6 5 9 12 3
(203,*) 110 127 129 145 146 112
(204,*) 68 133 81 80 141 70
(205,*) 61 94 128 110 111 145 129 62
(206,*) 68 133 81 80 141 107 106 69
(207,*) 61 94 128 110 112 146 145 129 62
(208,*) 28 68 133 81 80 141 107 106 29
(209,*) 72 101 122 125 145 129 149 89 119 73
(210,*) 7 22 147 90 149 127 128 35 34 8
(211,*) 2 6 8 34 35 94 61 60 56 4
(212,*) 68 134 131 130 103 101 122 123 124 69
(213,*) 2 6 7 22 147 90 149 62 60 56 4
(214,*) 7 22 147 90 149 62 61 94 35 34 8
(215,*) 2 6 8 34 35 128 127 62 60 56 4
(216,*) 72 101 122 125 111 110 127 149 89 119 73
(217,*) 28 68 134 131 130 103 101 122 123 124 29
(218,*) 68 134 131 130 103 101 122 114 99 100 69
(219,*) 68 134 131 130 103 101 122 114 113 124 69
(220,*) 3 12 9 5 8 34 35 94 61 60 56 4
(221,*) 72 101 122 125 146 112 110 127 149 89 119 73
(222,*) 68 134 131 130 103 101 122 123 124 106 107 70
(223,*) 28 68 134 131 130 103 101 122 114 99 100 29
```

```
(224,*) 28 68 134 131 130 103 101 122 114 113 124 29
(225,*) 68 134 131 130 103 101 122 123 113 99 100 69
(226,*) 7 22 147 90 149 129 145 111 110 128 35 34 8
(227,*) 3 12 9 5 7 22 147 90 149 62 60 56 4
(228,*) 3 12 9 5 8 34 35 128 127 62 60 56 4
(229,*) 28 68 134 131 130 103 101 122 123 113 99 100 29
(230,*) 68 134 131 130 103 101 122 114 99 100 106 107 70
(231,*) 68 134 131 130 103 101 122 114 113 124 106 107 70
(232,*) 2 6 7 22 147 90 149 127 128 94 61 60 56 4
(233,*) 7 22 147 90 149 129 145 146 112 110 128 35 34 8
(234,*) 2 6 8 34 35 128 110 111 145 129 62 60 56 4
(235,*) 69 124 123 122 101 103 130 131 134 133 81 80 141 70
(236,*) 80 141 107 106 124 123 122 101 103 130 131 134 133 81
(237,*) 68 134 131 130 103 101 122 123 113 99 100 106 107 70
(238,*) 61 94 128 110 111 125 122 101 72 73 119 89 149 62
(239,*) 2 6 8 34 35 128 110 112 146 145 129 62 60 56 4
(240,*) 28 70 141 80 81 133 134 131 130 103 101 122 123 124 29
(241,*) 69 100 99 114 122 101 103 130 131 134 133 81 80 141 70
(242,*) 69 124 113 114 122 101 103 130 131 134 133 81 80 141 70
(243,*) 80 141 107 106 100 99 114 122 101 103 130 131 134 133 81
(244,*) 80 141 107 106 124 113 114 122 101 103 130 131 134 133 81
(245,*) 61 94 128 110 112 146 125 122 101 72 73 119 89 149 62
(246,*) 3 12 9 5 7 22 147 90 149 127 128 94 61 60 56 4
(247,*) 3 12 9 5 8 34 35 128 110 111 145 129 62 60 56 4
(248,*) 28 70 141 80 81 133 134 131 130 103 101 122 114 99 100 29
(249,*) 28 70 141 80 81 133 134 131 130 103 101 122 114 113 124 29
(250,*) 68 134 131 130 103 72 73 119 89 149 129 145 125 123 124 69
(251,*) 69 100 99 113 123 122 101 103 130 131 134 133 81 80 141 70
(252,*) 80 141 107 106 100 99 113 123 122 101 103 130 131 134 133 81
;
param CycleCost :=
1 214719
2 215949
3 214379
4 214609
5 214109
6 214789
7 287522
8 286262
9 362255
10 359945
11 359465
12 357585
13 429268
14 430978
15 432598
16 578814
17 576054
18 650327
19 647377
20 727260
21 726300
22 722920
23 729290
24 798553
25 798843
26 799143
27 798473
28 800613
29 800623
30 801053
31 865836
32 869986
33 872266
34 871946
35 872376
36 872386
37 942859
38 941469
39 942059
40 943709
```

```
41 943599
42 944029
43 1012042
44 1014372
45 1015702
46 1016262
47 1016432
48 1015362
49 1017822
50 1087215
51 1087585
52 1087595
53 1088025
54 1087765
55 1088195
56 1089335
57 1154958
58 1158618
59 1158918
60 1159348
61 1160958
62 1159358
63 1159528
64 428718
65 429948
66 428378
67 428608
68 428108
69 428788
70 572854
71 571594
72 718920
73 716610
74 716130
75 714250
76 857266
77 858976
78 860596
79 1149478
80 1146718
81 1292324
82 1289374
83 1440590
84 1439630
85 1436250
86 1442620
87 1583216
88 1583506
89 1583806
90 1583136
91 1585276
92 1585286
93 1585716
94 1721832
95 1725982
96 1728262
97 1727942
98 1728372
99 1728382
100  1870188
101  1868798
102  1869388
103  1871038
104  1870928
105  1871358
106  2010704
107  2013034
108  2014364
109  2014924
110  2015094
111  2014024
```

112	2016484
113	2157210
114	2157580
115	2157590
116	2158020
117	2157760
118	2158190
119	2159330
120	2296286
121	2299946
122	2300246
123	2300676
124	2302286
125	2300686
126	2300856
127	605910
128	607140
129	605570
130	605800
131	605300
132	605980
133	809110
134	807850
135	1014240
136	1011930
137	1011450
138	1009570
139	1211650
140	1213360
141	1214980
142	1621990
143	1619230
144	1823900
145	1820950
146	2031230
147	2030270
148	2026890
149	2033260
150	2232920
151	2233210
152	2233510
153	2232840
154	2234980
155	2234990
156	2235420
157	2430600
158	2434750
159	2437030
160	2436710
161	2437140
162	2437150
163	2638020
164	2636630
165	2637220
166	2638870
167	2638760
168	2639190
169	2837600
170	2839930
171	2841260
172	2841820
173	2841990
174	2840920
175	2843380
176	3043170
177	3043540
178	3043550
179	3043980
180	3043720
181	3044150
182	3045290

```
183 3241310
184 3244970
185 3245270
186 3245700
187 3247310
188 3245710
189 3245880
190 856716
191 857946
192 856376
193 856606
194 856106
195 856786
196 1143518
197 1142258
198 1432250
199 1429940
200 1429460
201 1427580
202 1713262
203 1714972
204 1716592
205 2290806
206 2288046
207 2576318
208 2573368
209 2867250
210 2866290
211 2862910
212 2869280
213 3152542
214 3152832
215 3153132
216 3152462
217 3154602
218 3154612
219 3155042
220 3433824
221 3437974
222 3440254
223 3439934
224 3440364
225 3440374
226 3724846
227 3723456
228 3724046
229 3725696
230 3725586
231 3726016
232 4008028
233 4010358
234 4011688
235 4012248
236 4012418
237 4011348
238 4013808
239 4297200
240 4297570
241 4297580
242 4298010
243 4297750
244 4298180
245 4299320
246 4578942
247 4582602
248 4582902
249 4583332
250 4584942
251 4583342
252 4583512
;
```

Appendix D

MAPR(fall)

This appendix describes MAPR using *fall* pseudo-cost cost and the initial experiments that were performed using this method. The results showed no reasonable improvement in design cost and therefore fall pseudo-cost was removed from subsequent work in MAPR.

Pseudo-cost function, *fall*, is described by Eq.D.1 and an example of this pseudo-cost is shown in Figure D.1 below.

$$c_i(w_i) = \begin{cases} d_i \cdot M & (\, w_i \bmod M \,) = 0\,; \\ M - (w_i \bmod M) \cdot d_i & (\, w_i \bmod M \,) \neq 0\,; \end{cases} \qquad \textbf{(D.1)}$$

$c_i(w_i)$	- Pseudo-cost of span *i*.
d_i	- Geographical distance of span *i*.
w_i	- Accumulated demands crossing span *i*.
M	- Span modularity constant.

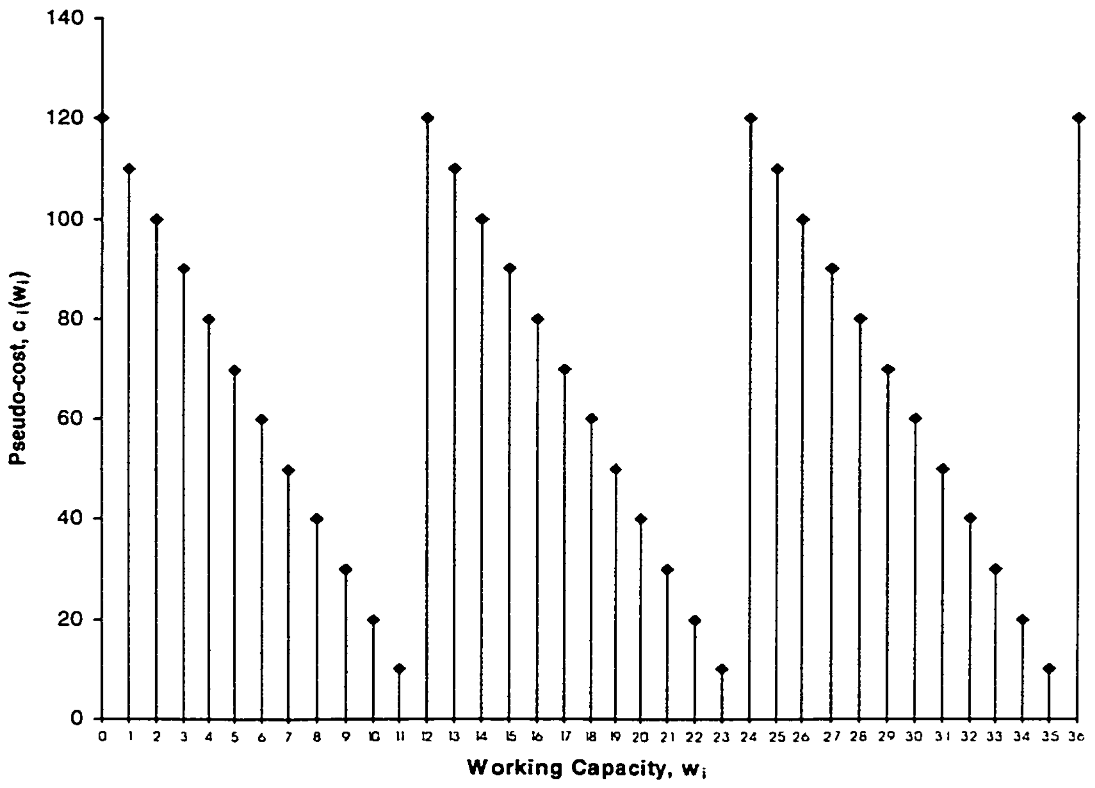


Figure D.1. Pseudo-cost function *fall* for d_i=10 and M=12.

The results of implementing MAPR(fall) on Net A, Net C and Net D are shown in Table D.1 below. Other results using rise and impulse pseudo-cost function are also included. The *No. of Unused Span* column is the number of spans that are found to have zero working capacity in the resulting capacitated graph after all the demands have been routed using the respective routing principles. The *Required No. of Span Modules* show the number of span modules that are required in the resulting min-cost ring coverage design.

Table D.1. MAPR results.

Network	MAPR Pseudo-cost	Design Modularity	# unused spans	Required No. of Span Modules
A	s.path	12	3	23
		24	3	20
	rise	12	6	21
		24	9	14
	impulse	12	11	22
		24	11	14
	fall	12	6	21
		24	6	18
C	s.path	48	7	62
		96	7	57
	rise	48	9	57
		96	15	49
	impulse	48	14	61
		96	17	47
	fall	48	8	64
		96	8	56
D	s.path	96	7	118
		192	7	118
	rise	96	21	104
		192	24	101
	impulse	96	25	137
		192	31	94
	fall	96	20	122
		192	23	117

As shown in Table D.1, MAPR(rise) and MAPR(impulse) in general out-performed MAPR(fall) except for Net A with design modularity 12, where MAPR(impulse) require a single span module more than MAPR(fall). All of the results by MAPR(rise) show improvements over shortest-path routing. Five out of six MAPR(impulse) results show improvements and four out of six MAPR(fall) results give improvements over shortest-path results. Therefore, with these results, the focus became rise and impulse, and fall was removed from further studies.