A Methodology for the Design of Multi-Channel Network Architectures

by

Mohammed Abdul Azeem Abed

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

INFORMATION AND COMPUTER SCIENCE

July, 1993

A methodology for the design of multi-channel network architectures

Abed, Mohammed Abdul Azeem, M.S.

King Fahd University of Petroleum and Minerals (Saudi Arabia), 1993

A METHODOLOGY FOR THE DESIGN OF MULTI-CHANNEL NETWORK ARCHITECTURES

Mohammed Abdul Azeem Abed

Information and Computer Science

July 1993

Dedicated to

My
Parents,
Sisters, and Brothers

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS

DHAHRAN, SAUDI ARABIA

This thesis, written by

Mohammed Abdul Azeem Abed

under the direction of his Thesis Advisor, and approved by his Thesis Committe, has been presented to and accepted by the Dean, College of Graduate Studies, in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

Thesis Committee:

S. Ghanta

Chairman (Dr. Subbarao Ghanta)

M Bozyigit

Member (Dr. Muslim Bozyigit)

Mohsen Guizani

Member (Dr. Mohsen Guizani)

Shakil Akhter

Member (Dr. Shakil Akhter)

[illegible]

Department Chairman

[illegible]

Dean, College of Graduate Studies

Date: 4/8/93

عمادة الدراسات العليا

COLL

Acknowledgment

In the name of Allah, Most Gracious, Most Merciful.

Read in the name of thy Lord and Cherisher, Who created. Created man from a {leech-like} *clot. Read and thy Lord is Most Bountiful. He Who taught {the use of} the pen. Taught man that which he knew not. Nay, but man doth transgress all bounds. In that he looketh upon himself as self-sufficient. Verily, to thy Lord is the return {of all}.*

(The Holy Quran, Surah 96)

Praise be to ALLAH for giving me the courage and patience to carry out this work. I am happy to have had a chance to glorify His name in the sincerest way through this small accomplishment and ask Him to accept my efforts. May He guide us and the whole humanity to the right path *(Aameen).*

Peace and mercy be upon His Holy Prophet.

Acknowledgment is due to King Fahd University of Petroleum and Minerals for support of this research.

There have been several who influenced my work and life positively, and I consider myself fortunate to have known them. Foremost among them is my family - my mother, my father, my sisters and my brothers.

People like my advisor, Dr.Ghanta are rare. There is so much to learn from them. I sincerely hope I have made good use of my opportunity. I thank Dr.Ghanta for his encouragement and inspiration. I also wish to thank my committee members Dr.Bozyigit, Dr.Mohsen, and Dr.Shakil for their active support, and valuable suggestions.

Special thanks are due to my brother Khalid, for his help, trust, and encouragement. There have been several friends of mine who have made direct and indirect contributions to my work and life at KFUPM. Among them Ismail, Neaz, Masud, Asaf, Belal, Sami, Idrees, Adel, Salah, Mahmood, Jaweed, Yahya, Garout, Husni, Shahid, Emad, Imad, Farooq, Dikko, Amir, Atif, Yousuf, Wasim, Ahmad deserve special thanks.

So, I conclude by acknowledging that this thesis and its author have benefited a great deal from the contributions of several individuals that remain unnamed in this acknowledgement.

Contents

List of Figures

List of Tables

Abstract

Name: MOHAMMED ABDUL AZEEM ABED

Title: A METHODOLOGY FOR THE DESIGN OF MULTI-CHANNEL NETWORK ARCHITECTURES.

Major Field: COMPUTER SCIENCE

Date of Degree: JULY 1993

Two emerging architectures are opening up new opportunities in distributed network architectures. The first one is the feasibility of optical fibers that can offer enormous bandwidth. The second one is the availability of real-time tunable transceivers. Suitable architectures are needed to utilize these potential technologies. In this thesis design of multi-channel network architectures (for local and metropolitan area networks) is considered. The connectivity that represents the way station's transmitters and receivers are tuned to different bands is the virtual topology of the multi-channel network architecture. Further the tunability of transceivers permits the virtual topology to be updated in response to changing traffic patterns. The large scale design of multi-channel network architecture addresses a number of problems: the design of physical and virtual topologies, and the assignment of traffic flows to channels i.e., optimal routing. In this thesis these problems are formally proposed and solved. Main focus is on the virtual topology design problem that maximizes the sum of total traffic flow and minimum channel flow. Using the optimization technique of genetic algorithms, the virtual topology design problem problem has been solved. The results obtained are encouraging in the sense that they are near optimal. Design of a distributed heuristic for virtual topology design problem is also discussed. A methodology for the design of multi-channel network architectures is devised. Finally observations with the genetic (optimization) algorithm are discussed and continuing efforts in this area are outlined.

Keywords: *Tunable Transceivers, Multi-Channel Architectures, Physical Topology, Virtual Topology, Genetic Algorithm, Distributed Virtual Topology Design Heuristic.*

Master of Science Degree
King Fahd University of Petroleum and Minerals, Dhahran
July 1993

خلاصة الرسالة

اسم الطالـــب	:	محمد عبدالعظيم عابد
عنوان الرسالة	:	اطار لتصميم تركيبات لشبكة متعددة المسارات .
التخصـــص	:	هندسة الحاسب الآلي .
تاريخ الشهادة	:	يوليو ١٩٩٣ م .

هناك حاجة ماسة لاستقلال عرض النطاق الترددى الواسع (فى حالة الالياف الضوئية) وكذلك التكنولوجيا المتغيرة . لقد تم تصميم تركيبة لشبكة متعددة المسارات (الشبكات المحلية والحضرية) فى هذه الرسالة .

لقد تم استخدام التركيبه الظاهرة لتصاميم الشبكات المتعددة المسارات وذلك فى المستوى التجريدى لمستقبلات ومرسلات المحطة . اضافة فان تناغمية المرسل - المستقبل تسمح للتركيبة الظاهرة أن تتوافق مع التغيرات فى الاشكال المرورية . لقد تم اقتراح وحل المشاكل المرطبة بالطبولوجيا الحقيقية والظاهرة فى هذه الرسالة وكانت النتائج تقريباً مثالية . ولقد تم مناقشة مشكلة طريقة التصميم على التركيبة الظاهرة الموزعة . واخيراً فقد تم استنباط اطار لتصميم تركيبات الشبكات المتعددة المسارات .

المداخل : مرسلات ومستقبلات متغيرة ، تركيبة متعددة المسارات ، التركيبة الحقيقية ، التركيبة الظاهرة ، لوغاريتميه ورائيه ، طريقة تصميم على التركيبة ، الظاهرة الموزعة .

درجة الماجستير في العلوم

جامعة الملك فهد للبترول والمعادن

الظهران ، المملكة العربية السعودية

يوليو ١٩٩٣ م

Chapter 1

Introduction

Chapter Synopsis: *Developments in the field of optical communication are discussed. Motivation for the work, objectives of the thesis, and background on lightwave networks are presented.*

1.1 Introduction and Motivation

Over the last twenty years optical fiber has emerged as the transmission medium of choice for high speed transmission. Optical fiber has had impact on all branches of information transmission. Optical fiber is taking over the role of copper in case of guided transmission and radio for unguided free space transmission. The developments in the field of lightwave networks during the past two-three decades, allow one to extrapolate that there are many unexploited opportunities. Optical fiber offers large bandwidth that is about *10 orders of magnitude* greater than that of first generation phone lines. A number of research organizations worldwide are studying the possibility of all-optical networks. These networks would be capable of tapping a good portion of the approximately 50 Terahertz (THz) bandwidth or so available in the lightwave range of frequency spectrum. This represents atleast three orders of magnitude more than all of the current radio spectrum upto and including the microwave band. The bit error rate offered by the optical fiber is also about *ten orders of magnitude* better than that of first generation phone lines.

A variety of fascinating applications come to mind when one visualizes the significance of opening up this relatively untapped reservoir of bandwidth. Wavelength division multiplexing (WDM) allows these applications to be realized by providing the required bandwidth on optical links. But the main bottleneck is at nodes since the existing switching process, and buffering technologies lag behind the transmission capabilities [1]. WDM again provides a solution by simplifying switching. Here time slots and data channels are utilized such that identification of header is

1.2 Generations of Lightwave Networks

The growing research awareness in the field of lightwave networks has resulted in a number of ongoing activities in various research laboratories which are leading to what is called third generation lightwave networks. Figure 1.1 shows different generations of lightwave networks. In the first generation all links are copper and conversions between waveform and bits takes place at every node. In the second generation, all links are fiber (whose bit rate and bit error rate performance is improved, but whose propagation latency is unchanged, as schematized in the diagram by links that are fatter but of the same length). In the third generation networks, there are no electrical to optical conversions except at the end of a logical connection.

1.3 Suitable Architectures are Missing

In the second generation networks, with the use of fiber instead of copper we gain in bandwidth but the communication system esentially remained the same. New network architectures and protocols are needed to fully exploit the special properties of fiber, to satisfy the emerging applications, and to reach to the user expectations and perceptions. Reorganization of protocol layers is needed. Because of the low error rate and the high bandwidth offered by the fiber, the low level protocols that are meant for error masking purposes become redundant and unneccessary. New type of protocols called *lean protocols* which are a direct consequence of large bandwidth and low error rate offered by the fiber needs to be designed.

In the short term, we are limited by technology, and several devices will have to be made cost effective. In the longer term, we will have to develop radically new network architectures and protocols and tailor them to the emerging high-bandwidth applications. Many interesting problems in lightwave networks remains to be solved such as distributed routing, management and control of wavelength routing networks,

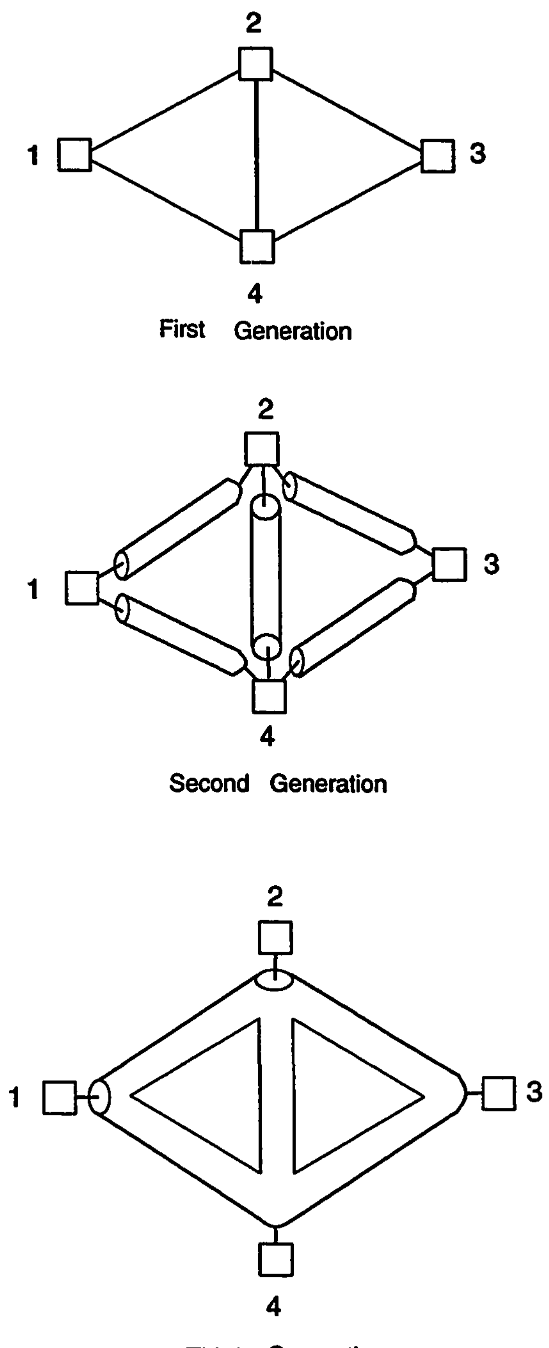


Figure 1.1: Lightwave Network Generations

interconnection of Local Area Networks (LANs) and Metropolitan Area Networks (MANs) using Wide Area Networks (WANs).

1.4 Applications

Many modern applications need high bandwidth for supporting multi-media services i.e., voice, video and data transmission at the same time. Super computer visualization is another application. High bandwidth is needed to interconnect super computers to workstations, where users can see and manipulate full-motion color graphics showing results of their calculations. In the field of medicine, medical imaging, requires high-resolution images to be transmitted uncompressed which requires enormous bandwidth. Past experience indicates that users of such networks, whether at home, business, or institution, will themselves develop undreamed of applications once presented with this wideband capability [30].

1.5 Objectives of the Thesis Work

The objectives of this thesis work are:

1. To explore:

- advances in the field of lightwave networks
- immediate problems being faced by the designers in the field of lightwave networks.

2. To find out potential problems of research in the field of lightwave networks in general and design of lightwave networks in particular.

3. To formulate lightwave network design problems, for fast changing traffic i.e., formulation of problem for logically rearrangeable lightwave networks.

4. To develop solution techniques for near optimal virtual topology design of lightwave networks under the constraints of connectivity between two nodes, maximum channel flow, etc.

6. To give a distributed virtual topology design heuristic useful in real life situation. To explore the use of distributed genetic algorithms for this purpose.

7. To provide a methodology for the design of lightwave networks.

1.6 Background

The multi-channel architecture is based on the efficient use of data-channels formed as a result of bandwidth partitioning. A data channel is established between any two nodes of the network by the assignment of wavelength. The available wavelengths are limited, so it is not possible to establish a data-channel between each pair of nodes [14]. As a result only a few nodes selected through some criteria of importance can be assigned wavelengths thus leading to a topology that is not fully connected.

Today the progress in the lightwave networks is limited mostly by technology available [27] and the cost consideration. The wishlist of components includes cheap fixed-tuned and rapidly tunable lasers and filters with narrow spectral widths, wideband optical amplifiers, lowloss optical switches, etc. Overall, having a few hundred optical channels within a 30 to 40 nm band around 1.5 micro meter appears to be quite feasible. Tuning rapidly between these channels also appears to be feasible in the near future.

In the short term, optics appears to be better suited for transmission and circuit-switching, rather than packet switching, and it appears that packet switching in these optical networks will have to be done electronically using the multihop approach. This may change in the future as components evolve. A truly all-optical switch, where control is also done optically, appears to be far away on the horizon [27]. Much work needs to be done to develop *slim or lean protocols* (discussed in Section 1.3) that can support a wide variety of services with widely varying requirements. New multiaccess protocols that provide these functions are just beginning to be developed for these networks.

1.7 Outline of the Thesis

Chapter 2 deals with a preview of previous work carried out in the field of lightwave networks. Chapter 3 discusses multi-channel architectures in detail. Chapter 4 presents a design methodology for the lightwave networks. Chapter 5 deals with formulation of design problems in multi-channel network architectures. Chapter 6 presents a genetic algorithm for solving virtual topology design problem. The validation and experimental results as well as results of various comparisons made are all provided in Chapter 7. A distributed virtual topology design heuristic is given in Chapter 8. The final Chapter is about the contributions made in the thesis, the conclusions drawn as well as suggestions for future research.

Chapter 2

Background Work

Chapter Synopsis: *Literature on lightwave networks is surveyed. Introduction to networks and protocols is given. Advances in the field of lightwave networks, multi-channel architectures, protocols for multi-channel architectures, combinatorial optimization techniques (mathematical programming, simulated annealing and genetic algorithms), and LAN/MAN problems and methodologies are discussed.*

2.1 Networks and Protocols

In a networked computer system, a user must explicitly logout of a machine, explicitly submit jobs remotely, move files around, and handle all the network management personally. In a distributed system most of the above mentioned activities are automatically done by the system without the user's knowledge. In effect, a distributed system is based on a computer network and whose software gives it a high degree of cohesiveness and transparency.

The main goals of a computer network are to provide resource sharing, high reliability and cost effectiveness. Mainframes are roughly a factor of ten faster than the fastest single chip microprocessors, but they cost a thousand times more. This imbalance has caused many system designers to build systems consisting of powerful personal computers, one per user, with data kept on one or more shared file server machines. This leads to networks with many computers located in the same building. Such a network is called Local Area Network (LAN) to contrast it with the far flung Wide Area Networks (WAN). Some important uses of computer networks are access to remote programs, access to remote databases, and value added communication facilities.

In any network there exists a collection of machines called hosts intended for running user applications. The hosts are connected by a communication subnet. Two types of channels are used in communication subnets. They are *point-to-point* channels and *broadcast* channels. Some broadcast systems also support transmission to a subset of machines, known as *multicasting.*

Modern computer networks are designed in a highly structured way [4], as a hierarchy of layers or levels each one built upon its predecessor. The rules and conventions used in the conversation (between layer n on one machine and layer n on another machine) are collectively known as layer n protocol. Between each pair of adjacent layers there is an interface. The set of layers and protocols is called the network architecture. The network architecture of ISO/OSI (open system interconnection) is commonly used. The OSI model has seven layers namely physical, data link, network, transport, session, presentation, and application.

2.2 Lightwave Networks and Architectures

The advances in lightwave technology have revolutionized multiuser local communication systems. Single-mode fiber is the preferred transmission medium for long distance and point-to-point links. It is preferred due to its low-loss low-attenuation properties, and the huge bandwidth potential (measured in the range of tens of terahertz [1]). This new technology has many unexploited opportunities. To date fiber is used as a substitute for copper within the framework of existing network architectures. Due to such a deployment the performance is improved and the cost has gone down, but the networking system basically remained the same.

The physical level topologies, the layer structure, the protocols within the layers, and the network control functions in use today are all derived from heritage of voice grade telephone lines or local area coaxial cables. Some examples of such traditional network architectures which use fiber as replacement to copper are fiber-distributed data interface (FDDI) [29] and the distributed queuing double bus (DQDB) [25]. The total capacity in these networks is time-shared among the many users of the network. Each user has to have electronics that run at the aggregate bit rate at which the network operates. However, electronic speeds are limited to a few gigabits per second at most. Thus, these architectures cannot be extended to terabits per second capacities because of the basic electronic switching bottleneck. The optical network approach offers the potential of low-cost reliable interconnection of a large number of ports with flexible topologies.

The uniform traffic distribution requires for its fulfilment, a regular topology such as shufflenet [10]. A multihop network with each node having 2 transmitters and receivers is being developed at Colombia University [28]. Algorithms to determine optimal topologies are given in [18, 16]. Further single wavelength can be shared among many stations using wavelength division multiplexing [32, 24]. This can reduce the average number of hops between stations at the cost of having higher speed electronics at each station.

2.2.1 Classification of Network Architectures

Wavelength division multi-plexing concept is illustrated in Figure 2.1. Wavelength division multiplexing network architectures can be classified into broadcast and select networks and wavelength routing networks [15]. In case of broadcast and select networks transmission of each station is broadcast to all other network stations. In wavelength routing networks wavelength routing is used along with optical switching so as to be able to reuse the wavelengths in the network. Each of the above categories, can be classified as either single-hop networks, or as multihop networks.

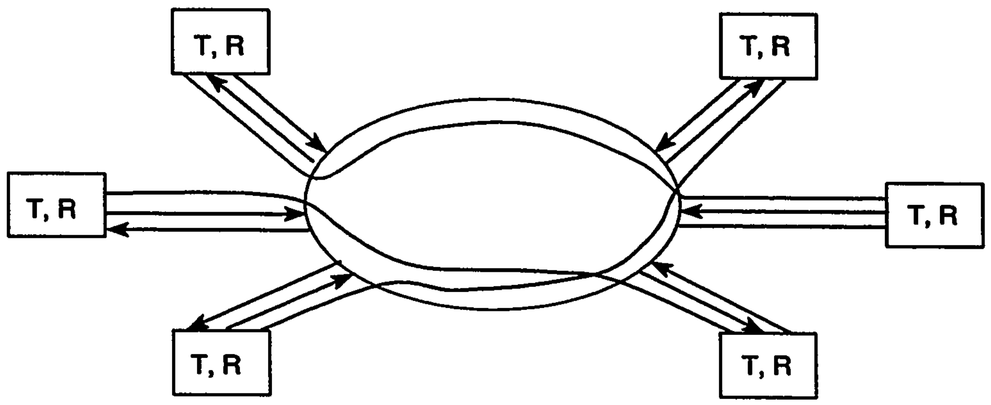


Figure 2.1: Connections Between Different Node Pairs at Different Wavelengths

Broadcast and Select Networks

Broadcast and select networks are further divided into *broadcast and select single-hop* networks and *broadcast and select multi-hop* networks.

Broadcast and Select Single-Hop Networks

In these networks, information once translated as light, reaches its final destination directly without being converted to electric form in between. They are also called all-optical networks. The examples of single-hop broadcast-and-select networks are Bellcore's LAMDANET [23], AT&Ts direct-detection receivers and tunable lasers, and IBMs Rainbow [26]. Media access protocols allow coordination of transmissions between various stations.

Broadcast and Select Multi Hop Networks

In these networks, each station is provided with a small number of tunable optical tranceivers; these transceivers can be tuned to different bands or channels of the optical fiber that are made possible by wavelength division multiplexing. Each transmitter in the network is tuned to a particular wavelength. In general with δ transmitters at each station (and δ receivers) it is possible to select the logical topology so that the average number of hops between stations is $O(\log_\delta N)$ where N is the number of stations.

2.3 Combinatorial Optimization

The formulation and anlysis of several network design problems involves dealing with combinatorial optimization problems. The various combinatorial optimization techniques that are commonly used are the mathematical programming, hill climbing technique, simulated annealing, and genetic algorithms.

2.3.1 Mathematical Programming

Mathematical Programming is 'programming' in the sense of 'planning' [11]. The common feature which mathematical programming techniques have is that they are all used for solving optimization problems, where something needs to be *maximized* or *minimized.* The quantity which is maximized or minimized is known as *objective function.* In this section we confine our attention to selected mathematical programming techniques such as linear, non-linear, and integer programming.

Linear programming [22] is concerned with the optimization of a linear function while satisfying a set of linear equality and/or inequality constraints or restrictions. The simplex method of linear programming enjoys wide acceptance because of (1) its ability to model important and complex decision problems, and (2) its capability for obtaining solutions in a reasonable amount of time. The linearity assumption of linear programming is not always valid in a practical problem, although it makes any model computationally much easier to solve. When we have to incorporate non-linear terms in a model (either in the objective function or in the constraints) we have to use *non-linear programming.* In practice the assumption that the variables can be fractional is perfectly acceptable, if the errors involved in rounding to the nearest integer are not great. If this is not the case then one has to resort to *integer programming* which permits variables to attain only integral values. Integer programming technique is computationally more expensive when compared to linear programming technique.

2.3.2 Simulated Annealing

Simulated annealing is another commonly used optimization technique. It is very time consuming [31] but yields good results. It is a good heuristic for solving any optimization problem. It can be considered as an improved version of the simple random pairwise interchange technique. The basic procedure in simulated annealing is to accept all moves that result in a reduction in cost. Moves that result in a cost increase are accepted with a probability that decreases with the increase in cost. A parameter T, called the temperature, is used to control the acceptance probability of the cost increasing moves. Higher values of T cause more such moves to be accepted. In most implementations of this algorithm, the acceptance probability is given by $\exp(-\Delta C/T)$, where ΔC is the cost increase. In the beginning, the temperature is set to a very high value so that most of the moves are accepted. Then the temperature is gradually decreased so the cost increasing moves have less chance of being accepted. Ultimately, the temperature is reduced to a very low value so that only moves causing a cost reduction are accepted, and the algorithm converges to a low cost configuration.

2.3.3 Genetic Algorithms

Genetic algorithms (GAs) are search techniques which emulate the natural process of evolution as a means of progressing toward the optimum. The algorithm starts with an initial set of random configurations called a population. Each individual in the population is a string of symbols called genes. The string made up of genes is termed chromosome which represents a solution to the optimization problem. A set of genes that makes up a partial solution is called a schema. During each iteration (generation), the individuals in the current population are evaluated using some measure of fitness. Based on the fitness value, two individuals at a time (called parents) are *selected* from the population. The fitter individuals have a higher probability of being selected. Then, a number of genetic operators such as *crossover* and *mutation* are applied on the selected parents to generate new individual solutions called offsprings. These genetic operators combine the features of both parents.

2.4 LAN/MAN Design Problems and Methodologies

The need for high bandwidth communications provides an incentive for the deployment of Fiber Optic Local Area Networks (FOLAN). FOLAN provides communications for customers requiring high bit rate services at low cost. The cost may be lowered by sharing the large bandwidth that optical communication channels provide.

Fiber optics is becoming increasingly important for local area networks due to high band width, light-weight, low-loss, ineffectiveness from electro magnetic interference of the fiber. Optical fiber also offers excellent security as it is difficult to wiretap without detection [5]. The key issues in the design of optical local area networks are the connection topology to be used, communication protocols, and the hardware requirements. The advances in optical technology specially fiber-optics and opto-electronics have created the need to reexamine the way communications are provided. We need to find ways through which the broadband of the fiber optics may be fully exploited.

In recent years, the number of stations attached to a local area network and its usage by each station has increased rapidly. The increased usage is due to rise in distributed services, shared storage with diskless workstations, information servers, distributed image servers, distributed image intensive applications, graphic terminals, etc., and is expected to grow even further with the incresing scope of multimedia services. Most local area network based distributed systems have a serial component of communication that sooner or later becomes the bottleneck. For a truly Multiple Instruction Multiple Data (MIMD) computation multi-channel architectures are needed [34]. Multi-channel local area networks (M-LANs) try to achieve very high data rate [5]. M-LANs use lower data rate buses connected to the stations. Buses in M-LANs need not be physically separate; they can be obtained by dividing the bandwidth of a single physical connection. M-LANs offer the reliability characteristics while providing a large bandwidth.

Some work was carried out on fiber LAN/MAN architectures in the past. The work does not satisfactorily exploit the advantages offered by fiber. Part of the challenge to researchers in this area is to find ways of utilizing the enormous bandwidth capacity of the fiber by designing LAN/MAN architectures that can operate successfully within the constraints that fiber imposes.

In conclusion, fiber-optic networks with terabit/s capacities are not just an exiting area for research, but appear to be perfectly practical to build [34]. Many interesting problems in lightwave networks remain to be solved. These include the development of sequential and distributed algorithms for adapting the network topology to changing traffic patterns in a multihop network, distributed routing, management and control of wavelength routing networks, interconnection of LANs and MANs using WANs, effective optimization techniques for optimizing physical and virtual topology of lightwave networks, and frameworks for the design of lightwave networks.

2.5 Summary

In this chapter we introduced networks and protocols. Then we discussed wavelength division multiplexing principle along with classification of network architectures. Then we talked about combinatorial optimization techniques like mathematical programming, simulated annealing and genetic algorithms. Finally, we presented problems and methodologies for LANs/MANs.

Chapter 3

Multi-Channel Architectures and Protocols

Chapter Synopsis: *A brief introduction to multi-channel network architectures is given. Different goals and advantages of multi-channel architectures are provided.*

3.1 Introduction

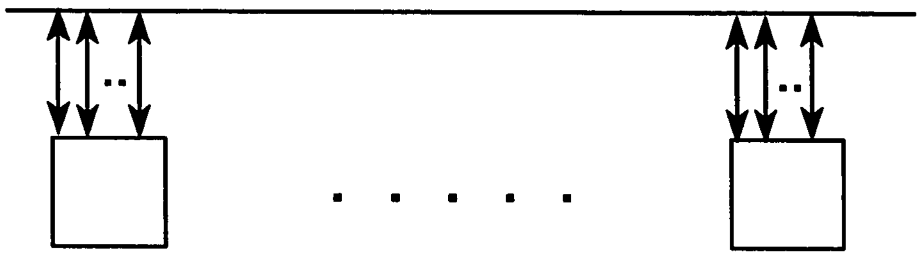
Current technology allows as many as 1000 high bandwidth (100-200 Mb/s) independent channels, supported on a single optical fiber. In the near future, the bandwidth of each channel is likely to reach 1 Gb/s [3]. The multi-channel technology will allow growth in capacity and increase in processing power. Advances in fiber-optic technology have made different approaches to parallel processor interconnection feasible.

3.2 Technologies: Tunable Transmitters and Receivers

Developments in *tunable lasers* and *tunable receivers* are making future capabilities of optical fiber hard to predict [8]. A technology that is comercially available today is the tunable Fabry-Perot filter. Such devices can be made tunable over almost any desired range, but the tuning speed is limited by mechanical inertia to the order of hundreds of microseconds, fast enough for all circuit applications, but not fast enough for packet switching. The future tunable transmitter and receiver devices are very promising specially for packet switching.

3.3 Multi-Channel Architectures

Multi-channel architectures provide independently selectable channels using single optical fiber [34]. Figure 3.1(a) shows how different stations are connected by a single bus. This is how we see the connections physically. Figure 3.1(b) shows the logical connections between different stations, this virtual diagram is obtained by tuning different transmitters and receivers to different bands (frequencies), which are made possible by wavelength division multiplexing.



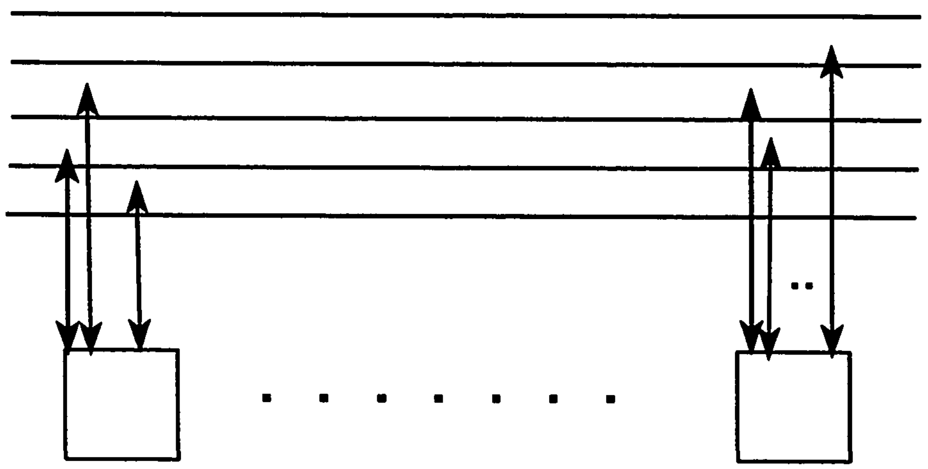


Figure 3.1: Realization of Multi-Channel Architecture

3.4 Goals of Multi-Channel Network Architectures

The multi-channel architecture is targeted to be implemented in an optical wavelength division multiplexing network, addressing the mismatch between optical transmission rates and electronic processing speeds to alleviate the bottlenecks created at intermediate nodes. The solution to the problem of congestion and long waiting time at the intermediate stations is to have a channel connection established between two intermediate stations. Wavelengths are assigned to channels.

Multi-channel architectures allow us to create a general purpose computing platform that can support large number of users with vastly different computing requirements. It will also allow us to create a parallel processing test-bed to study all aspects of parallel processing. With the help of tunable transceivers, multi-channel architectures allow the partition and allocation of bandwidth by task. Multiple tasks with small communication requirements can be combined to better utilize bandwidth.

The proposed multi-channel architecture enables us to construct an integrated packet and circuit switching alternative. Packets will be routed through logically adjacent instead of routing between all physically adjacent links between the source and the destination.

The use of data-channels will enable the use of switches with large setup time. Switching need not be on a per packet basis. from the wavelength continuous property of the data-channel. This is due to the fact that a channel incoming on one wavelength need not be switched to another wavelength. The multi-channel architecture reduces the number of active nodes through which a packet is switched between source and destination thus alleviating the processing and buffering bottlenecks. The multi-channel architecture represents a possible solution to congestion problem at intermediate nodes, and provides a viable solution for networks having non-uniform traffic patterns. This is because of the possibility to reassign data-channels.

3.5 Advantages of Multi-Channel Architectures

Multi-Channel Architecture (MCA) allows different virtual buses with a single fiber which alleviates communication bottlenecks by providing parallel communication. MCA make real time processing possible. By dedicating channels for real time operation, delay in response can be controlled. MCA allows development of new parallel operating systems for efficient allocation of resources (including bandwidth) to tasks. MCA subsumes all the fixed interconnection networks which are its special cases, these fixed interconnection networks can be obtained by tuning transmitters and receivers at a node to different channels. MCA allows greater flexibility in programming parallel algorithms. Different phases of a parallel solution to a problem may require different topologies which can be realized by tuning to different channels. MCAs are making LANs/MANs into truly distributed systems with parallel computation and communication which is a radical departure from conventional network architectures

3.6 Protocols for Multi-Channel Architectures

Communication protocols are an important aspect of LANs and MANs. Design of good protocols that could effectively use the large bandwidth communication media is difficult. At low rates, that is 10 Mb/s, a carrier sense multiple access/ collision detection (CSMA/CD) protocol is used. It appears that the fiber-based architectures would be restricted to unidirectional links. However the study of LAN and MAN architectures is an important aspect of an emerging field. More recently work along these lines has been generalized to cover architectures based on multiple buses [15].

3.7 Summary

In this chapter, we gave brief introduction to multi-channel network architectures then we talked about different goals and advantages of multi-channel architectures. Finally, we talked about protocols for multi-channel architectures.

Chapter 4

A Design Methodology for Multi-Channel Network Architecture

Chapter Synopsis: *A methodology for the design of multi-channel network architecture is presented.*

4.1 Introduction

A methodology for the design of multi-channel network architecture based on multi-channel architecture model and recent concepts and techniques can be defined. The methodology supports regularity in design and simplicity in the design process.

4.2 Methodology

The methodology is presented in iterative steps which involves:

1. Multi-channel network architecture requirement analysis
2. Selection of tentative physical interconnection topology (such as tree,..)
3. Detailed simulation of the physical, virtual and optimal routing problem
4. Verification and validation of the solution for the satisfaction of requirements. After going through the verification step, it will be clear, whether performance cost and physical constraints are good enough. If not another iteration of the design process is done making better choices of design variables.

The various steps of the methodology are as follows:

Step 1: Multi-channel network requirement analysis.
Analysis is done with applications of multi-channel network in view.

- Number of multi-channel network stations n
- Number of channels m
- Number of transceivers per node T
- Serviceability requirement
- Fault tolerance and reliability requirement
- Extendibility requirement

Step 2: Based on requirements in Step 1, the design problems formulated in Chapter 5 are solved for the given problem.

• In the physical topology problem it needs to be decided which physical distributed topology among the star topology, tree topology, 2-clustered tree topology and 4-clustered tree topology to be used. The coupler locations are obtained by solving a genetic algorithm as discussed in Chapter 6.

• The virtual topology problem is also solved using genetic algorithm as explained in Chapter 5. As a result a tuning matrix (which gives the connectivity between stations) is obtained.

• Flows have to be optimally assigned over the links of the logical diagram (obtained from connectivity tuning matrix) yielding the routing problem. The genetic algorithm can be applied to the routing problem to minimize the maximum flow.

4.3 Summary

In this chapter we gave a design methodology for multi-channel network architecture. The methodology is given in iterative steps involving requirement analysis of the network, design of physical and virtual topology and finally validation of the results.

Chapter 5

Design Problems and their Formulations in Multi-Channel Network Architectures

Chapter Synopsis: *The need for virtual topology design in lightwave networks is discussed. Various problems that need to be solved while designing lightwave networks are stated and formulated. Solutions to these problems are outlined.*

5.1 Introduction

The multi-channel network architecture is a wavelength division multiplexing architecture that was proposed for local and metropolitan based communication networks

[1, 2]. Each station in a multi-channel network consists of a small number (say two to four) of transmitters and receivers. Each transmitter in the network is tuned to a different wavelength. Although the network is physically a star (or a bus, or any other broadcast topology), stations can transmit data directly only to those stations that have a receiver tuned to one of their transmit wavelengths. At each intermediate station, optical data is converted to electronic form, the packet headers are decoded, and the packet is then switched electronically and transmitted on the appropriate wavelength enroute to its final destination.

The originally proposed logical interconnection pattern was meant for uniform traffic condition [1, 10], and consisted of several stages connected through a perfect shuffle [12] or simplified routing at intermediate nodes in case of deBruijn graph [19]. In case of perfect shuffle connection, the last stage is connected back to the first stage as shown in the Figure 5.1. Each link in this diagram consists of distinct wavelength (channel), with all channels multiplexed onto the optical medium. In general with δ transmitters at each station (and δ receivers) the average number of hops a packet would take is θ ($\log_\delta N$) where N is the number of stations.

An interesting feature of the multihop network is that the logical topology can be changed simply by retuning some of the receivers (or transmitters) in the network. This means that it is possible to reconfigure the logical topology according to traffic patterns. In addition, a single wavelength or channel can be shared among many stations. This can reduce the average number of hops between stations, at the cost of having higher speed electronics at each station.

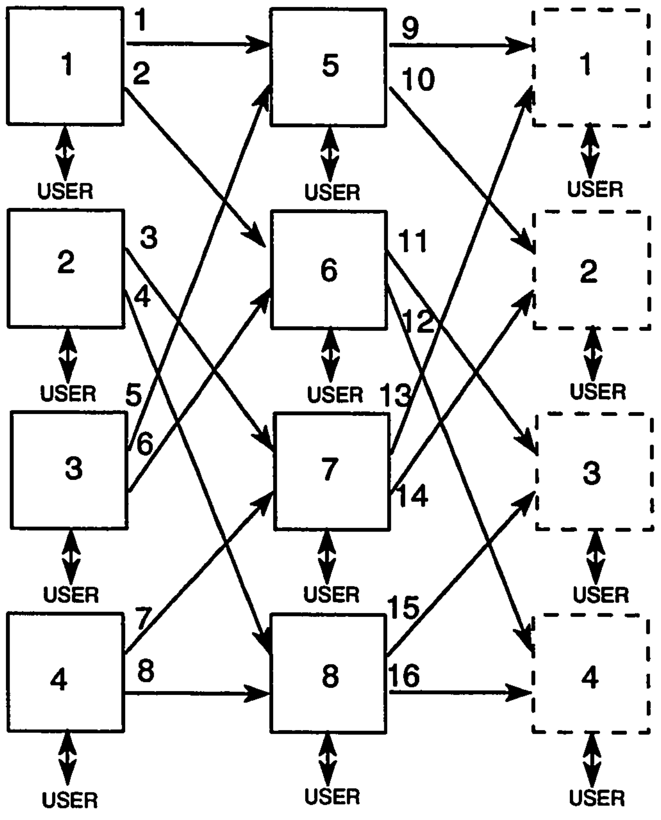


Figure 5.1: Virtual Perfect Shuffle Interconnection

The large scale design of multi-channel architectures addresses a number of problem areas: the selection of topology (physical topology), the selection of an interconnection graph (virtual topology) and the problem of optimal routing. These problems will be formally proposed and formulated in this Chapter. Focus will be on virtual topology design problem, which is the problem of determining the interconnection graph or effective tuning matrix that maximizes the channel traffic and the total traffic between different stations. We will first discuss the physical topology design problem, then the design of virtual topology and finally the problem of optimal routing.

5.2 Formulation of Physical Topology Design Problem (PTDP)

The linear (geographical) arrangement of the network stations as shown in Figure 5.2 gives the physical topology of the multi-channel architecture. In other words physical topology of multi-channel network architecture is the layout of the optical fiber, associated optical components (couplers, amplifiers), and the locations of the station themselves.

Station attached to a folded bus shown Figure 5.2 is not a desirable one, because the transmitter to receiver power loss attributable to the couplers, taps and connectors increases linearly with number of stations [17]. Other examples of physical

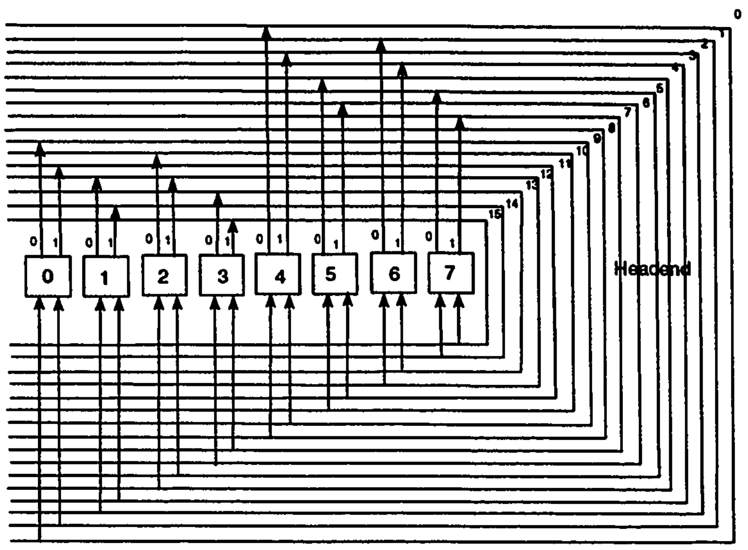


Figure 5.2: Physical Bus Topology Using Wavelengths 0-15

distributed topologies such as the tree topology and the star topology are more desirable since they accommodate many more stations than the linear bus for a given power margin [13].

5.2.1 Physical Design Topology Problem

The particular physical topology chosen from the different alternatives (eg. bus, star, tree, and ring) is largely a function of power margin [13], but this aspect of network design does not concern us here. The physical topology design problem is to determine the optimal deployment of optical fiber and optical components. In other words physical topology design problem is to decide which physical topology

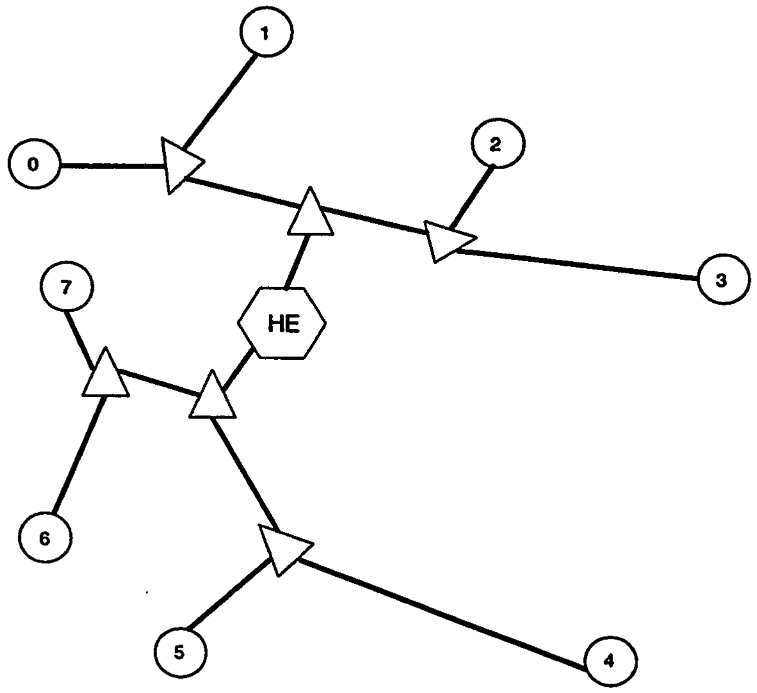


Figure 5.3: Physical Topology

should be selected to minimize the cost of multi-channel network architecture (as reflected by the total length of optical fiber).

In the physical topology design problem, if we employ the tree topology or star topology the problem becomes one of finding locations for the couplers on the internal nodes of the tree (the triangles), with the objective of minimizing the length of optical fiber used in the multi-channel network for LAN/MAN design we assume that the optical fiber from one point to another is modelled by the Euclidian distance metric.

5.2.2 Problem Formulation

We can formulate the physical topology design problem as a constrained optimization problem in which, given the locations of a set of points (stations) in the plane, choose another set of points (couplers) so that the spanning tree for these sets of points has minimum length. This problem is the geometric steiner problem, which is intractable [21]. Thus, we restrict slightly the statement of physical topology design problem as follows: To find the optimal positioning of couplers, given the basic shape of the tree. This basic shape is the class of topologies such as star, binary tree, binary tree with clusters of stations arranged on buses at the leaves. Once we choose topological class and heuristically group stations into clusters, the physical topology design problem becomes one of locating the nonstationary equipment such as the couplers and headend, so that the cost of the connecting optical fiber is minimized. Informally, the problem can be stated as follows:

Given:	1)Station-Coupler connectivity
	2)Station locations
Objective:	Minimize total length of optical fiber.
Variables:	Coupler locations
Constraints:	1) No segments has more than a given number of taps.
	2) Length of the segment $\leq$ maximum length

5.2.3 Problem Solution

The physical topology design problem can be solved using the genetic algorithm discussed in the next Chapter. In the physical topology problem the coupler locations will form the chromosome and the objective function minimizes the total length of optical fiber. The constraint of maximum allowable length depending upon the type of network used (LAN or MAN) will be incorporated in the objective function as a penalty function. The only inputs to the physical topology design problem are the locations of the stations to be connected, indication of what class of topology is being considered and the maximum allowable length of the fiber. The principle outputs are the locations of the couplers, from which the multi-channel network distance matrix can be calculated.

5.3 Formulation of Virtual Topology Design Problem (VTDP)

The interconnection graph, an abstraction representing the way the station's transmitters and receivers are tuned to WDM channels, is called the multi-channel network virtual topology.

A key property of multi-channel network is the relative independence between the virtual topology (logical interconnection among nodes) and the physical topology (fiber layout). As a result of this property any virtual topology (subject to degree requirements) can be mapped onto a physical topology. The optical transmitters and receivers (which can be tuned to any of the wavelengths) help in adaptively optimizing the virtual topology of a multi-channel network. This capability of actually changing the virtual topology (reconfigurability) in response to prevailing conditions in the network is an important property of multi-channel network architecture.

5.3.1 The Virtual Topology Design Problem (VTDP)

The virtual topology design problem involves different choices of design variables and constraints. The multi-channel network has a physical and a virtual topology. The virtual topology can be easily obtained by means of a tunable matrix which indicates, how the receivers and transmitters are tuned to logical channels (or bands). The network designer can choose the virtual topology of a multi-channel network architecture to optimize a given characteristic of the network.

The performance measures of interest are the delays over the network and the throughput achieved by the network. We consider the potential increase in the throughput for a given traffic pattern. The potential increase in the throughput is maximized by minimizing the maximum traffic flow or vice versa over the channels in the network. This measure i.e., the potential increase in the throughput can also be maximized by maximizing the total traffic flowing from source to destination in single hop. It may be the case that no path from node i to j may exist (i.e the entry of connectivity matrix is 0), although element of the traffic matrix (λ matrix) is nonzero. The maximization of total traffic will avoid this situation effectively.

The $MIN - MAX$ performance doesn't take propagation into account and would not be suited for wide area networks where propagation delay plays a prominent role. The $MIN - MAX$ performance measure would however capture some quantitative aspects of transmission delay. The $MIN - MAX$ could take the possibility of channel sharing into account with the need to specify the media-access-protocols used.

5.3.2 Problem Formulation

The traffic flowing from a source to destination and the traffic flowing in a channel is basically our performance metric. The decision variable in the formulation of virtual topology design problem is the tuning matrix X_{ijk}, an entry of which specifies whether station i transmits to station j over channel k. The tuning matrix whose third dimension represents the channel used is shown in Figure 5.6. The entries of the tuning matrix assumes value from the set $0, 1$. The problem may thus be expressed as maximization of the total traffic flowing between each pair and the channel traffic, subject to the degree constraints and channel flow constraints. In summary we can state the virtual topology design problem as follows:

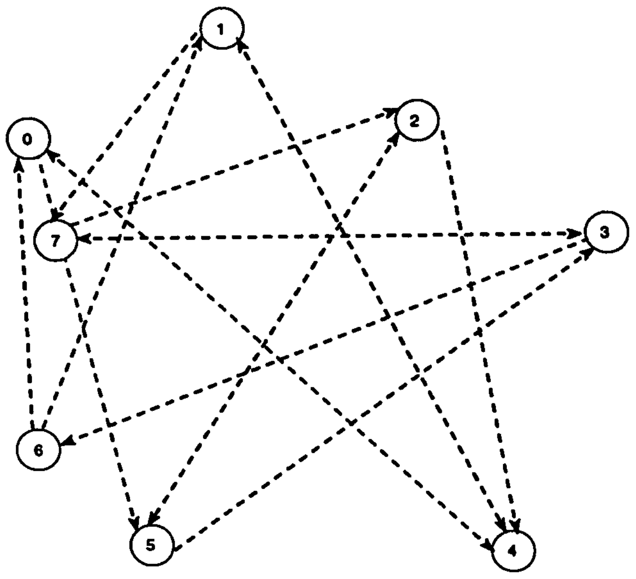


Figure 5.4: Perfect Shuffle Virtual Topology

Given:	Load matrix λ_{ij}, Number of nodes N,
	Number of nodes M, Number of tranceivers per node T
Objective:	Maximize (the sum of total traffic flow and minimum channel flow)
Variables:	Tuning matrix X_{ijk}
Constraints:	1) Station connectivity
	2) Channel flow constraints
	3) $X_{ijk} \in 0, 1$

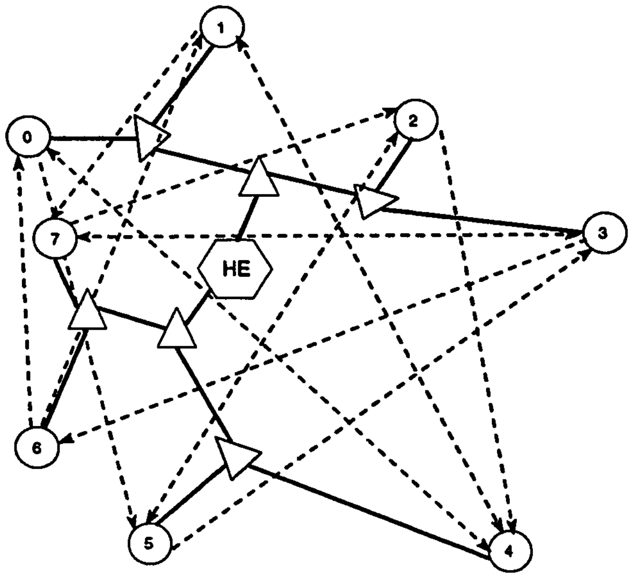


Figure 5.5: Virtual Topology Mapped Over Physical Topology

In detail the problem can be expressed as:

$$Max\ \{C_{w1} \sum_{k=1}^{m}\sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_{ij}.X_{ijk} + C_{w2}\ \min_{k=1}^{m} \sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_{ij}.X_{ijk}\ \} \tag{5.1}$$

$$Such\ \ that\ \sum_{k=1}^{m} or_{j=1}^{n} X_{ijk} \leq T\ \forall i \in \{1..n\} \tag{5.2}$$

$$\sum_{k=1}^{m} or_{i=1}^{n} X_{ijk} \leq T\ \forall j \in \{1..n\} \tag{5.3}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} X_{ijk} \geq 1\ \forall k \in \{1..m\} \tag{5.4}$$

$$X_{ijk} \in \{0,1\} \tag{5.5}$$

$$X_{ijk} = \begin{cases} 1, & if\ s_i\ is\ transmitting\ to\ s_j\ over\ channel\ k \\ 0 & otherwise. \end{cases} \tag{5.6}$$

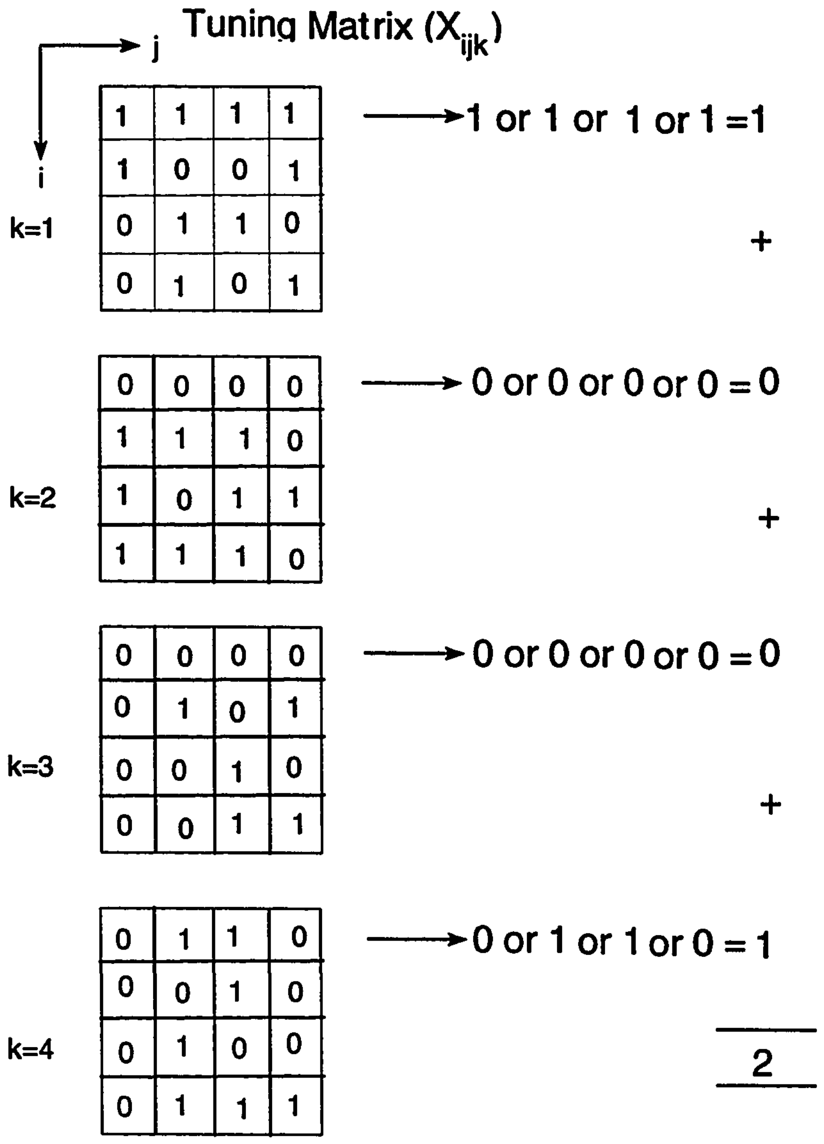


Figure 5.6: A $4 \times 4 \times 4$ Tuning Matrix

Equation (5.1) is the objective function which is the maximization of traffic flowing from source to destination in one-hop and maximization of minimum channel traffic. Equation (5.2) and (5.3) are the station's degree constraints which require that every station is assigned to less than or equal to T reception and transmission channels. Equation (5.4) is the usage constraints which requires atleast one transmitting and one receiving station per channel i.e., each channel gets used by atleast a pair of nodes. The last Equation (5.5) says that decision variable X_{ijk} assumes values from the set $\{0, 1\}$.

5.3.3 Problem Solution

The virtual topology design problem is a combinatorial optimization problem for which there are no known efficient solution procedures due the nonlinear constraints and non-linear terms used in the objective function. We use genetic algorithms to solve the virtual topology design problem. Due to fast convergence and no need for initial solutions genetic algorithms are preferable.A genetic algorithm for solving a problem begins with a population of chromosomes that encode solutions to the problem. The algorithm requires an evaluation function that will take a chromosome and returns a numerical measure of its fitness in the environment of the problem. The fitness function plays the role of the environment in natural evolution. When population of the chromosomes has been evaluated, the chromosomes become the parents of a new population. A parent's chromosomes will reproduce more frequently if it is fitter than the other chromosomes in its generation.

In the process of reproduction, genetic operators may alter the composition of a parent. Such operators include mutation operator that randomly alters gene values and crossover operator that combines the chromosomal matter of two parents. A genetic algorithm proceeds in this way, creating a population, evaluating it and reproducing its members to form a new population, until a stopping criterion is met. In general, the best individual evolved by this process is taken to be the solution to the problem.

5.4 Formulation of Optimal Routing

For a given allocation of transmitters and receivers to the nodes of the network, we get the connectivity or tuning matrix by solving the virtual topology design problem mentioned in the previous section. Once a connectivity diagram (from tuning matrix) is obtained, flows have to be optimally assigned over the links of the logical diagram, yielding the routing problem for the multi-channel network. Routing is not done on an aggregate basis. It is done on a packet-by-packet basis.

Given:	Load matrix lambda, number of nodes n, number of channels m
Objective:	Minimize the maximum flow over the links in the network
Variable:	Flow matrix f_{ij}
Constraints:	Flow constraints, other constraints

For the routing problem we introduce the following flow variables:

${f_{ij}}^{st}$= traffic due to source destination pair(s,t) flowing from node i and node j. The total flow on link (i, j), f_{ij} is obtained by summing over all source-destination pairs (s, t) the flows ${f_{ij}}^{st}$. We choose to consider the potential increase in the throughput for a given traffic pattern, and that measure is maximized by minimizing the largest flow over the links in the network i.e., minimize max$\{f_{ij}\}$

$$Minimize\ max_{(i,j)}\{f_{ij}\} \tag{5.7}$$

$$\sum_{j=1}^{n} {f_{ij}}^{st} - \sum_{j=1}^{n} {f_{ji}}^{st} = 0 \quad \forall i, j, s, t \tag{5.8}$$

$$f_{ij} = \sum_{s,t} {f_{ij}}^{st} \quad \forall i, j, s, t \tag{5.9}$$

$${f_{ij}}^{st}, f_{ij} \geq 0 \quad \forall i, j, s, t \tag{5.10}$$

Equation 5.9 specifies that for each source-destination pair (s, t) the traffic flowing into a node balances the traffic flowing out of a node. The routing problem can be formulated as a linear programming problem.

5.5 Summary

In this chapter, we proposed and formulated three different problems namely physical topology design, virtual topology design, and optimal routing. The design of virtual topology is the selection of optimal channel assignment, maximizing the throughput of the network. Similarly, the optimal routing problem is the minimization of maximum flow, subject to flow constraints.

Chapter 6

A Genetic Algorithm for Virtual Topology Optimization Problem

Chapter Synopsis: *The details of a genetic algorithm used for solving virtual topology design problem are discussed. Genetic algorithm and the operators used are introduced. Design of objective function, selection operator, crossover operator, and mutation operator are presented. The working of a genetic algorithm is illustrated with an example.*

6.1 Introduction

Genetic algorithms (GAs) are derived by analogy from biological process of evolution. The problem of virtual topology design, has not been addressed by a genetic algorithm before. The evaluation function is highly complex. The number of possible solutions are vast. The virtual topology design problem is one of finding an optimal solution in a large search space with many local optima, and genetic algorithms are designed to do just that. The genetic algorithm we are about to design has the following essential parts: technique for creating the initial population, the evaluation function (or the fitness function), the operators that would alter the genetic composition of parents (such as crossover and mutation operators) when they reproduce, and the parameter string that would control the behavior of the algorithm and its genetic operators.

```
Algorithm GeneticVTDA:
Input values Ng,Np,Mp,Cp
(* Ng:number of generations, Np:number of individuals, Mp:Mutation probability,
Cp:Crossover probability *)
BEGIN
   Initialize;
(* Procedure initialize will initialize a population at random. The procedure will
also evaluate fitness of each individual based on the objective function value *)
   FOR i← 1 TO Ng
   Generation;
(* Procedure generation creates a new generation through Select, crossover and
mutation, in the process it will use the procedures select, crossover, mutate and
objective function*)
```

```
    Statistics;
(* Procedure statistics calculates population statistics such as minimum, maximum
and average fitness and sum of fitness of the new population and the individual with
maximum fitness*)
    Result;
(* Procedure results give the new population*)
    ENDFOR
    Return the highest fitness individual obtained in statistics
END GeneticVTDA
```

6.2 Working of Genetic Algorithms

Genetic algorithms start with a set of problem solutions called initial population. Then they transform these solutions in a way that resembles the mechanics of natural evolution which includes reproduction and survival. Figure 6.1 shows the working of genetic algorithms i.e., formation of different generations. Figure 6.2 shows how a new generation is formed.

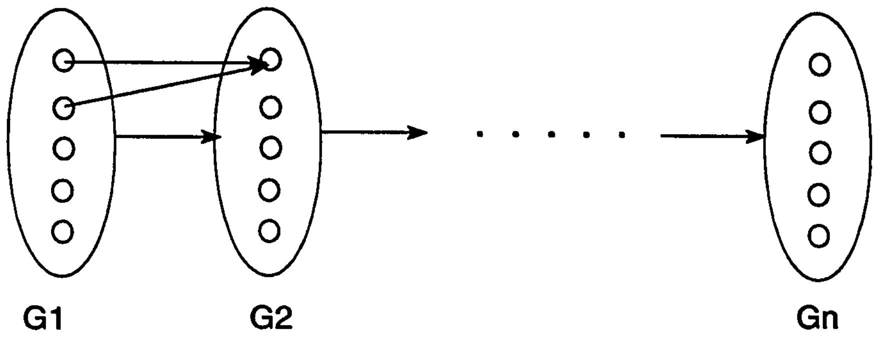


Figure 6.1: Working of Genetic Algorithms

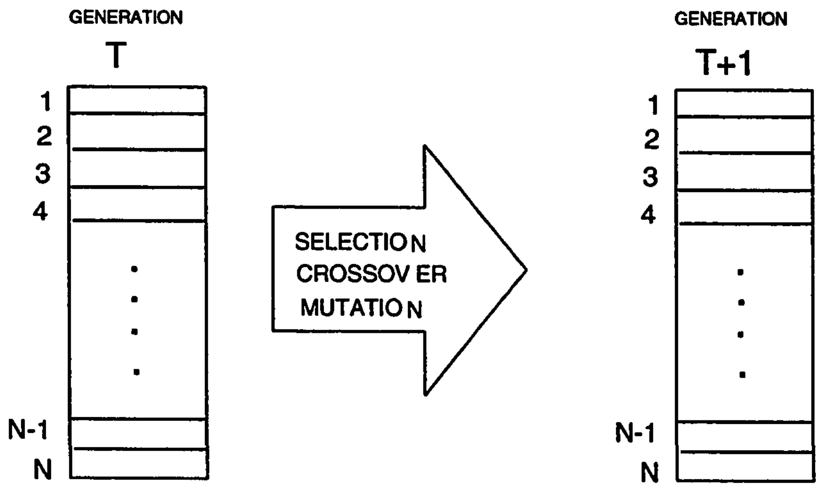


Figure 6.2: Formation of New Generation

6.3 Genetic Algorithm (GA) for Virtual Topology Design Problem

In this section we discuss the design of a genetic algorithm for virtual topology design problem. We give the basic structure of the algorithm, discuss the design of fitness function, and then we talk about chromosomal representation and initial population generation. Finally, we give algorithms for different genetic operators such as selection, crossover, and mutation operators. The basic structure of genetic algorithm for virtual topology design problem is:

STEP1: Begin with a population of individuals (tuning matrix X_{ijk}) whose entries are generated at random.

STEP2: Determine the fitness of each individual in the current population (i.e., the value of objective function $f(X_{ijk})$ for each individual)

STEP3: Select the parents for the next generation with a probability proportional to fitness.

STEP4: Mate the selected parents to produce offspring i.e., by means of crossover operation selecting the sight of crossover randomly.

STEP5: Perform mutation using mutation operators.

STEP6: Repeat STEP2 to STEP5 until a satisfactory solution is found.

6.3.1 Fitness Function

A clear and correct formulation of fitness function is necessary for a good genetic algorithm. Since our problem is complex consisting of multiple constraints, we have to find a means to represent these constraints as parts of the fitness function (or objective function). Equality constraints may be summed into a system model, we are really concerned about inequality constraints. One way out, is to evaluate the objective function, and check to see if constraints are violated. If not, the parameter set is assigned the fitness value corresponding to the objective function evaluation. If constraints are violated, the solution is infeasible and thus has no fitness. This procedure is fine except that many practical problems (such as our problem) are highly constrained; finding a feasible solution is difficult. As a result, we usually want to get some information out of infeasible solutions, perhaps by degrading their fitness ranking in relation to the degree of constraint violation. This is what is suggested in the next method called penalty method.

In penalty method, a constrained problem is transformed to an unconstrained problem by associating a cost or penalty with all constraint violations. This cost is included in the objective function evaluation. Our highly constrained problem formulated in Equations (5.1) to (5.6) can be easily transformed to unconstrained form.

The objective function of the virtual topology design problem, after application of penalty method would look like

$$Maximize\ f(X_{ijk}) \tag{6.1}$$

$$\alpha(X_{ijk}) = \textstyle\sum_{k=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_{ij}.X_{ijk} \tag{6.2}$$

$$\beta(X_{ijk}) = \textstyle\sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_{ij}.X_{ijk} \tag{6.3}$$

$$\delta(X_{ijk}) = (T - \textstyle\sum_{k=1}^{m}\ or_{i=1}^{n}\ X_{ijk})^2 \tag{6.4}$$

$$\phi(X_{ijk}) = (T - \textstyle\sum_{k=1}^{m}\ or_{j=1}^{n}\ X_{ijk})^2 \tag{6.5}$$

$$f(X_{ijk}) = A.\,\alpha(X_{ijk}) + B.\ \min_{k=1}^{m}\ \beta(X_{ijk}) - C.\textstyle\sum_{i=1}^{n} \delta(X_{ijk}) - D.\textstyle\sum_{j=1}^{n} \phi(X_{ijk}) \tag{6.6}$$

The coefficients A, B, C, D used in Equation (6.6) are called the penalty coefficients and the functions in Equations (6.4) and (6.5) are the penalty functions. We usually square the penalty functions or the violations of the constraint. In the genetic algorithm's evolution, the unconstrained solutions are dropped in favour of their variants which are constrained solutions, as the penalty coefficient becomes larger. We give different penalty coefficients different values so that moderate violations of the constraints yield a penalty that is some significant percentage of a nominal operating cost.

We now explain the significance of functions in the Equations (6.1) to (6.6) in detail. The function in Equation (6.6) is the main objective function expressed as sum of different functions. The functions in Equations (6.2) and (6.3) are the actual min-max functions that are being optimized. The function in Equation (6.2) tries to maximize the traffic flowing from any station i to any station j ($1 \leq i \leq n$ and $1 \leq j \leq n$) in a single hop. There can be a situation where a path from station i to station j may not exist or the tuning matrix entry X_{ijk} is zero $\forall k$, although the corresponding elements of the traffic matrix λ_{ij} is non zero. Equation (6.3) will balance the channel traffic flow by maximizing channel traffic. The penalty functions in Equations (6.4) and (6.5) will penalize the main function for violation of constraints. The functions in Equations (6.4) and (6.5) are the penalty functions which would penalize heavily if the degree constraint is violated. A degree constraint is one that requires the number of connections (or Ored entries of X_{ijk} matrix) to be greater than or equal to the number of transceivers T.

6.3.2 Chromosomal Representation

Our chromosomal representation is straightforward. Chromosome i.e., the tuning matrix in our case is a $0 - 1$ matrix. Extensive work has been done by Holland and his coworkers on bit string representation [6]. They have shown that bit string mechanism is an effective mechanism in domains about which little is known especially in function optimization. A lot is known about the genetic operators that work well with $0 - 1$ representation of individuals. The use of 0's and 1's does not require decoding as the actual values of entries of tuning matrices are 0's and 1's.

6.3.3 Initial Population Generation

There are various methods of generating initial populations. The population can be generated randomly. A good deal can be learned by initializing a population randomly. Moving from a randomly created population to a well adapted population is a good test of the genetic algorithm, since the critical features of the final solution will be produced by the search and recombination mechanisms, rather than the initialization procedures. In our case we use randomly generated initial population to make use of the advantages offered by this technique. However, the initial populations contributed by the greedy method or clustering technique is fitter initially, it rapidly converges to a local optimum.

6.3.4 Genetic Operators

Genetic operators for bit string representations have been extensively studied. As the chromosome in our case is three dimensional matrix we have to tailor the genetic operators according to the domain of our application. We choose the set of genetic operators such as selection, crossover and mutation operators, which attempts to move towards promising parts of search space. Crossover is the main genetic operator. It operates on two individuals and generates an offspring. It is an inheritance mechanism where the offspring inherits the characteristics of both its parents. Mutation produces incremental random changes in the offspring generated by the crossover.

Selection Operator

The selection operator or the reproduction operator selects the individuals (tuning matrices in our case) according to its objective function value (or fitness). The selection will be done randomly, which helps in selecting those individuals whose objective function value is high when compared to others. This is the simulation of " survival of the fittest" concept. Once an individual (a tuning matrix) is selected for reproduction, an exact replica is entered into a mating pool, ready to be acted upon by the cross over operator.

Crossover Operator

The sharing of the chromosomal material (genes) during mating is called crossover. There are many different crossover methods, the two important among them are *single-point crossover* and *two-point crossover.* The single point operator chooses a random cut point and generates the offsprings by combining the segment of one parent to the left of the cut point with the segment of the other parent to the right of the cut. Two-point crossover takes two parents, selects two cutpoints and combines the chromosomal material of the parents by swapping the material between the cut points of the second parent for the material between the cut points of the first. We confine ourse!f to the single point crossover method. The Figures 6.4 and 6.5 illustrate the single point crossover operation.

After the selection of parents for mating, we perform a biased coin flip with a certain probability of heads that will determine whether to proceed with the crossover. If the

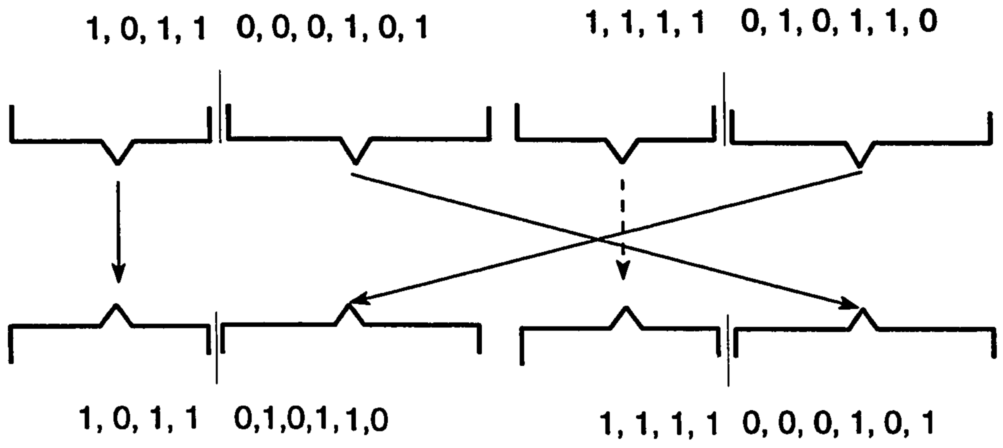


Figure 6.3: Crossover Operation on Bit Strings

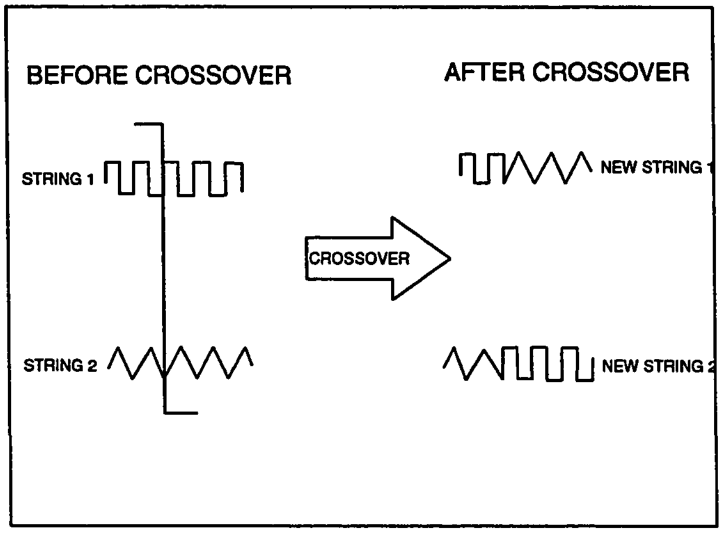


Figure 6.4: Single Point Crossover Operation

coin toss is successful, we choose, at random a particular locus called the crossover point. Two children are produced by splicing the genes upto the crossover point from one parent with the genes beyond the crossover point from the other parent. If the coin toss is not successful, the parents are simply returned as new children. The crossover point chosen actually represents the row number beyond which swapping of rows occurs between the two parents, to produce an offspring.

Mutation Operator

The random modification of a gene is called *mutation.* The mutation rate controls the rate at which new genes are introduced into the population for trial. If it is too low, then population evolves steadily towards a local optimum condition. If it is high there will be too many random disturbances, and the offsprings will start losing their resemblance to their parents and the algorithm will lose its ability to learn from the history of the search.

The theory of genetic algorithm shows how the crossover mechanism can result in fitter population. It can however, happen that crossover results in children who are less fit than their parents. By subjecting each of the genes in a chromosome to a very small probability of mutation, the result of a bad crossover is reversed. Mutation can flip gene back to favorable value. If all the genes are subjected to mutation, we could end up with a situation wherein a favorable gene is altered. Thus, we need to keep the value of probability of mutation very small. Thus, mutation avoids quick convergence, which allows a thorough exploration of state space.

6.3.5 Generation of New Population

A simplistic approach to building a new population is to mate enough parents so that enough children are produced to completely replace their parents. This technique called the *generational replacement*, allows for a thorough mixing of genes in the new generation, but it has some drawbacks. There is no guarantee that all or even most of the children will turn out better than their parents. Thus generational replacement might result in a loss of individuals with the best genes. Not only could the best individuals be lost, but the population as a whole could diminish in overall fitness.

We could counter some of the negative effects of generational replacement by retaining a certain number of the best individuals from the previous generation. This strategy is called *elitism.* At the opposite end of the spectrum from generational replacement is *steady state reproduction.* In this method, a certain number of individuals are replaced in each successive generation. In the design of our algorithm we employ the elitism strategy for generating the next population. We take the union of the individuals to avoid the redundant individuals and retain n best fit individuals. This retained population is the new population, ready to be acted upon by different genetic operators.

6.3.6 Working of Genetic Algorithm for Virtual Topology Design: An Example

Let us see step by step the working of our genetic algorithm for virtual topology design problem for a case of $3 \times 3 \times 3$ i.e., each individual is a $3 \times 3 \times 3$ tuning matrix. We assume the values of lambda matrix, λ_{ij}=$\{\{0, 0.2, 0.3\}, \{0.8, 0, 0.4\}, \{0.1, 0.1, 0\}\}$, the mutation rate as 0.001 and the coefficient values to be A=1, B=1, C=5, D=5, we also assume the value of T to be 2. The objective function is the same as that given in the Equation 6.7.

Initially the tuning matrices are generated randomly. We start with the population of 4 where each entry in the tuning matrix is selected by tossing a fair coin. Then the fitness of each tuning matrix is found using fitness function (objective function) and the traffic matrix λ_{ij}.

A generation of our genetic algorithm begins with the reproduction. We select the mating pool of the next generation by spinning the weighted roulette wheel four times. Actual simulation of this process using coin tosses has resulted in tuning matrix 1 and tuning matrix 3 receiving two copies each and tuning matrix 2 and 4 receiving no copies each.

With an active pool of strings looking for mates, simple crossover proceeds in two steps: (1) strings are mated randomly, using coin tosses to pair off the couples, and (2) mated string couples crossover, using the crossing sites, which represents the column number in the tuning matrix being considered. The random choice of mates has selected the first string to be mated with the fourth. With a crossing site of two tuning matrices cross and yield two children. The remaining two children in the mating pool are crossed at site 1.

The last operator, mutation, is performed on bit by bit basis. We assume that the probability of mutation in this test is 0.001. With four $3 \times 3 \times 3$ bit matrices we should expect $108 \times 0.001 = 0.108 bits$ to undergo mutation during a generation. Simulation of this process indicates that no bits undergo mutation for this probability value. As a result, no bit position is changed from 0 to 1 or vice versa, during the generation. Following reproduction, crossover, and mutation, the new population is ready to be tested. To do this, we simply calculate the fitness function (objective function) values for all the new generation individuals (tuning matrices) and repeat the procedure.

6.4 Summary

The design of logical topology is complex as it requires fast solution, we have considered it in detail. The formulation permits channel sharing also. A genetic algorithm for solving the virtual topology design problem is presented in this Chapter. Attempt was made to make the genetic algorithm as general as possible, so that it can be used equally effectively for physical topology design problem and other combinatorial optimization problems.

Chapter 7

Validation and Experimental Results

Chapter Synopsis: *The way simulation was performed is discussed along with the algorithmic parameters used. Results are illustrated graphically. Validation performed to check the results of genetic algorithm are discussed. Interpretation of results based on extensive testing is given.*

7.1 Introduction

Considering the discussion of Chapter 5, about the solution to the design problems for multi-channel networks, a Mathematica [35, 33] program was developed to obtain the best tuning matrix with the maximum value for the objective function.

7.2 Development of a Program for Virtual Topology Design Problem

In this section we discuss in detail the development of program for virtual topology design problem. We present the algorithmic parameters involved along with the implementation details. The following algorithmic parameters are of significance:

- Number of nodes n and the number of channels m
- Population size N_p
- Chromosome size i.e., size of tuning matrix $(n \times n \times m)$
- Number of generations N_g
- Number of runs N_r
- Weighting terms/penalty coefficients for different terms in the objective function A, B, C, D
- Probability of mutation P_{mutate}
- Number of transceivers at each station T
- Load matrix λ_{ij} of size $(n \times n)$

7.2.1 Implementation Details

The flow of execution of program is as follows. In the beginning the program implementing genetic algorithm reads various parameters mentioned in the previous subsection. The main functions used are the objective function, flip function, mutation function, crossover function, and display function. The objective function is a straightforward implementation of the Equation (6.6) where constraints are given in the form of penalty terms. This function measures the fitness of an individual tuning matrix. We select parents with a probability which is directly proportion to their fitness values. We construct a roulette-wheel on which each member of the population is given a sector whose size is proportional to the individuals fitness values. Then we spin the wheel and whichever individual comes up becomes a parent. We have implemented this method as follows:

Step 1: A list of the fitness values of all individuals in the population is constructed.
Step 2: A random number between 0 and the total of all of the fitness in the population is generated.
Step 3: The first individual whose fitness, added to the fitness of all other elements before it, from the list in Step 1, is greater than or equal to the random number from Step 2 is returned.

Mutation is to escape getting stuck in local minima/maxima. Mutation operator changes the gene value from 0 to 1 or 1 to 0. If all the genes are subjected to mutation, we could end up with the situation where a favorable gene is altered. For this reason we keep the probability of mutation very small.

The crossover function returns two children to be added to the next generation of the population. At the start of the first generation (first iteration of the DO loop), the fitness of each individual of the population is found. Then the top n fit individuals are displayed. Then two fit individuals are picked using the selectOne function which implements roulette wheel scenario. These two individuals are mated using the crossover function which returns two newly generated children. The process of crossover is repeated until sufficient individuals are mated to produce enough children for the next generation. Then the current population is combined with the newly generated individuals by means of a union function that eliminates duplicates. These distinct individuals become the population for the next generation. The operations take place exactly in the same way, until the fitness values converge to a certain value, which is the value of the highest individual required.

7.3 Simulation Details

The evaluation of each tuning matrix requires multiplication of the tuning matrix with load matrix. The total traffic flow is obtained by summing the traffic flow at all the channels. For example the traffic in the first channel is obtained by multiplying load matrix (λ_{ij}) with the tuning matrix (X_{ijk}) for k=1 which is added to the product of load matrix and tuning matrix for k=2 and so on). Since negative values can also be obtained due to the penalty functions used, fitness values are normalized by subtracting smallest fitness value from all the other values.

We used the *elitism* technique for generation of next population. We retained the best n individuals of each generation, taking union of fitness values of n best individuals and the children obtained in each generation. Finally we took the best individual found, after observing satisfactory convergence of the individual values. We ran the algorithm many times for same number of generations and found the best values to be close in fitness value with an experimental error of 2 - 3 percent, even though the starting values were highly variant due to the random generation of individuals.

7.4 Validation and Algorithmic Check

The virtual topology design problem was solved using exhaustive search technique, linear programming technique and genetic algorithms. Validation of exhaustive search implementation, linear programming solution and genetic algorithms was done to compare the quality of results obtained. The results obtained from the genetic algorithm were compared to the results obtained from linear programming solution which was then compared to exhaustive search approach.

Linear programming technique for our problem was implemented using Mathematica [35] ConstrainedMax function. The objective function for maximizing total traffic and channel traffic was supplied to ConstrainedMax along with constraints and variables. The constraints supplied were the connectivity constraints and the limit constraints (to ensure that every entry of the tuning matrix is a 0 or 1). The variables supplied to the ConstrainedMax function were the entries of tuning matrix. In the exhaustive search approach, all possible vectors were generated for small sizes of tuning matrices. The values of the vector generated was evaluated if they satisfied the constraints of our problem.

7.4.1 Validation of Linear Programming Function ConstrainedMax using Exhaustive Search Approach

Different examples with different objective functions were taken such as counting the number of 1s in a 0-1 matrix, multiplication of two matrices. These problems were solved using exhaustive search approach and the results were checked by hand. Different kinds of objective functions with different constrained sets were solved using ConstrainedMax function, the same problem with same constraints was solved using exhaustive search. Figure 7.1 illustrates two examples. Example 1 shows the results obtained by solving *count-of-number-of-1s-in-a-0-1-matrix* problem using exhaustive search technique and linear programming technique. Similarly a small instance of $3 \times 3 \times 3$ of *virtual topology problem* is solved in Example 2. It can be seen from Figure 7.1 that the values obtained from exhaustive search technique and linear programming technique are same.

```
Exhaustive Search Technique  (Count of number of one's in a matrix)


In[1]:= <<of4.math
4

Linear Programming Technique (Count of number of one's in a matrix)

In[1]:= <<cm4.math

Out[1]= {4, {x[1, 1] -> 0, x[1, 2] -> 0, x[2, 1] -> 0, x[2, 2] -> 0,

 x[1, 3] -> 0, x[2, 3] -> 1, x[3, 1] -> 1, x[3, 2] -> 1, x[3, 3] -> 1}}
```

(a) Example 1

```
Exhaustive Search Technique  (Small instance 3.3.3 of virtual topology
problem)

Input set:

T=2;
n=3;
m=3;
l={{1.1,1.2,3.2},{1.3,1.1,3.3},{2.1,2.3,5.2}};

In[1]:= <<of1.math
41.6

Linear Programming Technique (Small instance 3.3.3 of virtual topology
problem)

In[1]:= <<cm1.math

Out[1]= {41.6,      {x[1, 1, 1] -> 0,  x[1, 1, 2] -> 1, x[1, 1, 3] -> 1,

     x[1, 2, 1] -> 0, x[1, 2, 2] -> 0, x[1, 2, 3] -> 0, x[1, 3, 1] -> 0,

     x[1, 3, 2] -> 0, x[1, 3, 3] -> 0, x[2, 1, 1] -> 0, x[2, 1, 2] -> 0,

     x[2, 1, 3] -> 0, x[2, 2, 1] -> 0, x[2, 2, 2] -> 1, x[2, 2, 3] -> 1,

     x[2, 3, 1] -> 0, x[2, 3, 2] -> 0, x[2, 3, 3] -> 0, x[3, 1, 1] -> 0,

     x[3, 1, 2] -> 0, x[3, 1, 3] -> 0, x[3, 2, 1] -> 0, x[3, 2, 2] -> 0,

     x[3, 2, 3] -> 0, x[3, 3, 1] -> 0, x[3, 3, 2] -> 1, x[3, 3, 3] ->1}}
```

(b) Example 2

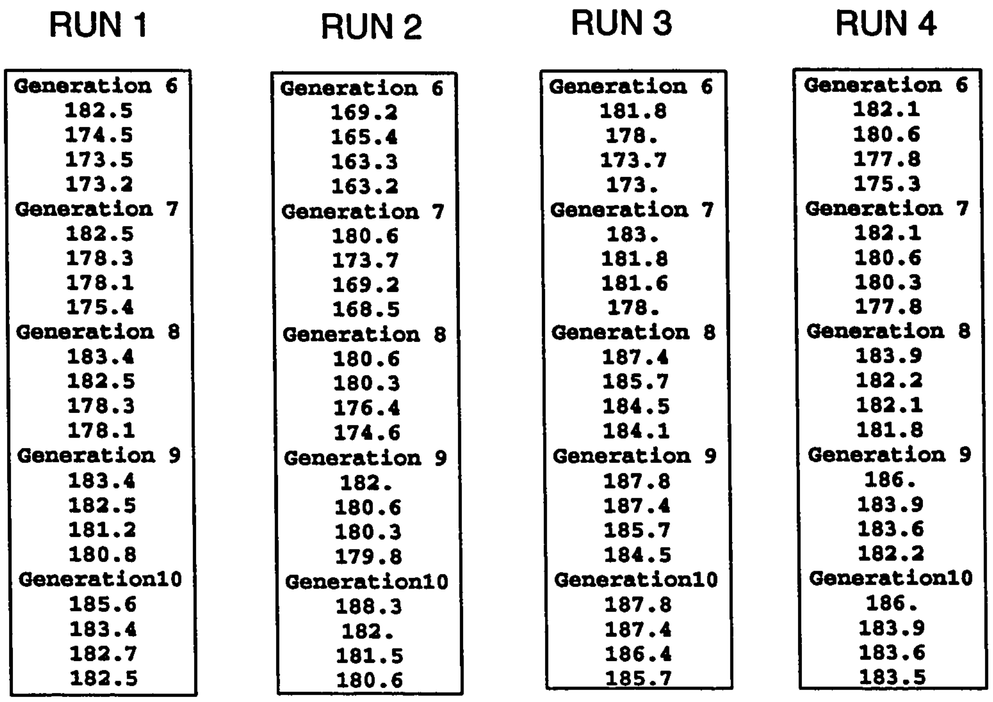
Figure 7.1: Validation of Linear Programming Results with Exhaustive Search Approach Results on Two examples

7.4.2 Validation of Genetic Algorithm using Constrained-Max

Genetic algorithm for different sizes of tuning matrices such as $3 \times 3 \times 3$, $4 \times 4 \times 4$, $5 \times 5 \times 5$, and $6 \times 6 \times 6$ were run, the results obtained were compared with the linear programming (ConstrainedMax) results for the same parameters. Figure 7.2(a) illustrate values obtained from different runs of the genetic algorithm for the same instance of the problem. Figure 7.2(b) gives the results obtained from the linear programming (ConstrainedMax) for the same instance of the problem that are comparable to the results obtained by ConstrainedMax results with a deviation of 2-7 %

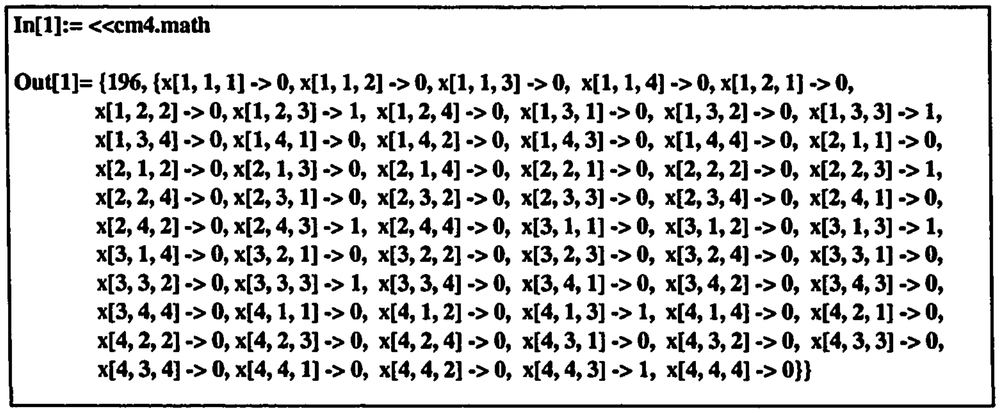
7.4.3 Large Scale Testing of Genetic Algorithm for VTDP Problem

We used a population of 40 chromosomes (i.e.,tuning matrices), each of size $n \times n \times m$ where n and m corresponds to nodes in the networks and channels in the fiber respectively. The range of values of n was from 3-10. We ran the algorithm for 25 generations. We ran 10 runs of our genetic algorithm with 20 individuals and 25 generations. The results of Run 1 and Run 2 are shown in Figure 7.3(a) and Figure 7.3(b) respectively. The plots show how the fitness values of individuals improve as the generations proceed. In the first generation many of the individuals have negative fitness values and in the final generation we could see a significant improvement



RUN 1	RUN 2	RUN 3	RUN 4
Generation 6	Generation 6	Generation 6	Generation 6
182.5	169.2	181.8	182.1
174.5	165.4	178.	180.6
173.5	163.3	173.7	177.8
173.2	163.2	173.	175.3
Generation 7	Generation 7	Generation 7	Generation 7
182.5	180.6	183.	182.1
178.3	173.7	181.8	180.6
178.1	169.2	181.6	180.3
175.4	168.5	178.	177.8
Generation 8	Generation 8	Generation 8	Generation 8
183.4	180.6	187.4	183.9
182.5	180.3	185.7	182.2
178.3	176.4	184.5	182.1
178.1	174.6	184.1	181.8
Generation 9	Generation 9	Generation 9	Generation 9
183.4	182.	187.8	186.
182.5	180.6	187.4	183.9
181.2	180.3	185.7	183.6
180.8	179.8	184.5	182.2
Generation10	Generation10	Generation10	Generation10
185.6	188.3	187.8	186.
183.4	182.	187.4	183.9
182.7	181.5	186.4	183.6
182.5	180.6	185.7	183.5

(a) Results obtained from Genetic Algorithm

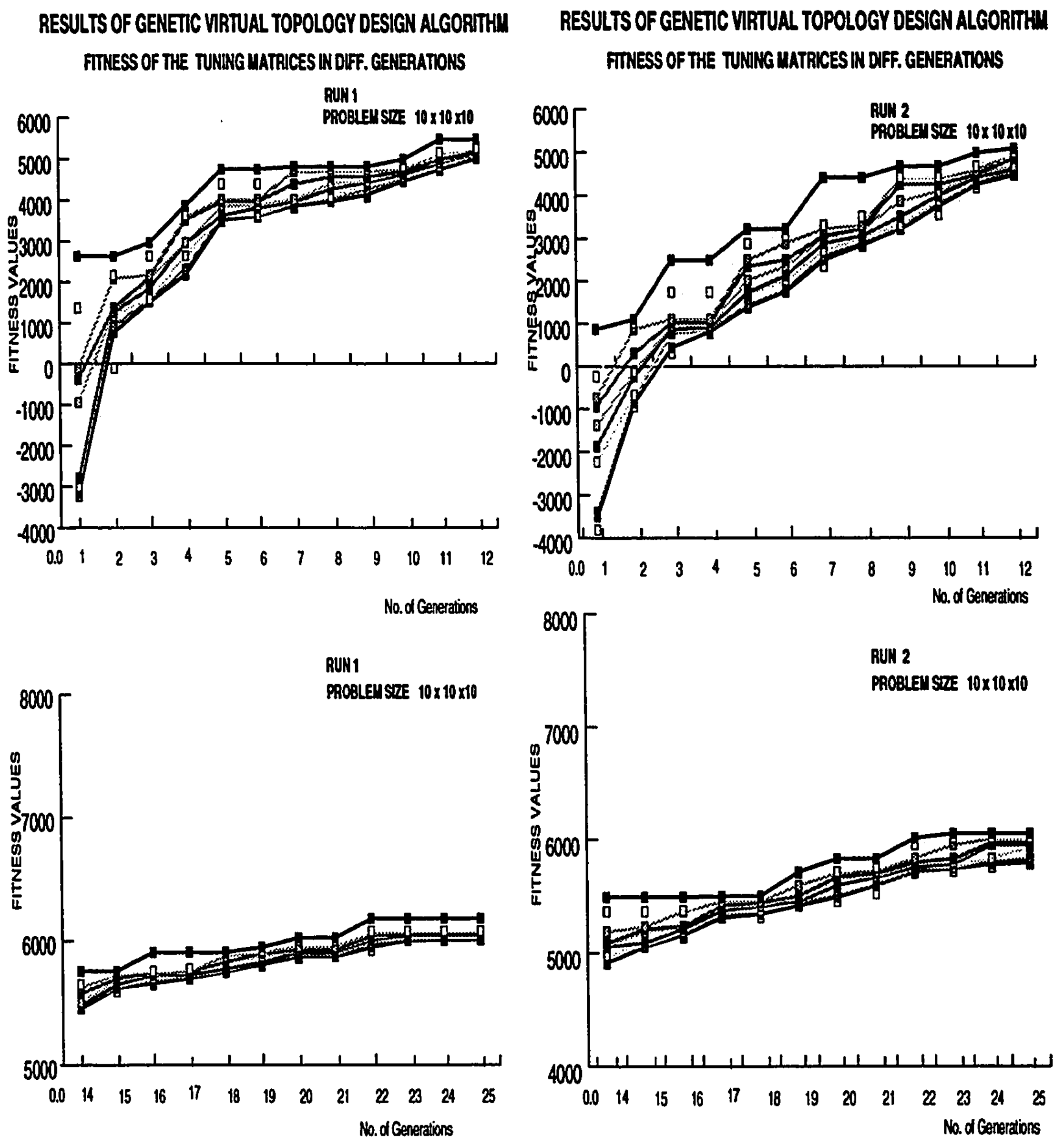


```
In[1]:= <<cm4.math

Out[1]= {196, {x[1, 1, 1] -> 0, x[1, 1, 2] -> 0, x[1, 1, 3] -> 0, x[1, 1, 4] -> 0, x[1, 2, 1] -> 0,
        x[1, 2, 2] -> 0, x[1, 2, 3] -> 1, x[1, 2, 4] -> 0, x[1, 3, 1] -> 0, x[1, 3, 2] -> 0, x[1, 3, 3] -> 1,
        x[1, 3, 4] -> 0, x[1, 4, 1] -> 0, x[1, 4, 2] -> 0, x[1, 4, 3] -> 0, x[1, 4, 4] -> 0, x[2, 1, 1] -> 0,
        x[2, 1, 2] -> 0, x[2, 1, 3] -> 0, x[2, 1, 4] -> 0, x[2, 2, 1] -> 0, x[2, 2, 2] -> 0, x[2, 2, 3] -> 1,
        x[2, 2, 4] -> 0, x[2, 3, 1] -> 0, x[2, 3, 2] -> 0, x[2, 3, 3] -> 0, x[2, 3, 4] -> 0, x[2, 4, 1] -> 0,
        x[2, 4, 2] -> 0, x[2, 4, 3] -> 1, x[2, 4, 4] -> 0, x[3, 1, 1] -> 0, x[3, 1, 2] -> 0, x[3, 1, 3] -> 1,
        x[3, 1, 4] -> 0, x[3, 2, 1] -> 0, x[3, 2, 2] -> 0, x[3, 2, 3] -> 0, x[3, 2, 4] -> 0, x[3, 3, 1] -> 0,
        x[3, 3, 2] -> 0, x[3, 3, 3] -> 1, x[3, 3, 4] -> 0, x[3, 4, 1] -> 0, x[3, 4, 2] -> 0, x[3, 4, 3] -> 0,
        x[3, 4, 4] -> 0, x[4, 1, 1] -> 0, x[4, 1, 2] -> 0, x[4, 1, 3] -> 1, x[4, 1, 4] -> 0, x[4, 2, 1] -> 0,
        x[4, 2, 2] -> 0, x[4, 2, 3] -> 0, x[4, 2, 4] -> 0, x[4, 3, 1] -> 0, x[4, 3, 2] -> 0, x[4, 3, 3] -> 0,
        x[4, 3, 4] -> 0, x[4, 4, 1] -> 0, x[4, 4, 2] -> 0, x[4, 4, 3] -> 1, x[4, 4, 4] -> 0}}
```

(b) Results obtained from Linear Programming Implementation

Figure 7.2: Validation of Genetic Algorithm Results with Linear Programming Results with One Example Instance



(a) Results obtained in RUN 1

(b) Results obtained in RUN 2

Figure 7.3: Number of Generations Vs. Fitness Values

in the fitness values. Using a suitable stopping criteria like time or convergence in values for three or more generations the algorithm can be stopped.

A program (shown in Appendix C) was written to compare the linear programming results and genetic algorithm results. This program finds out the following for a given size of the problem:

- best individual fitness value
- worst individual fitness value
- optimal fitness value which is considered to be the same as the linear programming value
- maximum percentage deviation of genetic algorithm results from linear programming result
- minimum percentage deviation of genetic algorithm results from linear programming result
- average percentage deviation of genetic algorithm results from linear programming result

The Mathematica program (shown in Appendix C) takes number of sessions N_s, number of runs N_r, number of generations N_g, number of individuals in the population N_p, etc., as input. Genetic operations are carried out on different tuning matrices in a generation. Many generations constitute a run of the program. The traffic matrix λ_{ij} changes with different sessions of the program. Many runs constitute a session.

Results obtained by Varying Values of N

Figure 7.4 shows the testing of the genetic algorithm for virtual topology design problem. The graph in Figure 7.4(a) shows how the best and worst values obtained from genetic algorithm vary from the linear programming value. The graph in Figure 7.4(b) shows the percentage deviation of genetic algorithm results from linear programming results. The maximum percentage deviation curve shows the percentage deviation of worst individual value (obtained from genetic algorithm) from linear programming value. Similarly the average and minimum percentage deviation curves show the percentage deviation of average and best individual values from linear programming value respectively. Tables 7.1, 7.2, 7.3, and 7.4 shows the best individual values, worst individual values, linear programming value, maximum percentage deviation, average percentage deviation, minimum percentage deviation corresponding to the Figures 7.4, 7.5, 7.6, and 7.7 respectively.

It can be inferred from the graphs in Figures 7.4(a), 7.5(a), 7.6(a), and 7.7(a) that the fitness values obtained from genetic algorithm are very close to the linear programming values. The average percentage deviation of the genetic algorithm values from linear programming values is very small i.e., in the range of 2-7 percent, as observed from Figures 7.4(b), 7.5(b), 7.6(b), and 7.7(b). Table 7.5 shows the average values of maximum, minimum, and average deviations for different sizes of tuning matrices. Figure 7.8 shows the maximum, minimum, and average deviation curves. From this figure it can be easily inferred that the average deviation of genetic algorithm value from linear programming value lies in the range of 2-7 percent.

Session No.	Results obtained for $3 \times 3 \times 3$ tuning matrix					
k	*BestValue*	*WorstValue*	*LPValue*	*Max.Dev.*	*Min.Dev.*	*Avg.Dev.*
1	118.234	115.922	119.242	2.78431	0.845498	1.43929
2	99.1591	88.0538	101.079	12.8862	1.89947	5.91547
3	114.354	108.179	118.675	11.6829	0.064951	5.73161
4	141.154	127.694	142.518	10.4018	0.957453	5.15496
5	91.9558	83.8085	92.3912	9.28951	0.471316	5.85088
6	152.23	139.451	153.906	9.39244	1.08937	5.723
7	91.1265	85.7078	94.5261	9.32896	3.59644	6.33702
8	123.579	115.716	124.147	9.32896	3.59644	6.33702
9	90.6829	84.1704	91.0671	7.57318	0.421853	4.89653
10	88.3484	86.7381	90.6727	4.33929	2.5633	3.30972

Table 7.1: Genetic Algorithm Values and Percentage Deviations for 3 X 3 X 3 Tuning Matrix

Session No.	Results obtained for $4 \times 4 \times 4$ tuning matrix					
k	*BestValue*	*WorstValue*	*LPValue*	*Max.Dev.*	*Min.Dev.*	*Avg.Dev.*
1	163.798	156.621	164.564	4.82694	0.465198	2.9035
2	152.23	139.451	153.906	9.39244	1.08937	5.723
3	197.147	193.837	199.086	2.63649	0.974263	1.54112
4	172.739	164.585	178.229	7.65565	3.08058	4.9918
5	166.927	160.152	171.613	6.67831	2.73053	4.60549
6	230.81	207.001	230.882	10.3433	0.0310591	4.35841
7	175.214	169.624	182.702	7.15831	4.09879	5.14854
8	91.1265	85.7078	94.5261	9.32896	3.59644	6.33702
9	175.517	166.134	183.383	9.40627	4.28928	6.41598
10	226.336	224.536	241.48	7.01667	6.27152	6.59504

Table 7.2: Genetic Algorithm Values and Percentage Deviations for 4 X 4 X 4 Tuning Matrix

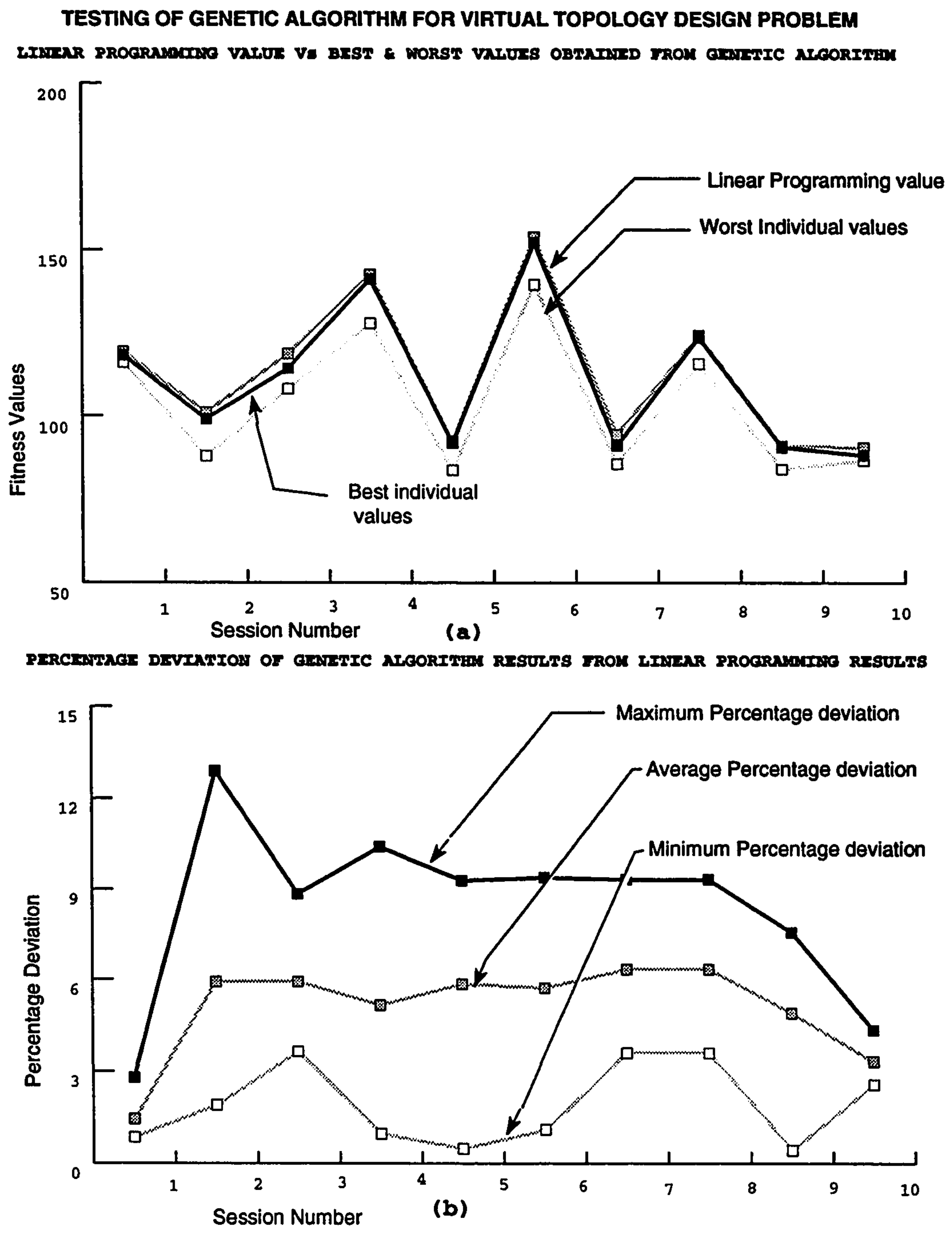
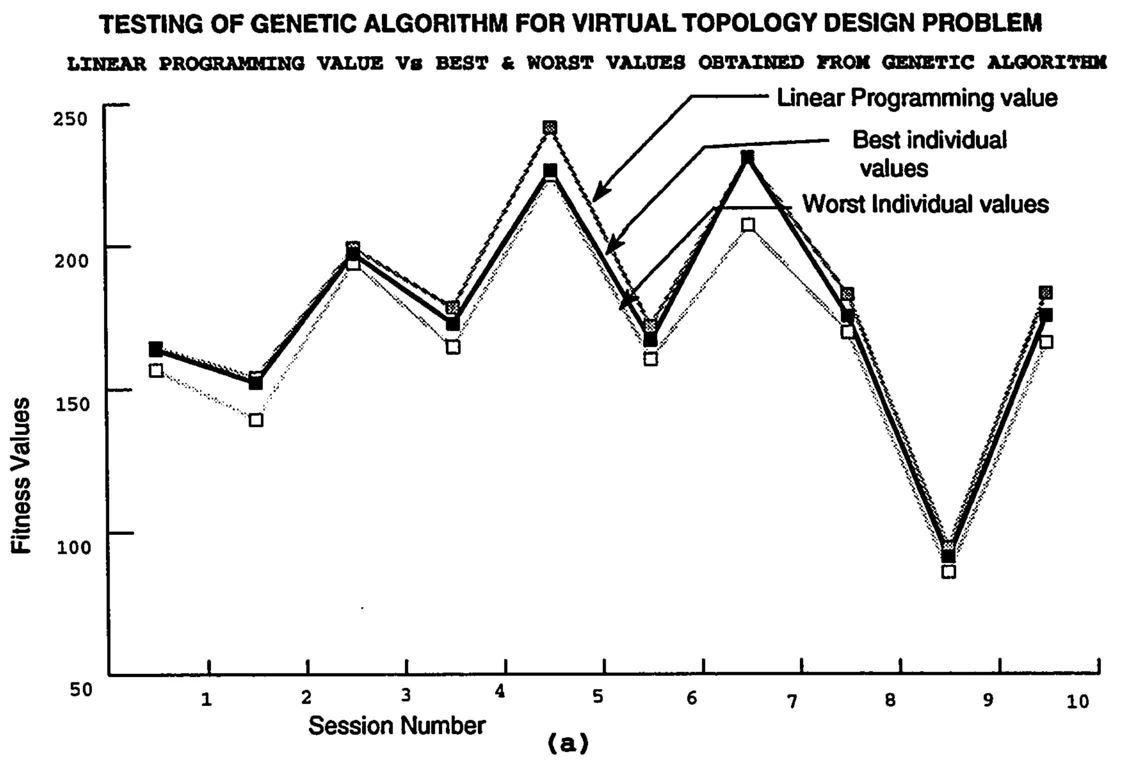


Figure 7.4: Testing for 3 X 3 X 3 Tuning Matrix



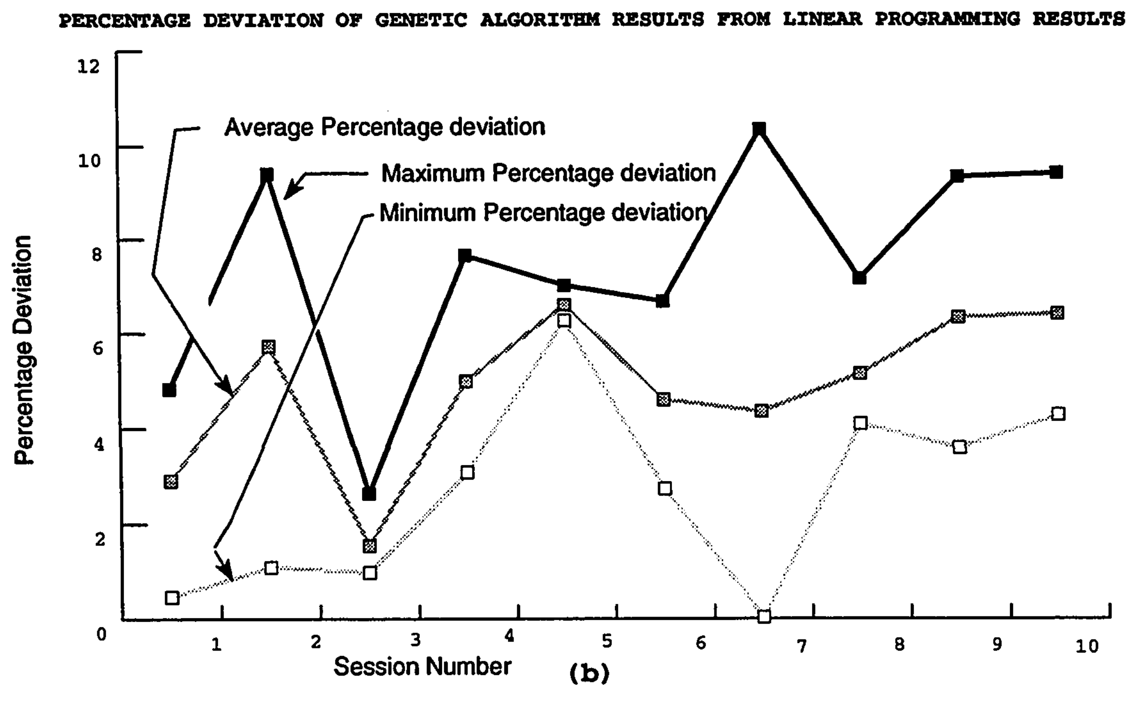


Figure 7.5: Testing for 4 X 4 X 4 Tuning Matrix

Session No.	Results obtained for $5 \times 5 \times 5$ tuning matrix					
k	*BestValue*	*WorstValue*	*LPValue*	*Max.Dev.*	*Min.Dev.*	*Avg.Dev.*
1	262.929	237.086	265.401	10.6686	0.931415	6.32016
2	325.121	302.854	334.212	9.38246	2.71998	5.16324
3	265.634	246.882	274.867	10.1812	3.35914	5.80526
4	360.635	331.534	364.926	9.15033	1.17572	4.27484
5	331.699	304.247	332.751	8.56613	0.316008	3.58874
6	313.111	293.572	322.291	8.91094	2.84821	5.88381
7	303.252	296.596	309.696	4.22999	2.08083	3.4294
8	267.587	250.185	277.663	9.89613	3.62891	6.49897

Table 7.3: Genetic Algorithm Values and Percentage Deviations for 5 X 5 X 5 Tuning Matrix

Session No.	Results obtained for $6 \times 6 \times 6$ tuning matrix					
k	*BestValue*	*WorstValue*	*LPValue*	*Max.Dev.*	*Min.Dev.*	*Avg.Dev.*
1	434.414	408.436	446.705	8.5669	2.75149	6.09734
2	543.16	489.968	546.252	10.3036	0.566047	5.55607
3	500.049	441.915	500.374	11.6829	0.064951	5.73161
4	467.584	443.237	479.231	7.51076	2.43028	5.5506
5	475.031	469.636	499.459	5.97109	4.89092	5.44995
6	482.039	436.403	490.168	10.9687	1.65848	4.8425
7	356.948	329.42	360.169	8.5374	0.894244	5.35852
8	410.953	376.135	412.681	8.85584	0.418671	3.49469

Table 7.4: Genetic Algorithm Values and Percentage Deviations for 6 X 6 X 6 Tuning Matrix

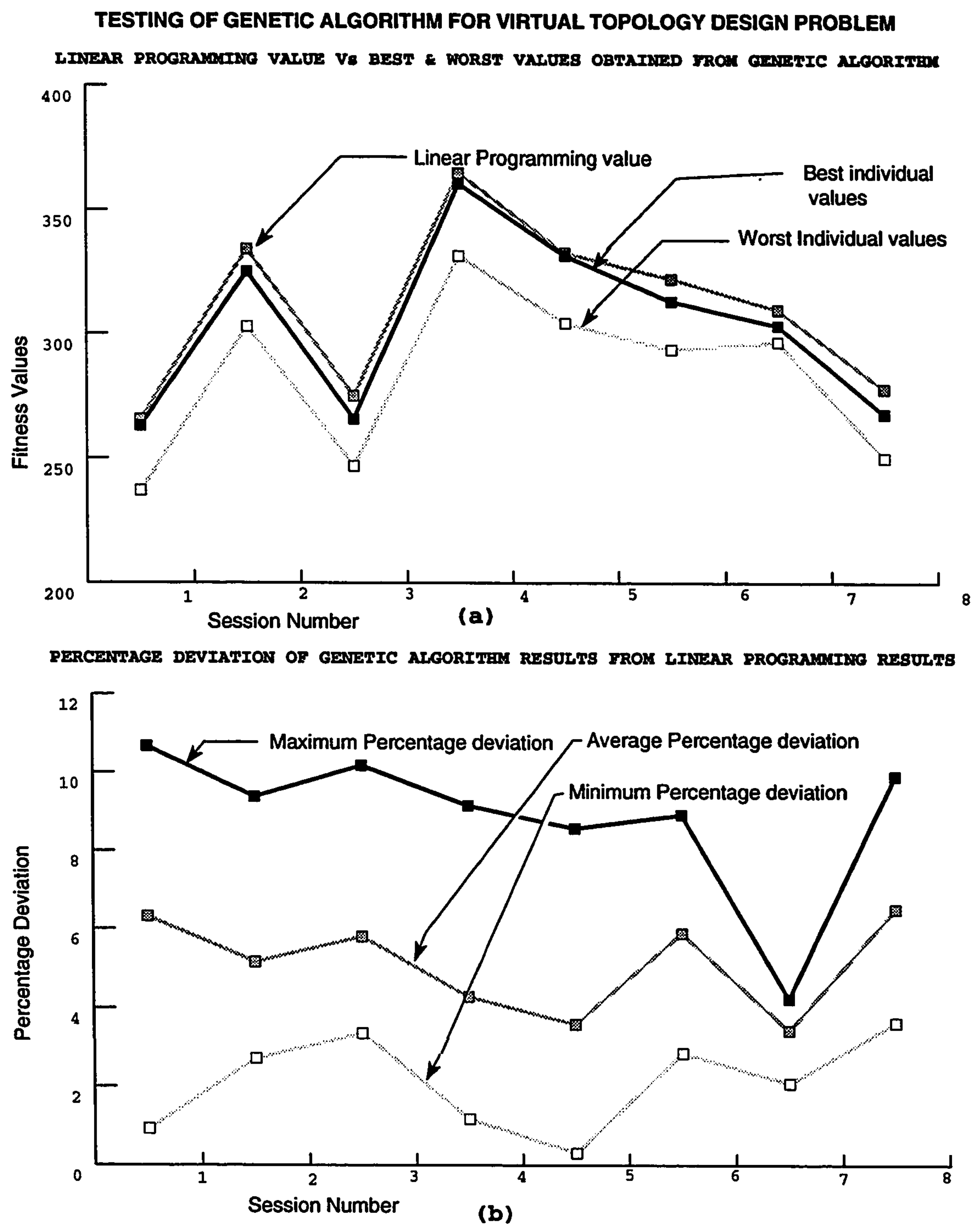


Figure 7.6: Testing for 5 X 5 X 5 Tuning Matrix

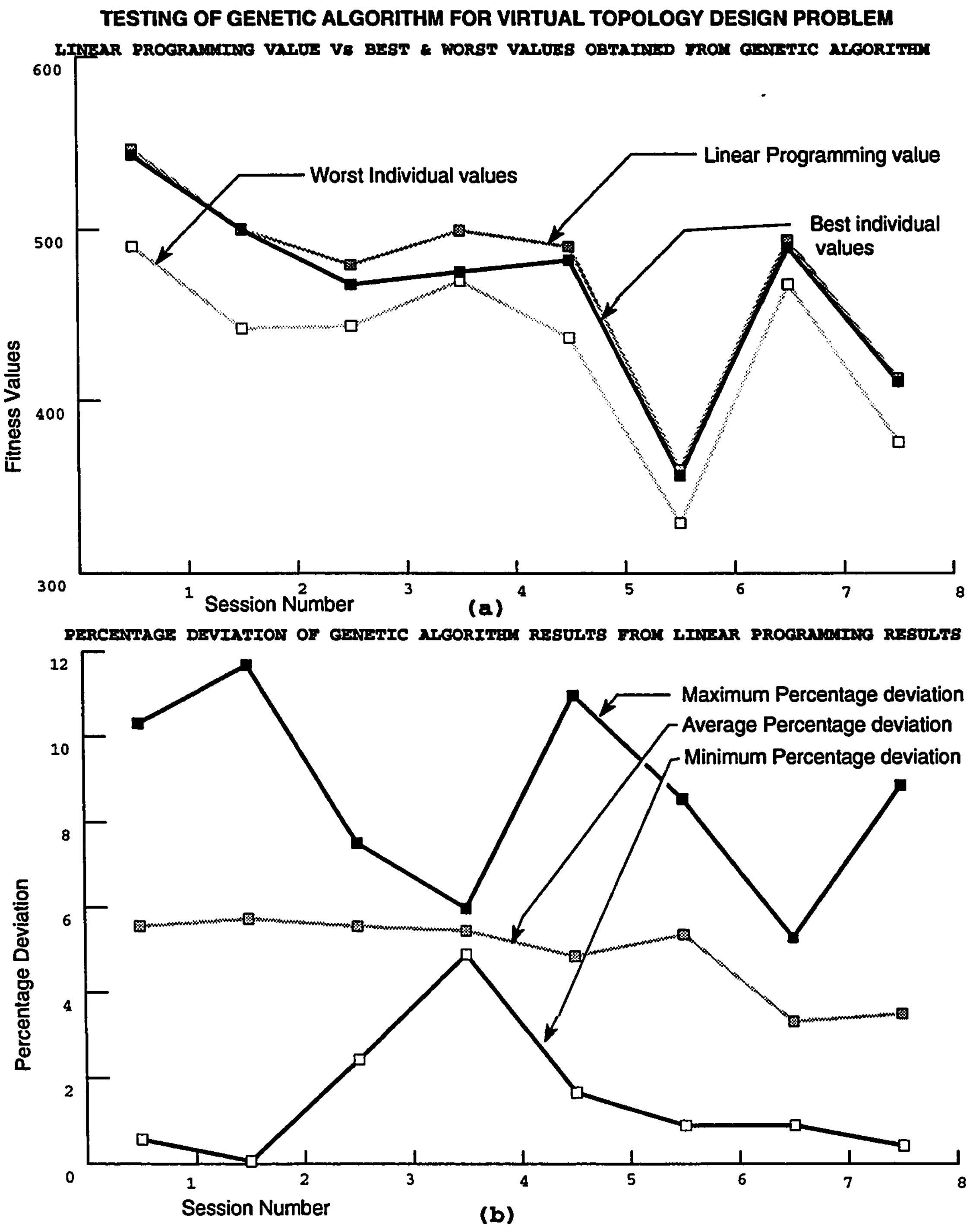


Figure 7.7: Testing for 6 X 6 X 6 Tuning Matrix

Tuning Matrix (type)	Average Percentage Deviations		
X_{ijk}	$Max.Dev.$	$Min.Dev.$	$Avg.Dev.$
3 ×3 × 3$Case$	8.700	1.552	5.070
4 ×4 × 4$Case$	7.446	2.664	4.863
5 ×5 × 5$Case$	7.099	1.707	4.098
6 ×6 × 6$Case$	7.240	1.367	4.213

Table 7.5: Average of Percentage Deviations for Different Tuning Matrices

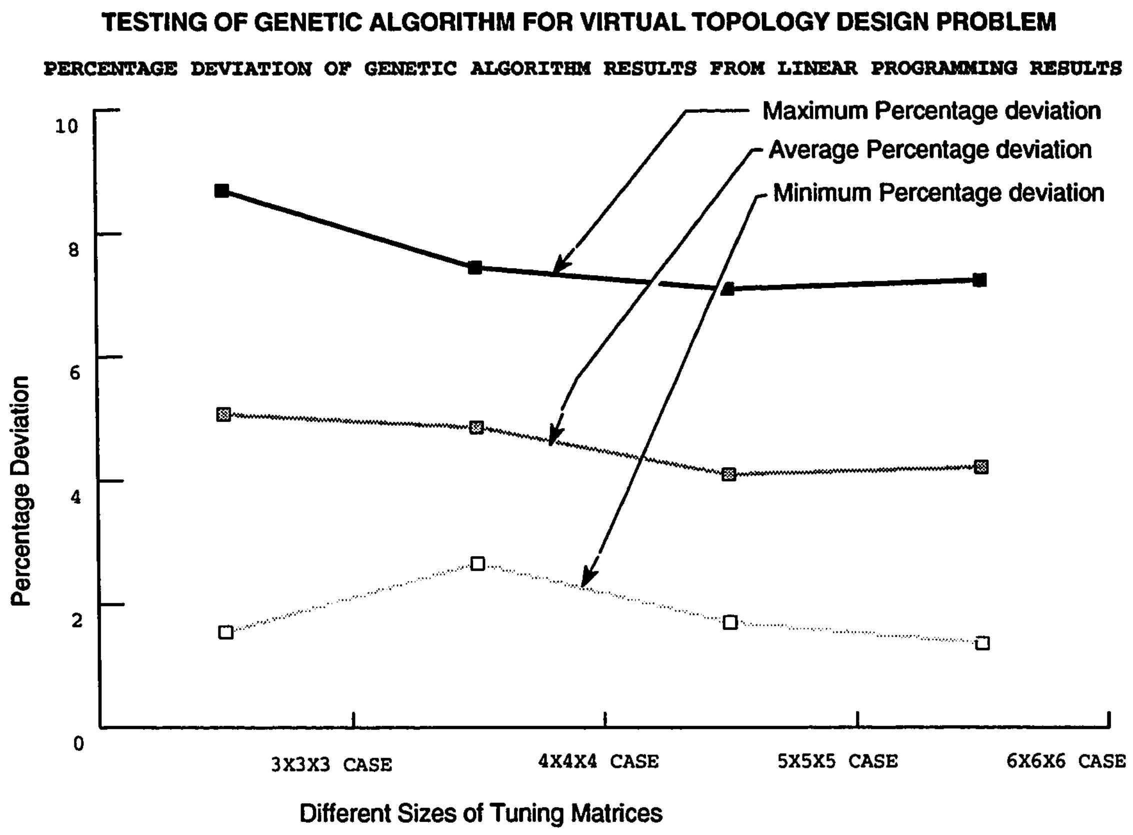


Figure 7.8: Average Deviation Curves for Different Tuning Matrices for Different Values of N

Results obtained by Varying Values of T

Figure 7.9 shows the large scale testing results obtained by varying values of T. The graph shows how the best and worst values obtained from genetic algorithm vary from the linear programming value. Tables 7.6, 7.7, and 7.8 shows the maximum percentage deviation, average percentage deviation, minimum percentage deviation for different values of T. It can be inferred from the graph in Figure 7.9 and Table 7.9 that the fitness values obtained from genetic algorithm are very close to the linear programming values. The average percentage deviation of the genetic algorithm values from linear programming values is very small i.e., in the range of 2-7 percent.

Results obtained by Varying Values of P_{mutate}

Figure 7.10 shows the large scale testing results obtained by varying values of P_{mutate}. Tables 7.10, 7.11, and 7.12 shows the maximum percentage deviation, average percentage deviation, minimum percentage deviation for different values of P_{mutate}. It can be inferred from the graph in Figure 7.10 and Table 7.13 that average percentage deviation of the genetic algorithm values from linear programming values is very small i.e., in the range of 2-7 percent.

Session No.	Results for T=1			Results for T=2		
k	*Max.Dev.*	*Min.Dev.*	*Avg.Dev.*	*Max.Dev.*	*Min.Dev.*	*Avg.Dev.*
1	13.524	0.845498	7.0725	2.78431	0.845498	1.43929
2	3.96887	1.88767	3.07386	12.8862	1.89947	5.91547
3	5.42163	1.22527	3.56151	11.6829	0.064951	5.73161
4	8.63409	5.45232	7.26105	7.57318	0.421853	4.89653
5	9.72648	4.00439	6.04321	4.33929	2.5633	3.30972

Table 7.6: Percentage Deviations for 3 X 3 X 3 Tuning Matrix for Different Values of T

Session No.	Results for T=1			Results for T=2		
k	*Max.Dev.*	*Min.Dev.*	*Avg.Dev.*	*Max.Dev.*	*Min.Dev.*	*Avg.Dev.*
1	5.49211	4.74952	5.18988	4.82694	0.465198	2.9035
2	7.63562	0.461681	4.48811	9.39244	1.08937	5.723
3	11.6396	1.2154	5.75677	2.63649	0.974263	1.54112
4	14.5551	4.70826	8.03628	7.65565	3.08058	4.9918
5	8.95491	5.35953	7.74484	6.67831	2.73053	4.60549

Table 7.7: Percentage Deviations for 4 X 4 X 4 Tuning Matrix for Different Values of T

Session No.	Results for T=1			Results for T=2		
k	*Max.Dev.*	*Min.Dev.*	*Avg.Dev.*	*MaxDev.*	*Min.Dev.*	*Avg.Dev.*
1	12.6179	1.95054	7.62304	9.38246	2.71998	5.16324
2	12.7748	5.81267	8.60355	10.1812	3.35914	5.80526
3	11.9831	0.34890	7.27017	9.15033	1.17572	4.27484
4	10.3977	3.63991	7.09717	8.56613	0.316008	3.58874
5	8.09536	4.75000	6.42250	4.22999	2.08083	3.4294

Table 7.8: Percentage Deviations for 5 X 5 X 5 Tuning Matrix for Different Values of T

Tuning Matrix	Average % Deviations for T=1			Average % Deviations for T=2		
(type)	*Max.Dev.*	*Min.Dev.*	*Avg.Dev.*	*Max.Dev.*	*Min.Dev.*	*Avg.Dev.*
3 ×3 × 3*Case*	8.262	2.806	5.400	8.700	1.552	5.070
4 ×4 × 4*Case*	9.658	3.300	7.416	7.446	2.664	4.863
5 ×5 × 5*Case*	11.174	3.300	6.964	7.099	1.707	4.098

Table 7.9: Average of Percentage Deviations for Different Tuning Matrices for Different Values of T

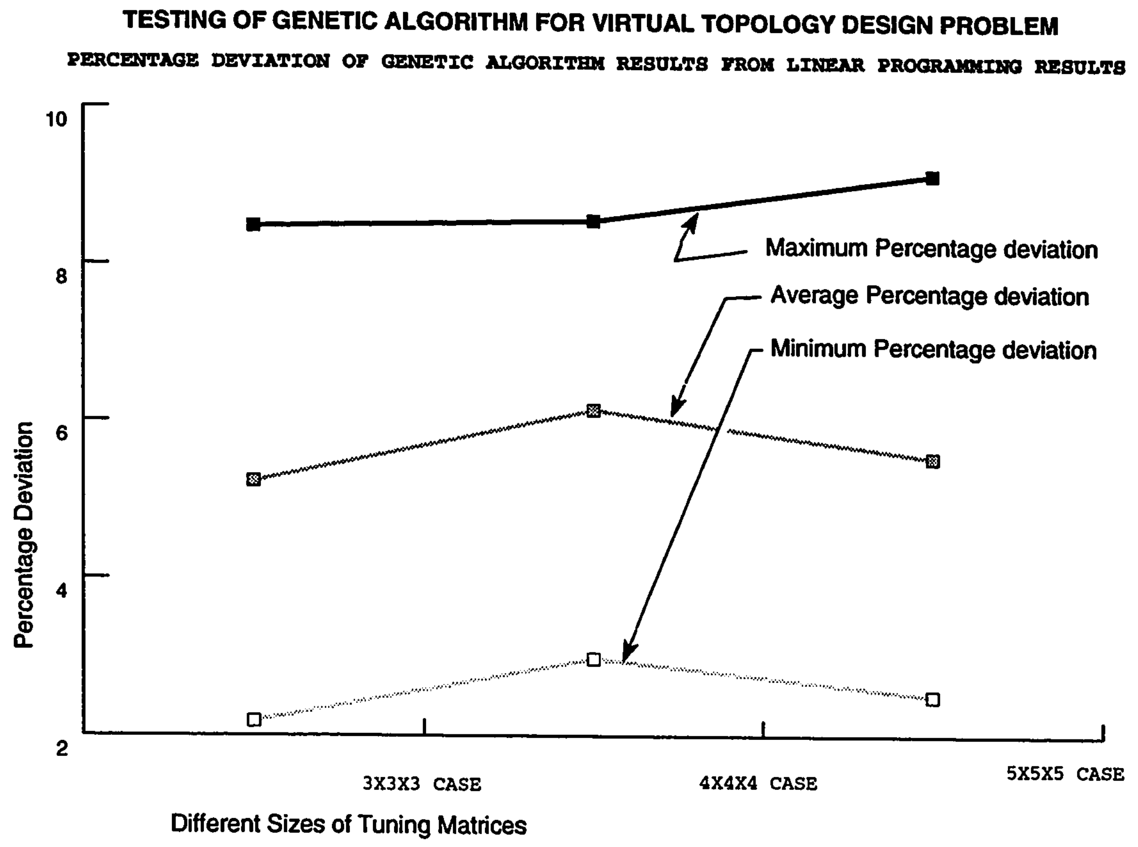


Figure 7.9: Average Deviation Curves for Different Tuning Matrices for Different Values of T

Session No.	Results for $P_{mutate} = 0.001$			Results for $P_{mutate} = 0.005$		
k	*Max.Dev.*	*Min.Dev.*	*Avg.Dev.*	*Max.Dev.*	*Min.Dev.*	*Avg.Dev.*
1	2.78431	0.845498	1.43929	9.87509	5.31594	7.19612
2	12.8862	1.89947	5.91547	6.84177	1.49316	4.68203
3	11.6829	0.064951	5.73161	11.0721	2.94187	5.99864
4	10.4018	0.957453	5.15496	9.80994	5.49872	7.27304
5	9.28951	0.471316	5.85088	12.8984	0.569658	5.46479
6	9.39244	1.08937	5.723	9.59495	0.782671	6.03059

Table 7.10: Percentage Deviations for 3 X 3 X 3 Tuning Matrix for Different Values of P_{mutate}

Session No.	Results for $P_{mutate} = 0.001$			Results for $P_{mutate} = 0.005$		
k	*Max.Dev.*	*Min.Dev.*	*Avg.Dev.*	*MaxDev.*	*Min.Dev.*	*Avg.Dev.*
1	4.82694	0.465198	2.9035	5.57271	1.64352	3.99178
2	9.39244	1.08937	5.723	6.12751	4.57028	5.46723
3	2.63649	0.974263	1.54112	8.44458	0.752183	4.68935
4	7.65565	3.08058	4.9918	12.2612	1.16052	5.24113
5	6.67831	2.73053	4.60549	11.6811	0.231681	5.50362
6	10.3433	0.031059	4.35841	8.18524	1.96221	4.49661

Table 7.11: Percentage Deviations for 4 X 4 X 4 Tuning Matrix for Different Values of P_{mutate}

Session No.	Results for $P_{mutate} = 0.001$			Results for $P_{mutate} = 0.005$		
k	*MaxDev.*	*Min.Dev.*	*Avg.Dev.*	*Max.Dev.*	*Min.Dev.*	*Avg.Dev.*
1	10.6686	0.931415	6.32016	6.45817	2.82675	4.30725
2	9.38246	2.71998	5.16324	6.97678	3.27391	4.85807
3	10.1812	3.35914	5.80526	8.33408	1.41777	4.42019
4	9.15033	1.17572	4.27484	7.96947	3.85336	6.22589
5	8.56613	0.316008	3.58874	10.258	4.88383	6.98752
6	8.91094	2.84821	5.88381	9.63495	4.38367	6.83804

Table 7.12: Percentage Deviations for 5 X 5 X 5 Tuning Matrix for Different Values of P_{mutate}

Session No.	Results for $P_{mutate} = 0.001$			Results for $P_{mutate} = 0.005$		
(type)	*Max.Dev.*	*Min.Dev.*	*Avg.Dev.*	*Max.Dev.*	*Min.Dev.*	*Avg.Dev.*
3 ×3 × 3*Case*	8.262	2.806	5.400	10.0167	2.7672	6.108
4 ×4 × 4*Case*	9.658	3.300	7.416	8.7118	1.7185	4.898
5 ×5 × 5*Case*	11.174	3.300	6.964	8.2733	3.4383	5.6083

Table 7.13: Average of Percentage Deviations for Different Tuning Matrices for Different Values of P_{mutate}

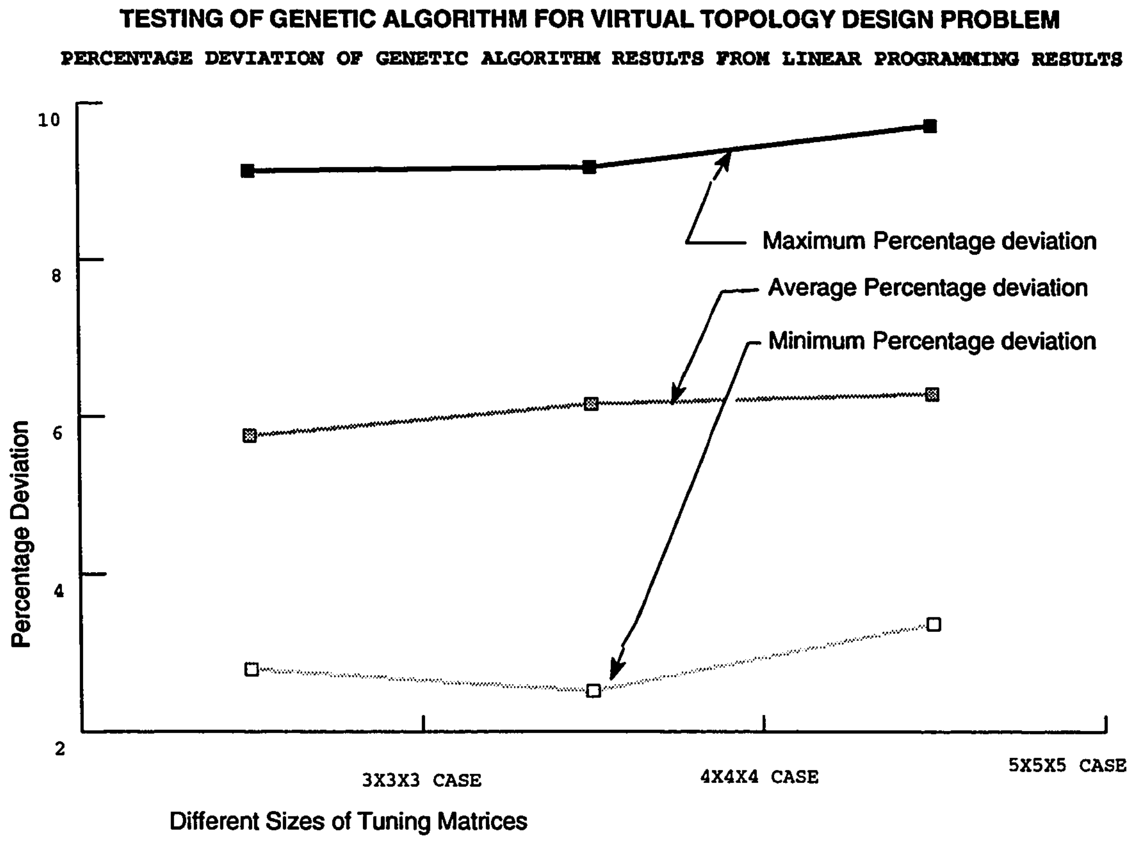


Figure 7.10: Average Deviation Curves for Different Tuning Matrices for Different Values of P_{mutate}

7.5 Observations

In this section we discuss our experiences and observations with genetic algorithms. The probability of mutation should be as small as possible. If the probability of crossover is large then the possibility of thorough mixing of chromosomal information is more. If the probability of mutation is large then there is a possibility of generating weak individuals. The elitism technique gives better results than total generational replacement, this is due to the fact that there is no surety that the next generation obtained in total generational replacement is better than the current generation, whereas in elitism best individuals of the current generation are retained in the next generation population. Large population sizes give better results in fewer generations but the computation is very slow. Final values obtained in different runs are similar even though different initial configurations are used. This speaks for the robustness of the algorithm. The number of crossover operations that are necessary in a generation is about $n/2$. If the penalty coefficients are very large then feasible solutions may not be obtained at all in a few generations.

7.6 Summary

The results obtained by applying genetic algorithm to the virtual topology problem are interpreted and compared with the results obtained from linear programming technique which in turn is validated using the brute force approach. Finally, observations and inferences were discussed.

Chapter 8

A Distributed Virtual Topology Design Heuristic

Chapter Synopsis: *A distributed heuristic for the design of virtual topology of multi-channel networks is presented. The prototypes used for parallel implementation of genetic algorithms are presented. Various messages exchanged between the nodes for correct implementation of distributed genetic algorithm are also discussed.*

8.1 Introduction

Distributed heuristics are required for realistic implementation of virtual topology design in multi-channel networks. For decentralized nature of computation and for quick updates on topology for changing loads distributed heuristics are needed. Design of distributed heuristics for adapting the logical network topology to changing traffic patterns in multihop network is an interesting problem. Two simple approaches which can be considered for distribution of information are as follows:

The first approach is based on exchanging information between neighbors, eventually creating a global picture, in each node. This approach is useful when the life time of the information is long with respect to the information propagation time. However if the life time of the information is short and the propagation time is long, nodes will be making decisions based on out dated information most of the time. This approach will also incur an additional complexity due to the cost of bandwidth dedicated for control purposes. The second alternative is to emulate global knowledge by implementing a global policy. This can be done by allowing nodes to take decisions, using the same rules. This is what our heuristic will do. Thus for a virtual topology satisfying new traffic requirements distributed virtual topology design heuristic attempts to emulate centralized virtual topology design solution. This is done to maximize throughput as the centralized virtual topology design problem has a superior ability to adapt the virtual topology to changing traffic patterns.

8.2 Parallel Genetic Algorithms

Genetic algorithms are inherently parallel. Until recently little work has been done in mapping genetic algorithms to parallel hardware. In this section we examine the implementation of our virtual topology design algorithm on parallel architectures [7].

The four prototypes which are mostly used for parallel implementation of genetic algorithms are:

- Synchronous master-slave.
- Semi synchronous master-slave.
- Distributed, asynchronous concurrent.
- Network model.

The *synchronous master-slave* prototype is shown in Figure 7.1. In this prototype as the name suggests master process coordinates k slave processes. The master process controls selection, crossover, mutation and the performance of genetic operators. The slaves perform fitness function evaluation. The scheme is straightforward and easy to implement, however it suffers from two major drawbacks. First a fair amount of time is wasted if there is variance in the time of function evaluation. Second, the algorithm is not very reliable, since it depends on proper working of the master process i.e., if the master goes down the whole system halts.

The *semi-synchronous master-slave* solves the first drawback of master-slave prototype. This prototype relaxes the requirement for synchronous operation by inserting and selecting members on the fly as slaves complete their work. This prototype is also unreliable (like master-slave) because of its dependence on a single master process.

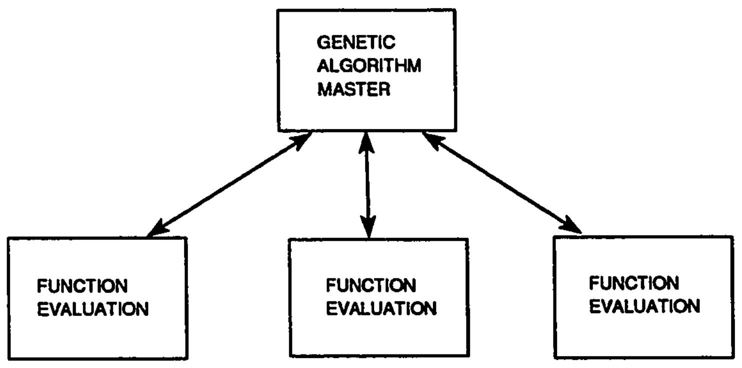


Figure 8.1: Synchronous Master Slave GA

In *asynchronous concurrent* prototype shown in Figure 7.2, k identical processors perform both genetic operations and function evaluation independently of one another, accessing a common shared memory. The shared memory requires that the processes avoid simultaneous hits on identical memory locations otherwise, there are no further timing requirements for this configuration. This scheme is slightly less straightforward to implement than either of the master-slave prototypes, but the reliability of the system is much improved. As long as one of the concurrent processes and some of the shared memory continue to function, some useful processing is performed.

The *network* prototype is depicted in Figure 7.3. In this scheme, k independent simple genetic algorithms run with independent memories, independent genetic operations, and independent function evaluations. The k processes work normally, with the exception that the best individuals discovered in a generation are broadcast to the other sub-population over a communications network. With the relatively inter-

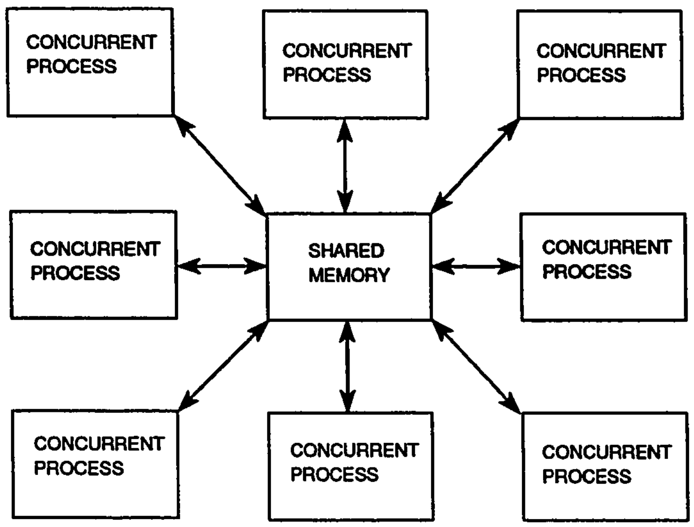


Figure 8.2: Asynchronous Concurrent GA

mittent need for communication, link bandwidth is reduced as compared to other schemes. Reliability of this scheme is high because of the autonomy of the independent processes.

For the design of virtual topology which would help in adapting to changing traffic pattern in multi-channel network we explore the following approach: Independent simple genetic algorithms for the virtual topology problem are run in each node, but there will be interaction of information among nodes from time to time. This interactions would be in the form of messages. The best individuals discovered in a generation in a node are broadcast to other nodes over the network. Here also we assume a master node which would issue the initialize and hangup messages as discussed in the next section. The objective function of each genetic algorithm in

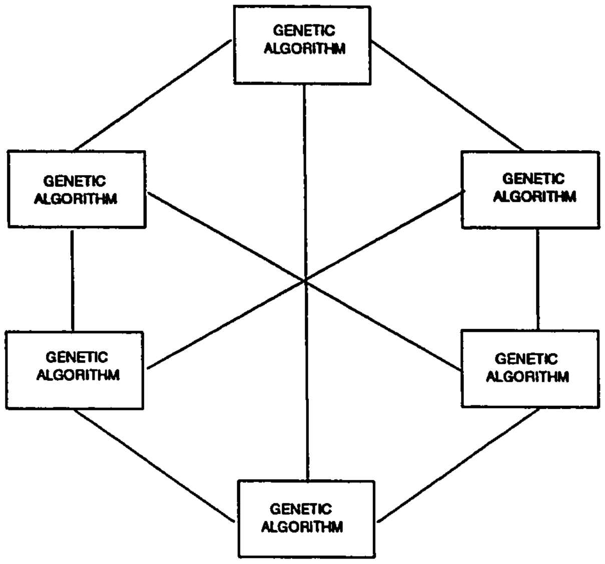


Figure 8.3: Network GA

each node will be a combination of actual function for maximizing throughput and the constraints meant for proper usage of resources (transceivers per node). Here also penalty would be imposed for violation of constraints. Due to autonomy of the independent processes, this scheme is more reliable and acceptable.

8.2.1 Distributed Genetic Algorithm for Virtual Topology Design Problem

In the distributed genetic algorithm for virtual topology design problem five types of messages are exchanged between the nodes. These messages are *Initialize*, *Request*, *Accept*, *Reject* and *Hangup*. The details of theses messages are given below:

• **INITIALIZE(nid)**

This message is sent to every node from an arbitrarily chosen supervisor node every δt seconds which would pass the traffic requirements of the other nodes. Every node after receiving this message will start processing its own genetic algorithm.

• **INFORM (Source, Destination,CON)**

This message is sent to *Destination* node from the source node in order to send the best individual (tuning matrix) obtained by it at the end of a generation. After receiving this message the destination node will include these best individuals received from other nodes in its population and processed for genetic operations.

• **SENDBEST(Source, Master, CON)**

This message is sent by each node to master node to inform it about the best individual obtained by it after all the generations.

• **REJECT(Source, Destination)**

This message acts as a rejection notice, issued when the request by a node is blocked.

• **BROADCASTBEST(Master, Destination, CON)**

This message is sent by the master to all the nodes, to broadcast the best individual it has selected from the best individuals sent to it by all the nodes. After receiving this message all the nodes will update their information (tuning matrices)

It can be seen that we are occasionally sending best individuals obtained in each generations in each node to the neighborhood node, in this way we are improving populations in parallel. We call this activity of sending best individuals to all the other nodes as migration. The advantage of *migration* is that by occasionally migrating individuals new, high fitness genetic material will be introduced into the evolving population of the node. The migrated individuals will replace the worst individuals in the population of other nodes. Migration like crossover can result in degradation if performed too frequently, and that is why it should be limited to best individuals and this operation should be less frequent. Thus, we have introduced an extra genetic operation (apart from selection, crossover and mutation) which is *migration*, which makes distributed genetic algorithm better in performance if used correctly.

8.3 Summary

Distributed heuristics are needed for realistic implementation of virtual topology design problem in multi-channel network. A distributed genetic algorithm is presented in this chapter.

Chapter 9

Conclusion

9.1 Summary

In this thesis we studied different phases of design of multi-channel network Architectures. In Chapter 3, we gave brief introduction to multi-channel network architectures then we talked about different goals and advantages of multi-channel architectures. Chapter 4 deals with the methodology for design of Multi-channel architectures.

In Chapter 5, we proposed and formulated three different problems namely physical topology design, virtual topology design, optimal routing, the design of physical topology which is to determine the optimal deployment of optical fiber and optical components to minimize the cost of fiber, the design of virtual topology is the selection of optimal channel assignment, maximizing the throughput of the network. Once a logical connectivity diagram is obtained, flows have to be optimally assigned over the links of the logical diagram, yielding the optimal routing problem, which is the minimization of maximum flow, subject to flow constraints.

Since the design of logical topology is complex as it involves real time responses, we have considered it in detail. The formulation is valid for sharing of channels since, the physical resources are limited, sharing of channels become necessary, assuming a collision free protocol. A genetic algorithm for solving the virtual topology design problem is presented in Chapter 6. Attempt was made to make the genetic algorithm as general as possible, so that it can be used equally effectively for physical topology design problem and other combinatorial optimization problems.

The results obtained by applying genetic algorithm to the virtual topology problem are interpreted in Chapter 7. Validation of experimental results obtained using Genetic Algorithm is done using the linear programming technique which in turn is validated using the brute force approach. The results obtained from the genetic algorithm are comparable to the result obtained by the linear programming technique. The factors which favours the use of genetic algorithm over other normal optimization techniques and search procedures are:

- GAs converge fast and are easy to implement .
- GAs search from a population of points, not a single point.
- GAs use payoff (objective function) information, not auxiliary knowledge.
- GAs use probabilistic transition rules, not deterministic rules.

Distributed heuristics are needed for realistic implementation of virtual topology design problem in multi-channel network. A distributed genetic algorithm is presented in Chapter 8.

9.2 Future Work

We have considered that the number of transceivers per node are fixed, this can be extended to the case of variable number of transceivers and buffers. Experimentation with different parallel techniques to find out which approach is best suited for virtual topology design problem can be done. Fine issues in the parallel implementation still need to be addressed. Future work can be carried out to design efficient protocols specially the lean protocols. With respect to multi-channel architectures work can be carried out for the development of new parallel operating systems for efficient allocation of resources to tasks, and to explore how different topologies can be realized by tuning to different channels. With respect to genetic algorithms experimentation can be done on different methods of crossover operation, probability of mutation and crossover used, use of advanced operators and techniques in genetic search like inversion, micro-operators, effective parallel implementation of genetic algorithm, tuning of penalty coefficients used.

Appendix A

Abbreviations and Nomenclature

A, B, C, D	:	Weighting terms / Penalty coefficients
CSMA/CD	:	Carrier Sense Multiple Access / Collision Detection
DQDB	:	Distributed Queuing Double Bus
FDDI	:	Fiber-Distributed Data Interface
FOLAN	:	Fiber Optic Local Area Network
GA	:	Genetic Algorithm
LAN	:	Local Area Network
LP	:	Linear Programming
MAN	:	Metropolitan Area Network
MCA	:	Multi-Channel Architecture
M-LANs	:	Multichannel Local Area Network
OSI	:	Open System Interconnection
PTDP	:	Physical Topology Design Problem
VTDP	:	Virtual Topology Design Problem
WAN	:	Wide Area Network
WDM	:	Wavelength Division Multiplexing
X_{ijk}	:	Tuning matrix (decision variable)
λ_{ij}	:	Traffic matrix or load matrix
n	:	Number of multi-channel network stations
m	:	Number of multi-channel network channels
T	:	Number of transceivers per node
$\mathbf{N_p}$	:	Population size
$\mathbf{N_g}$	:	Number of generations
$\mathbf{N_r}$	:	Number of runs
$\mathbf{P_{mutate}}$	:	Probability of mutation
$\mathbf{P_{crossover}}$	:	Probability of crossover

Appendix B

Mathematica Preliminaries

Introduction

Mathematica is a general computer software system and language intended for mathematical and other applications.

Mathematica can be used as numerical and symbolic calculator, visualization system, high-level programming language, modeling and data analysis environment, system for representing scientific knowledge, software platform, control language for external programs, and embedded system called from within other programs.

The simplest way to use *Mathematica* is like a calculator. Mathematical computations can be divided into three main classes: numerical, symbolic and graphical. *Mathematica* handles these three in a unified way.

The *Mathematica* Language

The most complete reference on *Mathematica* is [35, 33], which includes a full language description. With a programming language as rich as *Mathematica*, comprising about 800 different functions, it takes a good deal of experience to find the correct way to write a particular program. Meader's *Programming in Mathematica* [20] is an excellent tutorial, and is written by one of the developers of the system. To the reader who has never programmed in any language before, the book written by Abelson and Sussman [9] is recommended.

Mathematica supports several programming styles, including *Procedural programming*, *Functional programming*, *Rule-based programming*, etc.,. To a greater extent than imperative programming languages, such as Fortran, Pascal and C, the Lisp way of programming is the *Mathematica* way.

The self-help facility in *Mathematica* provides a brief description of each function available on-line. Typing ?*Append* gives you a description of function *Append*, while ??*Append* provides additional information which may or may not be useful.

Mathematica Interfaces

Many *Mathematica* systems are divided into two parts: the kernel, which actually performs computations, and the front end, which handles interaction with the user. The front end for *Mathematica* supports sophisticated interactive documents called *notebooks.* *Mathematica* follows many software standards that allow it to exchange material with other programs. Thus, for example *Mathematica* graphics are represented in PostScript. In addition, *Mathematica* can read data in various formats, and can generate output for systems such as C, Fortran and Tex.

Mathematica can communicate at a high level with other programs using the *MathLink* communication standard. With *MathLink*, one can use *Mathematica* to control external programs. One can also use *MathLink* to create programs that call *Mathematica* as if it were a subroutine. In this way one can set up ones own front end or control system for *Mathematica.*

Appendix C

Implementation of Genetic Algorithms for VTDP Problem

Genetic Algorithm for Virtual Topology Design Problem

```
BeginPackage["GeneticAlgorithmVTDP`"]

l={{5.9,2.9,9.0,2.2,3.3,4.5,8.9,8.7,9.9,4.8},
   {1.1,2.2,2.0,9.0,4.5,3.4,3.3,3.3,4.5,8.9},
   {3.3,3.4,2.8,9.0,4.5,4.6,2.2,4.4,6.7,5.5},
   {4.6,2.2,4.4,6.7,3.9,4.5,3.4,2.9,6.8,5.9},
   {5.9,7.7,1.1,2.9,3.3,4.5,2.2,4.4,6.7,6.7},
   {1.1,2.2,2.0,9.0,4.5,3.4,3.3,3.4,4.5,8.9},
   {3.3,3.4,2.8,9.0,2.2,4.4,3.4,4.5,3.4,6.7},
   {4.6,6.7,3.9,5.5,3.3,4.5,4.4,6.7,3.4,3.3},
   {5.9,7.7,4.4,4.5,3.4,3.3,2.2,3.3,4.5,3.4},
   {2.3,4.4,3.3,3.4,4.5,3.4,3.4,2.9,6.8,4.5}};

                  (*Load Matrix above*)
a=1;              (*Penalty Coefficient*)
b=1;              (*Penalty Coefficient*)
c=20;             (*Penalty Coefficient*)
d=40;             (*Penalty Coefficient*)
e=40;             (*Penalty Coefficient*)
n=10;             (*number of nodes inthe network*)
m=10;             (*number of channels*)
T=2;              (*number of transceivers per node*)
```

```
Np=20;              (*number of individuals in the population*)
Ng=25;              (*number of generations*)
pCrossover=0.75;    (*crossover probability*)
pMutate=0.1;        (*Mutation probability*)

pop = Table[Random[Integer, {0, 1}], {Np}, {m}, {n}, {n}];

(******************Functions Start*****************)

MatrixMult[matrix1_,matrix2_]:=Table[
      Apply[Plus,Transpose[matrix1[[i]] matrix2],{1}],{i,1,n}];

myOr=(Apply[Plus,Union[{#}]])&;

selectOne[foldedFitList_]:=
  Module[{randFit = Random[] Last[foldedFitList]},
    Position[foldedFitList, _?(#>=randFit&)]
      // First // First ];

displayBest[fitnessList_, number_]:=
  Print[ColumnForm[
    Take[Sort[fitnessList, Greater], number]]];

takeBest[fitnesList_, numb_]:= Drop[Sort[fitnesList], numb];

ObjectiveF[fgh_]:=
a Apply[Plus,Apply[Plus,Apply[Plus,Apply[Plus,MatrixMult[1,fgh],{2}]],{1}]]+
b Apply[Min,Apply[Plus,Apply[Plus,Apply[Plus,MatrixMult[1,fgh],{2}]],{1}]]-
c Apply[Plus,Apply[Subtract,{Apply[Plus,Apply[Plus,fgh,{1}],{1}],1}]] -
d (Apply[Subtract,{n T,Apply[Plus,Apply[myOr,
          Table[fgh[[i,j,k]],{i,1,n},{j,1,m},{k,1,n}],{2}],{1}]]}])^2 -
e (Apply[Subtract,{n T,Apply[Plus,Apply[myOr,
          Table[fgh[[i,j,k]],{i,1,n},{k,1,m},{j,1,n}],{2}],{1}]]}])^2;

flip[a_]:= Random[] <= a;

Mutate[pmutate_,asd_]:=
If[flip[pmutate],ReplacePart[asd,ReplacePart[dsa=asd[[
Random[Integer,{1,(n-1)}]]],dsa[[Random[Integer,{1,(n-1)}]]],
Random[Integer,{1,n-1}]],Random[Integer,{1,n-1}]]  ];

crossOver[pcross_, pmutate_, {parent1_, parent2_}]:=
  Module[{crossAt},
    If[flip[pcross],
        (* True: select cross site at random *)
      crossAt = Random[Integer, {1, n-1}];
        (* construct children *)
      {Join[Take[parent1, crossAt],
             Drop[parent2, crossAt]],
       Join[Take[parent2, crossAt],
             Drop[parent1, crossAt]]},
        (* False: return parents as children *)
      {parent1, parent2}
   ]  (* end of If *)   ];

Do[
   fitList = Apply[Plus,Map[ObjectiveF, pop],{1}];

   Print["Generation ", i, ":  Fitness of Best ", n];
   displayBest[fitList, n];
   (*MAKE THE FOLDED LIST OF FITNESS VALUES*)
   (*NORMALIZING FITLIST*)
   fitList=fitList-Min[fitList];
   fitListSum = Rest[FoldList[Plus, 0, fitList]];
   extendedL=Table[{fitList[[i]],i},{i,Length[fitList]}];
   nT=takeBest[extendedL,Max[(Length[extendedL]-Np),0]];
   indices=Table[nT[[i]][[2]],{i,Length[nT]}];
   newPop=pop[[indices]];
     (*DETERMINE THE NEW POPULATION*)
      pop=Union[newPop,Flatten[
             Table[
                 (*SELECT PARENTS*)
                   parents =
                   pop[[Table[selectOne[fitListSum],{2}]]];
                 (*CROSSOVER & MUTATE*)
                   crossOver[pCrossover, pMutate,parents],{Np}], 1]];
                 (*END OF FLATTENED TABLE*)

                 (* perform mutation *)
                   Mutate[pMutate,pop],
    {i,Ng}];
    fitList = Map[ObjectiveF, pop];
    extendedL1=Table[{fitList[[i]],i},{i,Length[fitList]}];
    nT1=takeBest[extendedL1,(Length[extendedL1]-1)];
    indices1=Table[nT1[[i]][[2]],{i,Length[nT1]}];
    newPop=pop[[indices1]]

EndPackage[]
```

Genetic Algorithm for Large Scale Testing of Virtual Topology Design Problem

```
BeginPackage["GeneticAlgorithmSimulator`"]

a=1;
b=1.5;
c=1.5 ;
T=2;
pCrossover=0.9;
pMutate=0.01;
Np=40;  (*number of individuals in the starting population*)
Ng=15;  (*number of generations*)
Nr=5;   (*number of runs*)
Ns=20;  (*number of sessions*)
m=6;    (*number of channels*)
n=6;    (*number of nodes*)
(*++++++++++++++++++++Functions+++Start+++++++++++++++++++++++++++*)
ss=Array[q,{6,6,6}];

myOr=(Apply[Plus,Union[#]])&;

(*Matrix Multiplication*)
MatrixMult[matrix1_,matrix2_]:=Table[
      Apply[Plus,Transpose[matrix1[[i]] matrix2],{1}],{i,1,n}];

(*Function for calculating fitness of the individual*)
  ObjectiveF[fgh_]:=Apply[Plus,Apply[Plus,Apply[Plus,MatrixMult[fgh,1],{2}],{1}]]
                b Apply[Plus,(Table[(Apply[Plus,Map[myOr,
                  Table[fgh[[j,k]],{j,1,n}],{1}]]-T),{k,1,m}])^2]-
                c Apply[Plus,(Table[(Apply[Plus,Map[myOr,
                  Table[fgh[[j,k]],{k,1,m}],{1}]]-T),{j,1,n}])^2];

selectOne[foldedFitList_]:=
  Module[{randFit = Random[] Last[foldedFitList]},
    Position[foldedFitList, _?(#>=randFit&)]
      // First // First ];

displayBest[fitnessList_, number_]:=
  Print[ColumnForm[
    Take[Sort[fitnessList, Greater], number]]];

takeBest[fitnesList_, numb_]:= Drop[Sort[fitnesList], numb];
```

```
flip[a_]:= Random[] <= a;

(*Mutation Operation*)
Mutate[pmut_,asd_]:=
If[flip[pmut],ReplacePart[asd,ReplacePart[dsa=asd[[
Random[Integer,{1,(n-1)}]]],dsa[[Random[Integer,{1,(n-1)}]]],
Random[Integer,{1,n-1}]],Random[Integer,{1,n-1}]]  ];

(*Crossover Operation*)
crossOver[pcross_, pmutate_, {parent1_, parent2_}]:=
  Module[{crossAt},
    If[flip[pcross],
          (* True: select cross site at random *)
        crossAt = Random[Integer, {1, n-1}];
          (* construct children *)
        {Join[Take[parent1, crossAt],
                Drop[parent2, crossAt]],
         Join[Take[parent2, crossAt],
                Drop[parent1, crossAt]]},
          (* False: return parents as children *)
        {parent1, parent2}
    ]  (* end of If *)     ];
   (*+++++++++++++++++++++++++++++++++++++++++++++++++++++++*)
   Do[
     Print["++++++++++++++++++++++++++++++"];
     Print["Session", i];
     Print["++++++++++++++++++++++++++++++"];
     l = Table[Random[Real, {0, 10}], {n}, {n}];
     restorel={0};
     Do[
      Print["++++++++++++++++++++++++++++++"];
      Print["Values obtained in the run ", i];
      Print["++++++++++++++++++++++++++++++"];
      pop = Table[Random[Integer, {0, 1}], {Np}, {m}, {n}, {n}];
      Do[
      fitList= Map[ObjectiveF,pop];
      Print["Generation ", i, ":  Fitness of Best ",4];
      displayBest[fitList, 4];
      (*MAKE THE FOLDED LIST OF FITNESS VALUES*)
      (*NORMALIZING FITLIST*)
      fitList=fitList-Min[fitList];
      fitListSum = Rest[FoldList[Plus, 0, fitList]];
      extendedL=Table[{fitList[[i]],i},{i,Length[fitList]}];
nT=takeBest[extendedL,Max[(Length[extendedL]-Np),0]];
indices=Table[nT[[i]][[2]],{i,Length[nT]}];
newPop=pop[[indices]];
(*DETERMINE THE NEW POPULATION*)
pop=Union[newPop,Flatten[
        Table[
            (*SELECT PARENTS*)
              parents =
              pop[[Table[selectOne[fitListSum],{2}]]];
            (*CROSSOVER & MUTATE*)
              crossOver[pCrossover, pMutate,parents],{Np}], 1]];
            (*END OF FLATTENED TABLE*)

            (* perform mutation *)
              Mutate[pMutate,pop],
{i,Ng}];
fitList = Map[ObjectiveF, pop];
extendedL1=Table[{fitList[[i]],i},{i,Length[fitList]}];
nT1=takeBest[extendedL1,(Length[extendedL1]-1)];
nT1=Flatten[nT1];
l1=First[nT1];
restore1=Join[restore1,List[l1]],
(*Print["Best individual ---------------------------->",Max[restore1]];
Print["Worst individual -------------------------->",Min[restore1]],*)
{i,Nr}];
ObjectiveF2= a Apply[Plus,Apply[Plus,Apply[Plus,
          Table[MatrixMult[ss[[i]],1],{i,1,n}],{2}],{1}]];
list1=Flatten[Table[0<=ss[[i,j,k]]<=1,{i,1,n},{j,1,n},{k,1,n}]];
list2=Table[Apply[Plus,Map[myOr,Table[ss[[i,j,k]],{i,1,n},{k,1,n},
                  {j,1,m}],{2}],{1}][[y]]<=T,{y,1,n}];
list3=Table[Apply[Plus,Map[myOr,Table[ss[[i,j,k]],{j,1,n},{k,1,n},
                  {i,1,m}],{2}],{1}][[y]]<=T,{y,1,n}];
constraints=Join[list1,list2,list3];
l2=First[Flatten[ConstrainedMax[ObjectiveF2, constraints,
{ss[[1,1,1]], ss[[1,1,2]], ... ,ss[[6,6,6]]}]]];
restore1=Rest[restore1];
Print["++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"];
Print["+++++++++++++++RESULTS OBTAINED+++++++++++++++++++++++++"];
Print["++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"];
(*Print[restore1];*)
Print["Best individual---------------------------------->",Max[restore1]];
Print["Worst individual--------------------------------->",Min[restore1]];
Print["Value obtained from linear programming solution-->", l2];
l3=l2-l1;
```

```
      l5=l2-restore1;
      (*Print[l5];*)
      l6=Min[l5];
      l7=Max[l5];
      l8=(Apply[Plus,l5])/Length[l5];
      (*Print["Difference (LP value-GA best individual values)-->", l5];*)
      l41=((l7/l2)*100);
      l42=((l6/l2)*100);
      l43=((l8/l2)*100);
      Print["Maximum percentage deviation--------------------->", l41];
      Print["Minimum percentage deviation--------------------->", l42];
      Print["Average percentage deviation--------------------->", l43];
      Print["++++++++++++++++++++++++++++++++++++++++++++++++++++++"],
 {i,Ns}]
EndPackage[]
```

Linear Programming Implementation of Virtual Topology Design Problem

```
BeginPackage["LinearProgrammingVTDP`"]
(*This program is the genetic algorithm implementation of VTDP problem*)

a=1;                (*Penalty Coefficient*)
T=2;
m=6;
n=6;
l={{1.1,1.2,3.2,1.0,3.1,4.5},
   {1.3,1.1,4.9,0.0,3.3,3.1},
   {2.0,9.0,9.0,4.5,3.4,3.3},
   {2.1,2.3,5.2,5.6,3.2,8.0},
   {2.2,2.0,9.0,3.3,4.5,3.4},
   {1.0,3.1,4.5,1.3,1.1,4.9}};

myOr=(Apply[Plus,Union[#]])&;

ss=Array[x,{6,6,6}];

MatrixMult[matrix1_,matrix2_]:=Table[
      Apply[Plus,Transpose[matrix1[[i]] matrix2],{1}],{i,1,n}];
ObjectiveF= a Apply[Plus,Apply[Plus,Apply[Plus,
                Table[MatrixMult[ss[[i]],l],{i,1,n}],{2}],{1}]];
```

```
list1=Flatten[Table[0<=ss[[i,j,k]]<=1,{i,1,6},{j,1,6},{k,1,6}]];
list2=Table[Apply[Plus,Map[myOr,Table[ss[[i,j,k]],{i,1,n},{k,1,n},
      {j,1,m}],{2}],{1}][[y]]<=T,{y,1,4}];
list3=Table[Apply[Plus,Map[myOr,Table[ss[[i,j,k]],{j,1,n},{k,1,n},
      {i,1,m}],{2}],{1}][[y]]<=T,{y,1,4}];
constraints=Join[list1,list2,list3];
ConstrainedMax[ObjectiveF, constraints,{ss[[1,1,1]], ss[[1,1,2]], ...,ss[[6,6,6]]}]
EndPackage[]
```

Exhaustive Search Implementation of Virtual Topology Design Problem

```
BeginPackage["ExhaustiveSearchVTDP`"]
a=2;                (*Penalty Coefficient*)
b=15;
T=2;
l={{1.1,1.2,3.2},{1.3,1.1,3.3},{2.1,2.3,5.2}};

Ored[x_]:=Apply[Plus,Union[x]];

MatrixMult[matrix1_,matrix2_]:=Table[
      Apply[Plus,Transpose[matrix1[[i]] matrix2],{1}],{i,1,3}];

displayBest[fitnessList_, number_]:=
          Print[ColumnForm[Take[Sort[fitnessList, Greater], number]]];

MatrixForm[pop=Partition[Partition[Partition[Flatten[
Outer[List,{0,1},{0,1},{0,1},{0,1},{0,1},{0,1},
           {0,1},{0,1},{0,1},{0,1},{0,1},{0,1},
           {0,1},{0,1},{0,1},{0,1},{0,1},{0,1}]],3],3],2]];

ObjectiveF[x_List]:= a Apply[Plus,Apply[Plus,Apply[Plus,Table[l*x]]]]+
b Apply[Min,Apply[Plus,Apply[Plus,Table[l*x]],{1}]];

EndPackage[]
```

Bibliography

[1] A.S.Acampora. A multi-channel multihop lightwave network. *Proc. IEEE GLOBECOM '87*, pages 37.5.1–9, 1987.

[2] Hlichuj A.S.Acampora, Karol. Terabit lightwave networks: The multihop approach. *AT&T Journal*, Dec. 87.

[3] Bannister and Gerla. Design of the wavelength division optical network. *IEEE ICC '90*, 90.

[4] Yitzak Birk. Fiber-optic bus-oriented single-hop interconnections among multi-transceiver stations. *Journal of Lightwave Technology, IEEE.*

[5] Yitzak Birk. Fiber-optic bus-oriented single-hop interconnections among multi-transceiver stations. *Journal of Lightwave Technology, IEEE*, pages 1657–1664, December 1991.

[6] Lawrence Davis. *Genetic Algorithms and Simulated Annealing.* Wiley Eastern, 1987.

[7] D.E.Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Wiley Eastern, 1989.

[8] Green. Exploiting photonic technology for gigabit computer networks. *North Holland*, 1991.

[9] G.J.Sussman H. Abelson and J. Sussman. *Structure and Interpretation of Computer Programs.* MIT Press, Cambridge, Mass, 1985.

[10] M. G. Hluchyj and M.J.Karol. Shufflenet: An application of generalized perfect shuffles to multihop lightwave networks. *J. of Lightwave Technology*, pages 1386–131397, 1991.

[11] H.P.Williams. *Model Building in Mathematical programming.* Wiley Eastern, 1990.

[12] H.S.Stone. Parallel processing with the perfect shuffle. *IEEE transactions on Computers*, 20, February 1971.

[13] I.Chlamtac and A.Ganz. Design and analysis of very high speed network architectures. *IEEE Trans. Commun*, 36(3):252–262, March 1988.

[14] I.Chlamtac and A. Ganz. Lightpath communications:an approach to high bandwidth optical WANs. *IEEE Trans. Commun.*

[15] A.Ganz I.Chlamtac and G.Karmi. Purely optical networks for terrabit communications. *Proc. IEEE INFOCOM'89*, pages 887–896, 1989.

[16] J.Bannister. The wavelength division optical network. *PhD. thesis,UCLA*, 1990.

[17] J.C.Palais. Fiber optic communications. *Prentice Hall, Englewood Cliffs,NJ*, 1988.

[18] J.F.P.Labourdette and A.S.Acampora. Logically rearrangeable multihop networks. *IEEE Trans. on Commun.*, 39(8):1223–1230, Aug. 1991.

[19] K.N.Sivarajan and Ramaswami. Multihop lightwave networks based on de Bruijn graphs. *Proc. of INFOCOM '91*, 1991.

[20] R. Meader. *Programming in Mathematica.* Addison-Weley, 1989.

[21] M.R.Garey and D.S.Johnson. Computers and intractability: A guide to the theory of NP-completeness. *Freeman, New York*, 1979.

[22] ans H.D.Sherali M.S.Bazara, J.J.Jarvis. *Linear Programming and Network Flows.* Wiley Eastern, 1990.

[23] M.S.Goodman. The LAMDANET multiwavelength network. *IEEE J. Sel. Areas Commun.*

[24] B. Mukerjee. WDM based local-lightwave networks. *IEEE Network*, pages 20–32, July 1992.

[25] R. M. Newman. The QPSX MAN. *IEEE Communications Magazine.*

[26] N.R.Dono. A wavelength division multiple access network for computer communications. *IEEE J. Sel. Areas Commun.*, pages 983–994, August 1990.

[27] Rajiv Ramaswami. Multiwavelength lightwave networks for computer communications. *IEEE Communications Magazine*, February 1993.

[28] R.Gidron. Teranet: A multigigabit/second hybrid circuit/packet switched lightwave nertwork. *Proc. OE/Fibers '91 Boston, MA*, pages 1579–1584, 1991.

[29] F. E. Ross. FDDI: a tutorial. *IEEE Communications Magazine*, pages 10–17, May 1986.

[30] M. Schwartz. Network management and control in broadband telecommunication networks. *Computer and Systems Sciences*, 72, 1991.

[31] K. Shahookar and P. Mazumder. VLSI cell placement techniques. *ACM Computing Surveys*, 23, June 1991.

[32] K. N. Sivarajan. Multihop local topologies in gigabit lightwave networks. *IEEE Lightwave Telecom. Syst. Magazine*, August 1992.

[33] Steven Skiena. *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica.* Wiley Eastern, 1990.

[34] T.S Wailes and D.G Meyer. Multiple channel architecture: A new optical interconnection strategy for massively parallel computers. *IEEE Journal of Lightwave Technology*, pages 1702–1716, December 1991.

[35] Stephen Wolfram. *Mathematica: A System for Doing Mathematics By Computer.* ., 1991.

Vita

- Mohammed Abdul Azeem Abed
- Born at Hyderabad, India in December, 1969.
- Received Secondary School Certificate from St.Xavier's High School, Hyderabad, India in June, 1984.
- Received Bachelor's degree in Computer Science and Engineering from Osmania University, Hyderabad, India in April, 1990.
- Completed Master's degree requirements at King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia in July, 1993.