University of Windsor
Scholarship at UWindsor

Electronic Theses and Dissertations | Theses, Dissertations, and Major Papers

2001

Investigation of genetic algorithms in optical network design.

Sridhar. Tadicherla
University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation

Tadicherla, Sridhar., "Investigation of genetic algorithms in optical network design." (2001). *Electronic Theses and Dissertations*. 2504.
https://scholar.uwindsor.ca/etd/2504

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI®

Investigation of Genetic Algorithms in Optical Network Design

by

Sridhar Tadicherla

A Thesis submitted to the Faculty of Graduate Studies and Research
in Partial Fulfillment of the Requirements for the Degree of Master of Science
at the University of Windsor
Windsor, Ontario,
Canada 2001

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisitions et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-67603-X

Canada

175264

Sridhar Tadicherla 2001
@ All Rights Reserved

Abstract

The recent advances in light wave communication technology over the past several years enabled us to share enormous optical bandwidth among users in local, metropolitan and wide-area networks. But with the increase in number of users utilizing the network it leads to congestion of network. Congestion is a major issue while evaluating the performance of a network. The lower the congestion in a network, the less is the cost of the hardware (optical & electronic). The problem we are studying is that of designing an optimum ordering of nodes if we are using a logical de Brujin topology. To determine this, we will use the Genetic Algorithm approach. Our approach involves the use of a new cross over strategy (sub-graph cross over) to solve the problem of designing large networks.

To My Parents

Acknowledgements

I would like to thank my supervisor Dr. Subir Bandyopadhyay for his able guidance and support all along my master's program. His advice, suggestions and comments have been invaluable. My sincere thanks go to the thesis committee, Dr. A. Jaekel, Dr. P.Andiappan and Dr. S.Goodwin for their interest and support.

Finally, I would like to thank my parents for their love and care, which helped me to become a better person.

Table of Contents

List of Figures:

List of Tables:

Chapter 1

Introduction

The recent advancements in optical networking technologies encourages us to search for a way to achieve the goal of high-speed data transfer as well as to cope up with the explosion of bandwidth demands. The emerging new technologies in the field of light wave communications over the past several years enabled us to share enormous optical bandwidth among users in local, metropolitan and wide-area networks. Optical Wavelength Division Multiplexed (WDM) networking technology is leading a bandwidth revolution in the infrastructure of the next generation Internet and beyond. Demand for bandwidth is rapidly increasing with the possibility of recent applications such as electronic commerce, video on demand, and global cooperative work. Optical fibers are now widely deployed by various telecommunication companies. A single strand of optical fiber with huge bandwidth, low signal attenuation and low signal distortion can support several hundred times of current traffic load by employing WDM technology. All optical wavelength division multiplexed networks using wavelength routing are considered to be the best bets for the next generation of wide area optical networks.

1.1 Motivation

The problem of designing a good logical topology for a given network is always an interesting and challenging thing due to the reason that it is very difficult to find the corresponding routes between any source-destination pair and it has attracted

considerable interest. This is an important problem because it determines the traffic load on the links and of the network there by helping in determining the maximum traffic a network can handle. Some recent approaches employed various optimization techniques to solve this problem. This usually results in a logical topology, an arbitrary graph, constructed in such a way to minimize the congestion of the network. The main drawback of these approaches is that as the number of nodes in the network increased, the problem quickly becomes intractable.

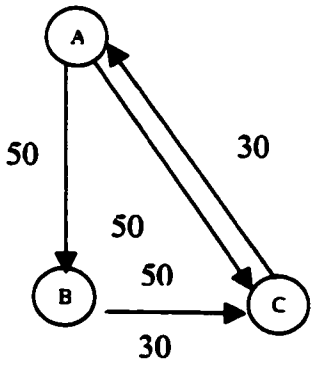
1.2 Problem outline

In a wavelength routed optical network, if the set of light paths are defined in different ways, there will be different logical topologies. An optical network topology can be single hop or multiple hops. In a single-hop network [ZA95], every source destination pair has a light path between them so that optical signals carry information from a source node to a destination node without undergoing any opto-electronic conversion. In a multi-hop network, the traffic from a source node S to a destination node D might travel using a lightpath L_1 from S to the node S_1, another lightpath L_2 from S_1 to the node S_2 ... using lightpath L_k from node S_k to the node D. In an arbitrary multi-hop topology there is no predefined connectivity pattern. A regular Multi-hop networks such as ShuffleNet, Toroid, de Bruijn graph, GEMNET, hypercube is another approach, which provide simple routing schemes due to their well-defined connectivity patterns.

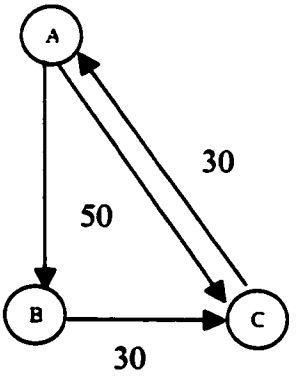
A single hop topology suffers from two major drawbacks; it needs expensive wavelength-agile transceivers and also pre transmission coordination between the prospective communicators. These different logical topologies have different

performance characteristics and are of great interest in further research to determine the optimal logical topology for a given network with a specified physical topology, expected traffic between all node pairs and other physical characteristics. Designing of a multi-hop logical optical network architecture involves meeting various criteria such as small average internodal distance, as it inversely proportional to the network utilization, adding or deleting nodes from the network should result in a minimum impact on network configuration and performance [SVS99]. These different logical topologies have different performance characteristics and it is an important research area to determine the optimal logical topology for a given optical network with a specified physical topology, expected traffic between all node pairs and other physical characteristics. In general, for designing optical networks, the important factors to be considered for evaluating the performance of a multi-hop network are the *delay* and the *congestion* [RS98] where the delay is the time taken by a signal to travel from the source node to the destination node and the congestion is the maximum load offered to any logical link.

Following fig 1.1 is a simple example showing the congestion, may be minimizing by choosing paths in a different manner



(a) Congestion is 80



(b) Congestion is 50

In fig 1.1 (a) 50 units of traffic is sent from A to C through the path A→B→C requiring 2 hops and 30 units of traffic from B to A through the path 2 hops, which gives the congestion value of 80(Load on edge BC). On other hand in figure 1.1 (b) A and B are sending same amount of traffic but over a different path. A is sending 50 units in a single hop from A→C and B is sending 30 units in a same way as in fig1.1 (a) along path B→C→A. The resulting value of congestion is 50 (along A→C), which is lower than congestion in fig1.1 (a). Therefore, by reducing the number of hops required to transmit data from A to C we are able to reduce the congestion of the network. It is expected that logical topologies with low diameter should perform better than irregular topologies, since there is and upper bound on the maximum path length [RS96].

1.3 Previous Approaches

In the past few years there has been a considerable research going on in the field of designing logical topologies, one popular approach is to formulate the problem as a combined design problem of logical topology and routing and to divide the problem into two major sub-problems

- Logical Topology Design Problem
- Routing Problem

There are two important approaches [RS96][RS98] to solve the problem of designing logical topologies using

a) Mathematical and optimization techniques and

b) Using regular topologies.

The first approach utilizes the mathematical optimization techniques to solve both logical design and routing problem simultaneously. It is based on the traffic demands of the various node pairs and attempts to design an optimal logical topology as well as a routing scheme over the given physical topology [RS96]. The first problem formulation involves MILP [RS96] that generates a large number of constraints, which is a major drawback even for small-sized networks. As the MILP formulation was found to be infeasible for practical networks, attempts were made to simplify the problem by dividing it into two parts. First to determine a logical topology using some heuristics and then find an optimal routing on that topology by formulating the routing as a LP problem. In regular topology approach traffic demands are ignored and logical topologies are designed with some desirable graph theoretical properties such as pre-determined routing, low diameter and high connectivity

Gazen and Ersoy [GAZEN99] developed an approach using genetic algorithms to solve the logical topology design problem. Here the problem again is divided into two independent problems, the connectivity problem and the flow assignment problem. The connectivity problem is solved using genetic algorithms, as it is best suited for the discrete optimization problems. It introduces minimum-hop routing with Flow deviation as their objective function. The flow deviation method works with a flow assignment algorithm. The two solutions are the combined linearly so as to improve the result. But again this approach uses mathematical linear equations to solve the flow assignment problem.

1.3 Proposed Approach

The problem addressed in this thesis can be presented as follows: For a given traffic matrix, find out a good mapping of physical nodes onto the nodes of the logical topology, in such a way that the congestion is minimized i.e., a proper ordering of nodes is required to generate a logical topology such that the congestion is minimized. Mapping of the existing nodes onto the nodes in the logical topology would generate various logical topologies. These different mappings will have different performance characteristics and it is an important research area of interest to find the best topology with minimum or optimal logical topology for a given optical network with a specified physical topology, expected traffic between all node pairs and other physical characteristics.

Different logical topologies and topological design techniques were proposed. It is shown that de Brujin graphs as logical topologies can support significantly more nodes than a ShuffleNet [HK88]. In the de Brujin graph [SR91], it is known that there are a number of nodes disjoint paths from a source to a destination, each having almost the same number of nodes as in the shortest path route [SSB88]. This means that the routing scheme can make full use of the rich interconnections available in a regular graph and the fact that numerous, relatively short, route exist between any source and destination. Given a target logical topology, it is much easier to find a good mapping between the logical and physical nodes such that the congestion is minimized [HJBS00].

For example Table 1-1 represents a traffic matrix and figure1-1. is the De Bruijn graph. The main goal of this thesis is to find a good mapping of physical topology (A, B, C...) to logical topology where the nodes are (000,001,., 111) in the De Brujin graph. Let

node A be mapped to 001, B to 010 and C to 000 etc. Genetic algorithms are used to solve the above problem by coming up with a good ordering of nodes in such a way that the target logical topology will be one with minimal congestion. Genetic algorithmic approach will generate different sets of ordering of nodes, which are nothing but the target logical topologies. But to come up with the best logical topology is of main concern. The problem of choosing the best one is solved by considering the congestion factor in the logical topology. The logical topology with minimal or optimal congestion is considered.

	A	B	C	D	E	F	G	H
A	0	8	6	3	0	1	2	8
B	2	0	9	3	5	4	7	6
C	1	5	0	6	9	8	4	1
D	2	5	8	6	3	9	7	4
E	0	2	5	4	7	8	6	3
F	2	0	1	4	3	0	8	2
G	0	2	8	6	8	6	3	4
H	8	2	0	1	0	3	6	5

Table 1-1 Traffic Matrix

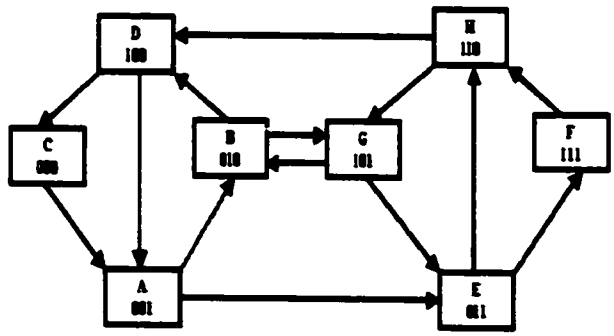


Figure1-2. Mapping in a De Brujin Graph for 8-node network (2^3)

The Genetic Algorithm introduced in this thesis works by creating many random solutions to the problem of mapping of nodes of physical topology on to the nodes of logical topology. This population of many solutions will then be subjected to an imitation of the evolution of species. All of these solutions are coded as a list of numbers representing the nodes, these solutions are also called as chromosomes.

Example 1-1

Let C_1 and C_2 be the two logical topologies (8-node De brujin graph) generated by the genetic algorithm for a given physical topology.

Chromosome C_1:

Logical	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
Physical	F	C	A	G	B	D	H	E

Chromosome C_2:

Logical	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
Physical	A	B	E	D	H	G	F	C

To find out a good mapping of physical nodes onto the nodes of the logical topology, we need to know the fitness of the two chromosomes. The chromosome with the highest fitness function will be considered among the two. In GA, an objective function is used to calculate the objective value of a chromosome. In this thesis, the chromosome represent a mapping between the physical and logical nodes and objective value is the congestion for that mapping. The value of congestion is dependent on how the communications are routed over topology. Here the chromosome C_2 is having a fitness value of 37 when compared to 24 for C_1.So C_2 will be considered as the better mapping.

1.4 Thesis Organization

In chapter 2, a detailed review of Optical Network Concepts are given to have an overview of the concepts used in wavelength routed multi-hop optical networks and in Chapter 3, we provide a literature review of Genetic Algorithms and its application in optical network design. Chapter 4 presents a new approach to design the optical networks using Genetic Algorithmic approach. Various crossover strategies are discussed and new crossovers are introduced for the proposed GA. The experimental results are presented and analyzed in Chapter 5.In Chapter 6 the contributions of this thesis and future work are suggested and concludes with a critical summary.

Chapter 2

Optical Networking Concepts

Most of the recent advents of new technologies in telecommunications and Internet industries are made possible by the increase in bandwidth due to the introduction of optical fibers. It is widely believed that all-optical networking using wavelength division multiplexing ought to become eventually the next networking generation after ATM (Asynchronous Transfer Mode). These new technologies had made new path for the recent exciting researches in the field of telecommunications and WWW.

Wavelength-division-multiplexed (WDM) optical networks [Green92] using wavelength routing are considered to be potential candidates for the next generation of wide-area backbone networks. A WDM all-optical network can use the large bandwidth available in optical fiber to realize many channels, each at different optical wavelength, and each of these channels can be operated at moderate bit rates. Networks using 20-100 wavelengths will be feasible in the next few years [GGR93] .Use of DWDM allows providers to offer services such as e-mail, video, and multimedia carried as Internet protocol (IP) data over asynchronous transfer mode (ATM) and voice carried over SONET/SDH. Despite the fact that these formats—IP, ATM, and SONET/SDH—provide unique bandwidth management capabilities, all three can be transported over the optical layer using DWDM. This unifying capability allows the service provider the flexibility to respond to customer demands over one network [SR91].

The logical topology design problem based on WDM technology in optical networks has been studied extensively in the past decade and there is ongoing research in this area. Here in this chapter a brief introduction of WDM technology will be given and also about the single and multi hop Optical networks. The new approach of using a regular graph, the De Brujin graph, as the logical topology is going to be discussed.

2.1 Wavelength Division Multiplexing

Wavelength division multiplexing (WDM) divides the huge bandwidth of a fiber into many non-overlapping wavelengths. These wavelengths are the WDM channels. Each channel can be operated asynchronously and in parallel at any desirable speed. It is thus the optical analog of the Frequency Division Multiplexing in the radio communication world where every station transmits at a different frequency and receivers that want to tune in to a particular channel must tune in to the desired frequency. The main advantage of WDM is that is can reuse each channel, the relative positions of each nodes can be changed dynamically based on the traffic fluctuations [GGR93].

In fig 2.1 wavelengths there distinct wavelengths λ_1, λ_2 and λ_3 are combined at multiplexer and carried over on a single fiber where optical amplifier has been used to maintain the purity of the signal and at the demultiplexer end those distinct wavelengths are retrieved.

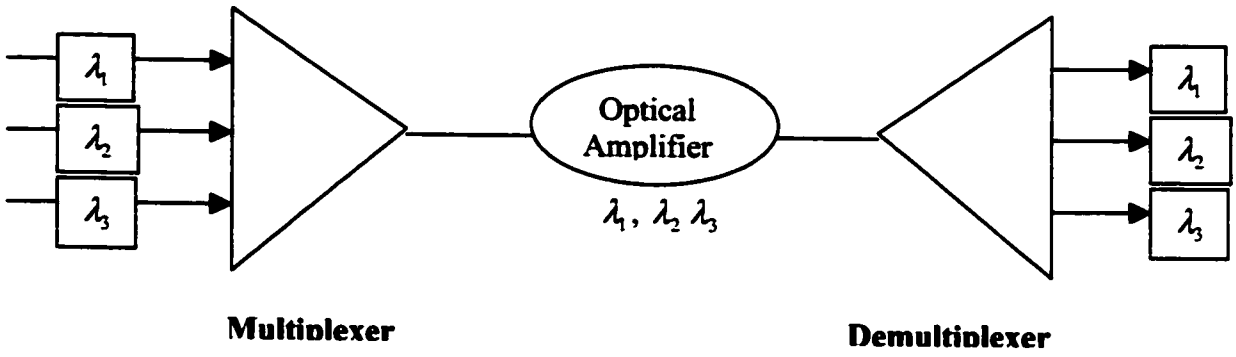


Fig 2.1: Wavelength Division Multiplexing (WDM)

2.2 Classification of WDM networks

A distinction must be made in classifying networks between the physical architecture of the system and its logical topology. The former is the actual physical structure of the network (i.e. the fibers that link together the nodes), whereas the latter describes the way in which data is transferred between the nodes on the network. The physical architecture and logical topology are often the same for a given network, but there are many instances where they differ.

Three basic architectures have been used, the bus, the ring and the star. The passive star has proved to be the most popular to date while the ring has the advantage of superior resilience, particularly against fiber breaks.

2.2.1 Single hop and Multihop networks

Multi-wavelengths networks can be classified on the basis of number of hops to reach from source to destination nodes. In a single-hop network, once the data stream has been transmitted as light, it continues without conversion into electronic form until it reaches its destination. For a packet transmission to occur, a transmitter at the source must be tuned to the same wavelength as a receiver at the destination for the duration of the packet transmission time. There are two main challenges to overcome in the implementation of single-hop WDM networks: the need for fast wavelength tuning in the transmitters and/or receivers and the development of efficient protocols to allocate wavelength channels for different connections within the network.

A method of avoiding the above problems is to use a multi-hop network. In this case, each node has access to only a small number of the wavelength channels used in the network. Fixed-wavelength transmitters and receivers are used for this purpose with the minimum requirement being a single transmitter and a single receiver tuned to different wavelengths [Acam87]. If a node wishes to transmit to another node whose receiver is tuned to a different set of wavelengths than the source transmitter, the latter will transmit to an intermediate node which has a receiver of the same channel as the source's transmitter and a transmitter with the same tuning as the destination's receiver. More than one intermediate node may thus be needed for a packet to reach its destination. At each intermediate node, the data packet is switched electronically to the next appropriate node and then retransmitted on a new wavelength carrier.

Design of multihop logical optical network architecture should meet the following important requirements [BS99]:

- Average internodal distance should be small as it is inversely proportional to the network utilization.
- Each node in the network should employ only a small number of transceivers
- Physical embedding of the logical network topology should require only a small set of distinct wavelengths.
- Adding or deleting nodes should have a minimal impact on network configuration and performance.
- Logical topology should be able to tolerate maximum node and link failures.

2.2.2 Wavelength Routed Optical Network (WRON)

Wavelength routing means that optical signals can be selectively routed, in the optical domain, based on their wavelengths. These types of networks are referred to as Wavelength Routed Optical Network (WRON). WRON is the combination of single hop and multi hop network and they show the characteristics of both. A wavelength Routed Switch is used at intermediate router nodes. Optical Networks using wavelength routing are considered to be the backbone for high-speed future networks. Figure 2.3 shows an optical network with 4 nodes. There are three light paths established over the physical network $A \rightarrow C$, $C \rightarrow A$ and $B \rightarrow A$. Communication from node A to node C is an example of single –hop communication since communication is done by one single light path $A \rightarrow C$. Communication from node B to node C is an example of multi-hop

communication since this communication is performed along the path $B \rightarrow A \rightarrow C$ which uses two light paths $B \rightarrow A$ and $A \rightarrow C$.

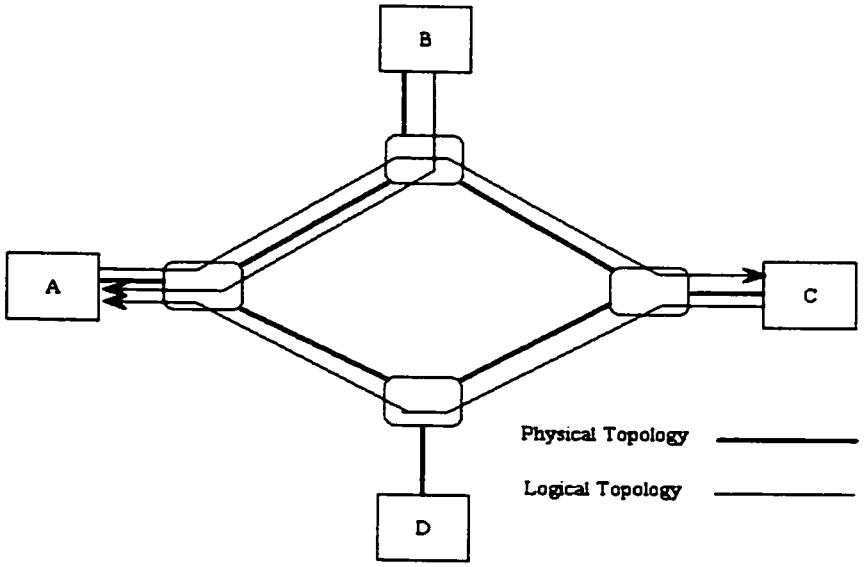


Figure 2.3 Wavelength Routed Optical Network (WRON)

2.3 Regular Logical Networks Topologies

In this section various regular multihop logical topologies for light wave networks will be discussed. First and fore most simple regular topologies: bus, ring, and star will be discussed. Shuffle-exchange based topologies -- ShuffleNet, de Bruijn graph, and GemNet are then considered.

2.3.1 Bus Topology:

In a logical bus topology, each node is equipped with two fixed-tuned transmitters and two fixed-tuned receivers. A transmitter of node i is connected to a receiver of node $(i + 1)$, and the other transmitter of node i is connected to the receiver of node $(i \ 1)$, except for the first node and the last node where only one pair of transceivers is used.

Average hop distance in an *N*-node bi-directional bus network is *N*/3 nodes. In a bus topology, if the unused transceivers of the end nodes are connected to the unused transceivers of the first node then a logical ring topology is obtained. The average hop distance in a bi-directional ring topology is *N*/4 nodes. Also, in a ring topology there are two link disjoint paths between any source and destination nodes, whereas in a bus topology this path is unique. Ring topologies are attractive for their simple interfaces and control. A unidirectional ring is minimal in the sense that it uses a minimum number of links to achieve full connectivity.

2.3.2 Star Topology

In a logical star topology, all the nodes transmit and receive from a central node, which can be an optical switch. Hop distance between any two nodes in a star topology is always two and the path between any two nodes is also unique. It is an important to note that the bus, ring, and star topologies are commonly found as physical topologies rather than logical ones.

2.3.3 Shuffle Net:

ShuffleNet topology was proposed by Stone [Stone71]. Acompora first proposed the Shuffle net architecture as a virtual optical network topology in 1987 [20]. A (p,k) ShuffleNet can be constructed out of $N=kp^k$ $(k = 1,2,3...)$ nodes which are arranged in k columns of p^k nodes each, with K^{th} *column* connected to the first .The connectivity between the successive columns is a p-shuffle is a generalization of the ($p = 2$, $k = 2$) perfect shuffle.

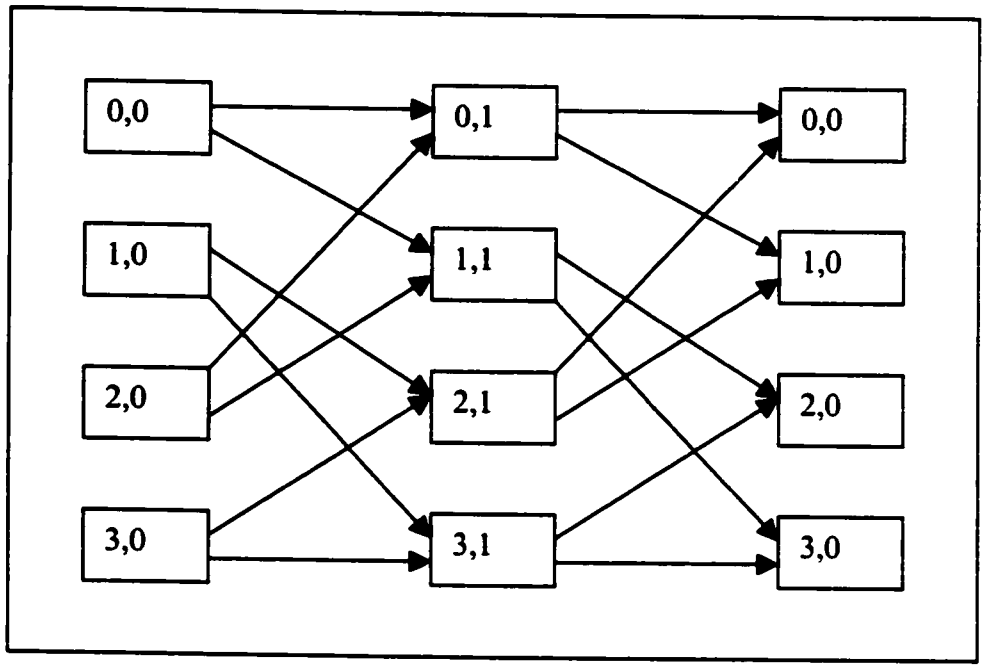


Figure 2.4 Shuffle Net

2.3.4 Gem Net:

GEMNet is a recently proposed class of network architecture that includes Shuffle Net Fig 2.7 and de Brujin graph as its special members [11,12]. Modularity of Gem Net is one and it generally has comparable or better properties than those of Shuffle Net or de Brujin graph. GEMNet has a simple routing scheme and low diameter, O(*log N*), where *N* is the number of nodes in the network. GEMNet can be defined for any arbitrary value of *N*.

In GEMNet, unlike in Shuffle Net, deBrujin, or Shuffle Net, the number of nodes, *M* in a column is not restricted to be of the form p^k. Also, for a given *N*, there exists, as many different GEMNet configurations as there are distinct ways of factoring *N* in to two ordered integers. GEMNet also reduces to a de Brujin graph of diameter D when $M = p^d$ and $k = 1$.There fore, GEMNet which is based on a generalized shuffle exchange

connectivity pattern, has much more flexible structure than that of Shuffle Net or de Brujin graph because the number of nodes is not restricted to any form

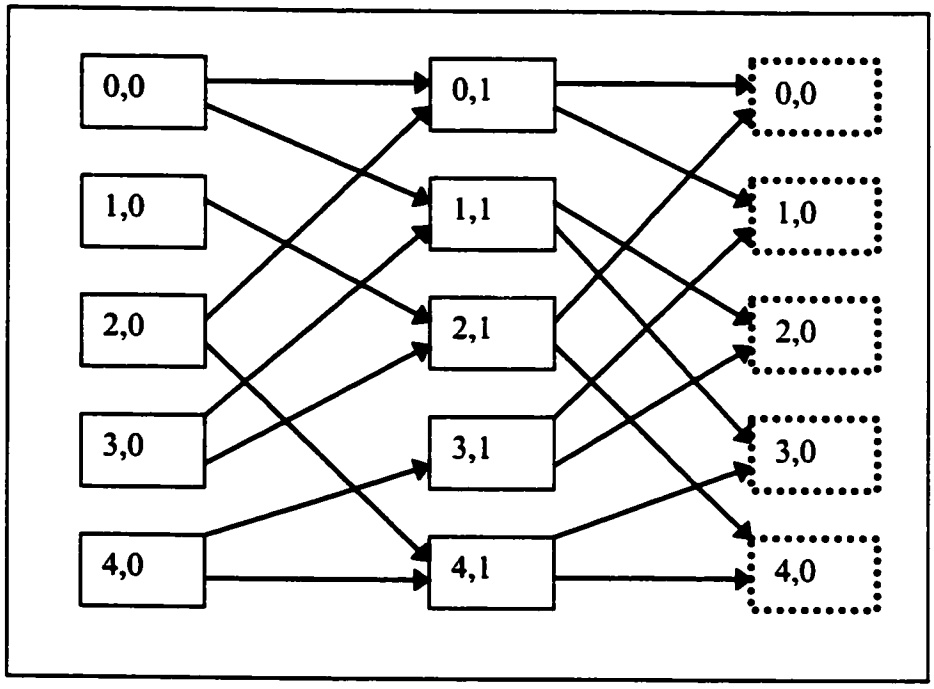


Figure 2.4 GEMNET

2.4 Advantages of Using Regular Topologies for Minimizing Congestion

Regular topologies such as GEMNET have symmetrical structure and thus results in small average hop distance in such topologies. For GEMNET architecture of *n* node the average hop distance is *O(log(n))*, which results in shorter paths for routing packets. Maximum offered load on a link is called Congestion. The smaller he value of congestion, the better the performance of the network. And it is shown that the de Brujin Graph's attractive low diameter property minimizes the congestion on logical topology.

Chapter 3

Genetic Algorithms Concepts

Genetic Algorithms (GA) are adaptive heuristic search algorithms premised on the evolutionary ideas of natural selection and genetic. The basic concept of GA's is designed to simulate processes in natural system necessary for evolution, specifically those that follow the principles first laid down by Charles Darwin of survival of the fittest [GOLD89]. They represent an intelligent exploitation of a random search within a defined search space to solve a problem.

Genetic Algorithms, first pioneered by John Holland, has been widely studied, experimented and applied in many fields in engineering worlds. GA's provide an alternative method to solving problem and consistently outperforms other traditional methods in most of the problems. Many of the real world problems involved finding optimal parameters, which might prove difficult for traditional methods but ideal for GAs. However, because of its outstanding performance in optimisation, GAs have been wrongly regarded as a function optimiser. In fact, there are many ways to view genetic algorithms. Perhaps most users come to GAs looking for a problem solver, but this is a restrictive view [DJONG93].

Genetic algorithm is a highly parallel mathematical algorithm that transforms a set or population of individual mathematical objects. In general a fixed length character strings patterned after chromosome strings, each with an associated fitness value, into a new

population (i.e., the next generation) using operations patterned after the Darwinian principle of reproduction and survival of the fittest and after naturally occurring genetic operations [KOZA 92].

3.1 Search Techniques

Genetic algorithms are a type of optimisation search techniques. Search techniques in general, as illustrated in fig1 can be grouped in to three broad classes [GOLD89] Calculus, enumerative and random search methods, which can be further subdivided. Calculus based methods include direct and indirect. Indirect is the search for the peaks of maxima by finding zero of the gradient. Direct techniques are those such as Newton's method. Random methods include simulated annealing, evolutionary strategies, genetic algorithms and the simple random walk through the search space. Enumerative methods are the brute force methods where all the solutions in the whole search space are generated.

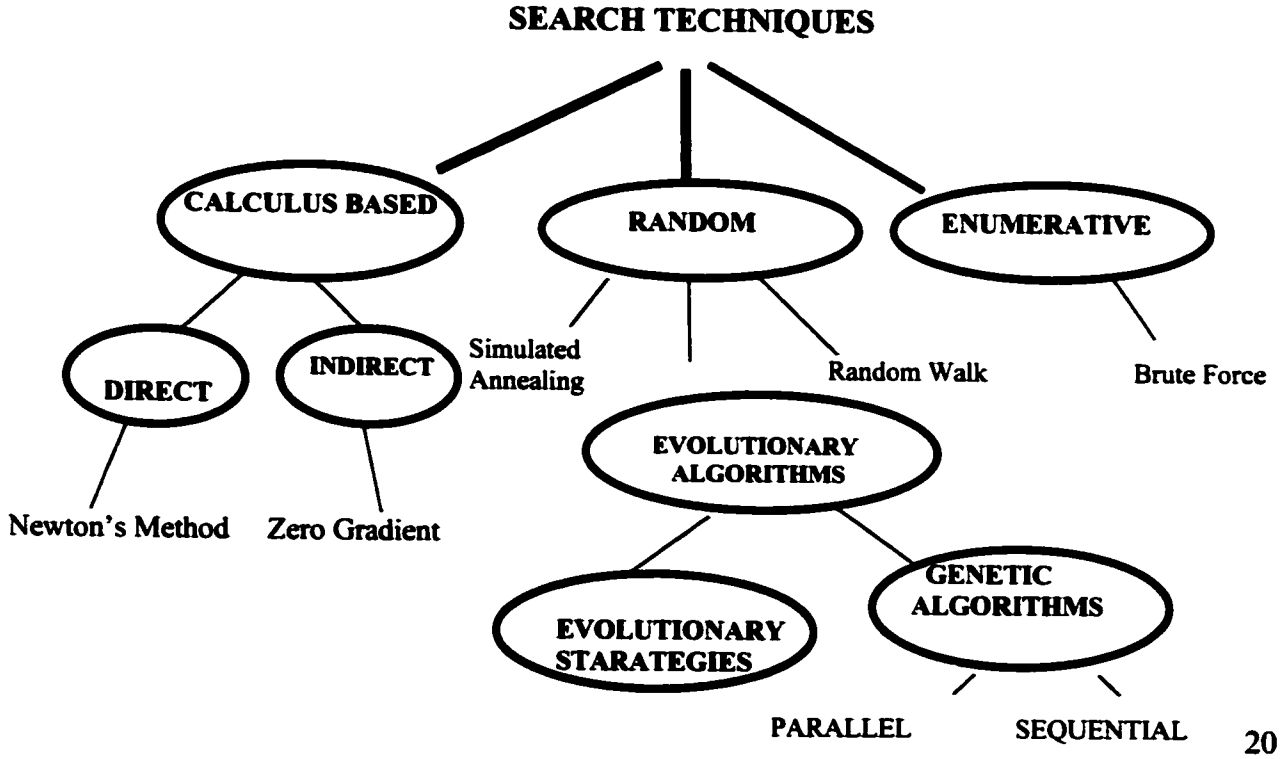


Figure 3.1. Search Techniques

3.1.1 Random Search

Random search is a brute force approach, usually used for difficult function where there are no other viable search methods. In this approach, points in the search space are selected randomly and their finesses are evaluated. This strategy is not very intelligent and is rarely used alone. It is very unfavourable, as there is no guidance to direct the search.

3.1.2 Gradient methods

There are number of methods for optimizing continuous functions, all of which are based on using information about the gradient of the function to guide the direction of search. A drawback is that these methods will only work on problems where the derivative of the function is continuous. These methods are generally referred to as hill climbing methods since they usually move upwards along a gradient in the hopes of finding maxima. On functions that are multi-modal, where there are many peaks these gradient techniques can be trapped in a peak that is perhaps the highest in the local vicinity but not the highest globally. Once they have reached the top of a local maximum no further progress can be made towards a solution.

3.1.3 Iterated Search:

It is a combination of both random and gradient search, which results in hill climbing search. In this method, the gradient method is used to find a peak. But once a peak has been located the hill climbing is stated over again, this time using a different randomly chosen point This gives the technique the advantage of simplicity and gives it ability to perform well even if the function has only a few local maxima.

This method is nothing more than several iterations of the gradient method and it cannot find an overall picture of the shape of the domain. Therefore, as the search progresses it cannot use any previously obtained information to choose a likely starting point for the next hill climb. These forces to evaluate points in regions of low fitness, just as often as in regions found to be of high fitness. A genetic algorithm, by comparison, starts with an initial randomized population but allocates increasing trials to regions of the search space found to have higher fitness.

3.1.4 Simulated Annealing

Simulated annealing is a process that is analogous to the cooling of a metal from a molten state .In this method, an initial search point is chosen at random. A move is made to another point in the search space also chosen at random. If the move is to a point with higher fitness then it is accepted. How ever, if the move is to a point with lower fitness its acceptance is determined by a portability function, which varies over time. This function begins with a value near one but gradually reduces towards zero [GOLD89].

In this way, the search can proceed up and down in the beginning but as the search starts to cool the amount of downward mobility is reduced. Towards the end of the search, only upward moves are allowed. Downward moves are essential if local maxima are to be escaped but after a period of time it is hoped that the search will be near the peak of the global maximum and the cooling temperature will force the search to proceed upwards towards this maximum. This technique only deals with one solution at a time and does not build up an overall picture of the search space. Information from previous moves is not used in the selection of the new moves. This is in direct contrast to a genetic

algorithm where the search space peaks are embedded in the chromosomes of the population individuals and tend to be passed forward during the search.

3.2 Genetic Algorithms vs. Traditional Methods

There are various traditional methods for search strategies, like Calculus-based Search, Dynamic Programming, Random Search, Gradient Methods, Iterated Hill climbing and Simulated Annealing. But Genetic Algorithms differ from the tradition search methods in different ways.

Genetic algorithms manipulate decision or control variable representations at a string level to exploit similarities among high-performance strings. Other methods usually deal with functions and their control variables directly. GA's deal with parameters of finite length, which are coded using a finite alphabet, rather than directly manipulating the parameters themselves. This means that the search is unconstrained neither by the continuity of the function under investigation, nor the existence of a derivative function. Moreover, by exploring similarities in coding, GA's can deal effectively with a broader class of functions than can many other procedures. Search from a population, not a single point.

Similarly, GA's finds safety in numbers, by maintaining a population of well-adapted sample points; the probability of reaching a false peak is reduced. The search starts from a population of many points, rather than starting from just one point. This parallelism means that the search will not become trapped on a local maxima - especially if a measure of diversity-maintenance is incorporated into the algorithm, for then, one candidate may become trapped on a local maxima, but the need

to maintain diversity in the search population means that other candidates will therefore avoid that particular area of the search space.

Evaluation of the performance of candidate solutions is found using objective, payoff information. While this makes the search domain transparent to the algorithm and frees it from the constraint of having to use auxiliary or derivative information, it also means that there is an upper bound to its performance potential.

3.3 Simple genetic Algorithm

Genetic Algorithms are used for a number of different application areas. An example of this would be multidimensional optimization problems in which the character string of the chromosome can be used to encode the values for the different parameters being optimized.

A genetic algorithm or any evolution program for a particular problem must have the following five components [MICH94]:

- A genetic representation for potential solution to the problem.
- A way to create an initial population of potential solutions.
- An evaluation function that plays the role of the environment, rating solutions in terms of their ***fitness***.
- Genetic operators that alter the compositions of children,
- Values for various parameters that the genetic algorithm uses i.e., population size, probabilities of applying genetic operators.

3.4 Types of Genetic Algorithms

In general Genetic Algorithms can be classified in to three major categories.

- •Generational GA
- •Steady-State GA
- •Generation Gap and CHC

3.4.1 Generational GA

In classical Generational GA the whole population is replaced by the new. This contrasts with the steady-state GA where one member of the population is replaced at a time. GAs that extrapolates between generational and steady state are said to have a generation gap. To extrapolate between the generational and steady state GA, De Jong introduced the notation of a generation gap [DJONG75, 93] G denotes the fraction of the population that is changed at each generation. Thus for a generational genetic algorithm G=1 while for a steady-state genetic algorithm G=1/P. Eshelman introduced an evolutionary algorithm called the CHC Adaptive Search Algorithm, where a new population is generated and added to the old population before selection [ES93]. This is related to the evolutionary strategies. The genetic drift caused by different generation gaps and for CHC has been calculated in [ROJERS99]. The larger the generation gaps the small the genetic drift. CHC has half the genetic drift of a generational GA.

3.4.2 Steady-State GA

In a steady-state genetic algorithm one member of the population is changed at a time. To perform selection a member of the population is chosen according to its fitness. It is copied and the copy mutated. A second member of the population is selected which is then replaced by the mutated string. In crossover two members of the population are chosen, a single child created which replaces a member of the population. Any selection method can be used to select the individual for mutation or the parents.

There are a number of different replacement strategies

- •Replace worst (often gives too rapid convergence)
- •Replace a randomly chosen member
- •Select replacement using the negative fitness.

Steady-state GAs have been investigated in [ROGERS99] and [ROGERS00]. The major difference between steady -state and generational GAs is that for each P members of the population generated in the generational GA there are $2P$ selections. Consequently the selection strength and genetic drift for a steady-state GA is twice that of the generational GA. The steady-state GA, therefore, appears twice as fast although it can lose out in the long term because it does not explore the landscape as well as the generational GA. The effects of genetic drift in steady state GAs and GAs with an arbitrary generation gap are discussed in [ROGERS00].

A typical GA consists of the following steps

1.Generate initial population

2.Measure fitness

3.Select a mating pool

4.Mutate each member of the mating pool

5.Pair member of the mating pool and perform crossover to obtain a new generation

6.Return to Step 2 until some stopping condition is satisfied.

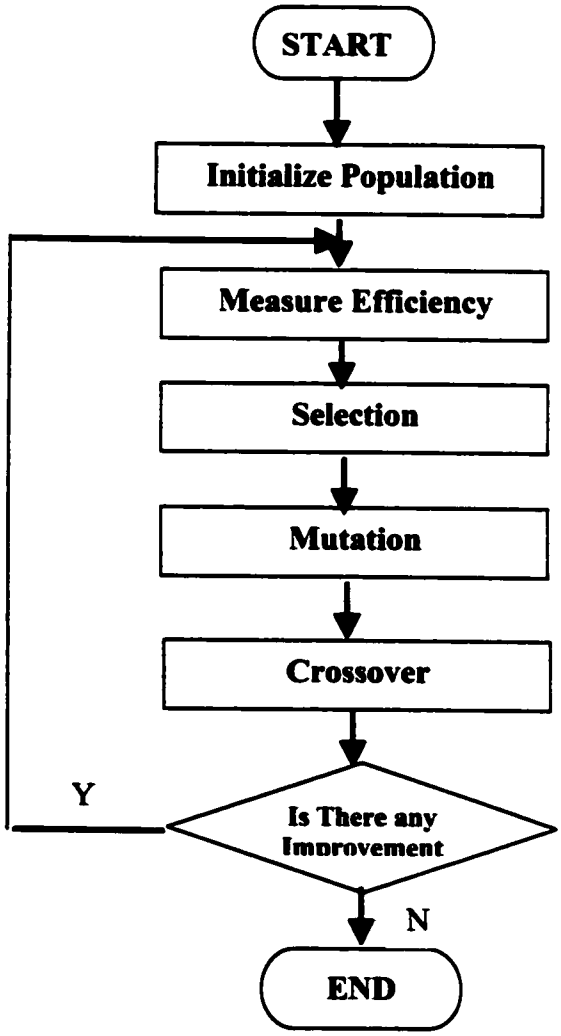


Figure 3.2 Typical Genetic Algorithm

3.5 Genetic Algorithm Mechanism

As mentioned earlier, Genetic Algorithms are biologically inspired search methods. The GA mechanism copies the necessary and essential operations of natural genetics. It operates on a large group of solutions known as a population and simulates the biological operations of survival, mating and mutation [HOLL92]. Similar to a biological cycle of birth, reproduction and death, a GA follows a similar cycle called the GA cycle. [GOLD89].

3.5.1 Solution Coding

Genetic algorithms use a vocabulary borrowed from natural genetics. Similar to a biological chromosome, a GA chromosome is a string of characters that represent the instructions for building an individual [MICH94]. Since in the GA realm and individual is a solution to a problem, a GA chromosome must contain the coded instructions for building a solution. That is the GA chromosome is the solution in a coded format. Chromosomes are made of units - genes arranged in linear succession. Every gene controls the inheritance of one or several characters. Genes of certain characters are located at certain places of the chromosome, which are called loci. Any character of individuals can manifest itself differently; the gene is said to be in several states, called alleles.

Each genotype or a chromosome represents a potential solution to a problem. An evaluation process run on population of chromosome corresponds to a search through a space of potential solutions. Such a search requires balancing two objectives, exploiting the best solutions and exploring the search space [BOOK82].

A GA performs a multi-directional search by maintaining a population of potential solutions and encourages information formation and exchange between these directions. The population undergoes a simulated evolution: at each generation the relatively GOOD solutions reproduce, while the relatively "bad" solutions die. To distinguish between different solutions an objective or evaluation function plays the role of an environment.

3.5.2 Representation

Finding a powerful representation of the problem is a vital part of making GA's work. A powerful representation of a problem allows simple operators to be used to explore the solution space effectively. That is it imposes a metric whereby neighboring solutions are likely to have similar fitnesses.

In a classic GA the problem is represented by a string of variables

$$S = \{ S_1, S_2, S_3 \ldots\ldots\ldots\ldots S_N \}$$

The mapping between the variables and the problem is essential if simple operators are going to be used to explore the search space. If phenotype centered operators are used the representation is irrelevant. String representations are not the only type of representation of the problem. Tree structures can also be used, this is common in genetic programming. In the traditional genetic algorithm's, genetic operators act on the genotype. That is, they blindly manipulate the string representing the problem. This is not necessary, operators can be considered acting in the phenotype space, i.e. the problem space. i.e., if we had a problem defined on a lattice then we could consider crossover operators, which divided the lattice into two contiguous pieces and swap the solution in either piece.

This would appear simple in the phenotype space but might be very complicated in the genotype space. For many simple test problems it is possible to choose a representation so that genotype space has a similar structure to phenotype space. When this isn't the case then one should consider using phenotype operators [ROGER00].

3.5.3 Fitness Function

A fitness function must be devised for each problem to be solved using genetic algorithms. For a given particular chromosome, the fitness function returns a single numerical ***fitness*** or ***figure of merit***, which is supposed to be proportional to the ***utility*** or ***ability*** of the individual, which that chromosome represents. Particularly for function optimizations, it is obvious what the fitness function should measure-it should just be the value of the function

3.5.4 Reproduction

In the reproduction phase of GA, individuals are selected from the population and recombined, producing offspring, which will comprise the next generation. Parents are selected randomly from the population using a scheme, which favors the more fit individuals. Good individuals will probably be selected several times in a generation, poor ones may not be at all. Having selected two parents their chromosome are recombined, typically using the mechanism of crossover and mutation. Some research is done in the field of multi parent recombination where more than two parents are considered for the reproduction in an evolutionary problem solver [EIBEN 97]. It was

proposed that there are three different types of multi-parent mechanisms with tunable arity.

3.5.4.1 Selection Methods in Genetic Algorithms

One of the most critical phases in a Genetic Algorithm is the choice of an appropriate selection method. Individuals for *recombination* are selected according to their fitness. Various methods have been suggested in literature [Bak85]. Basically, the various methods can be divided in *fitness scaling methods* or *ranking methods*.

The problem of *premature* or *rapid convergence* can be solved by Fitness scaling methods by scaling the "raw fitness" values, but still the selection probability is derived from the scaled fitness value. Whereas, ranking methods use the fitness values to produce a ranking list of the population and the probability of selection is derived from the rank. Recently, ranking methods have gained increasing popularity and are thought to be superior to fitness scaling methods [dlMT93].

The selection processes have to be optimized for the specific type of problem to be solved . A subset of the population was kept intact to the next generation and to reproduce. Too small subsets may lead to the loss of some promising genomes that initially do not have high fitness but may have good offspring. Too large subsets may lead to slow convergence. The choice of parents can be performed in several ways. You can simply have a pool of parents and randomly choose two parents (sampling without replacement). Another alternative is to assign a probability to be chosen to every parent in the pool. The probability corresponds to the fitness of the parent. Highly fit parents

have greater probability to reproduce. The second alternative probably leads to a higher convergence rate than the first.

To select the individuals to progress to the next population it is important to strike a balance between fitness and diversity. Frequently an individual will appear during the run of a GA's that has a fitness factor above the population average yet is only showing signs of a local optimum for the evaluation. These chromosomes are commonly referred to as super-individuals: chromosomes of high fitness but only for a local optimum. It is therefore important to guard against these super-individuals flooding the population causing premature convergence of the population to a local optimum. To guard against premature convergence it is important to use a selection method that promotes a diverse population. One that balances the need for super-individuals' genetic material to be spread throughout the population and the need to maintain a diverse population until a suitably optimum value has been located.

3.5.4.1.1 Tournament Selection

One of the most popular ranking methods is *Tournament Selection*, where a number of **n** individuals (typically **n = 2**) are chosen randomly from the population and the individual with the highest fitness is selected and placed in the *mating pool*. This procedure can be extended to probabilistic tournament selection borrowing an idea from *Simulated Annealing*. With a certain probability, which is decreasing with time the individual with a lower fitness is selected.

3.5.4.1.2 Roulette Wheel Selection

Roulette Wheel Selection is one of the most well known and more popular of the selection methods listed above. It is a fitness-based selection with each individual in a given population being assigned a wedge of a roulette wheel proportional to its fitness against the total fitness of the population. Subsequently the fitter individuals of a population tend to be selected more frequently being passed on to the next generation after crossover and mutation. Due to the random nature of each spin of the roulette wheel, stochastic errors can enter the selection process where individuals have a lesser or greater representation in the next population than the fitness proportion for the individual dictates.

3.6 Genetic Operators

In order for evolution to occur, there must be some genetic variation among the offspring. In natural life, this is insured by natural imperfections in the replication of the informational molecules. However, one way in which digital chemistry differs from organic chemistry is in the degree of perfection of its operations. In the computer, the genetic code can be reliably replicated without errors to such a degree that we must artificially introduce errors or other sources of genetic variation in order to induce evolution.

The main purpose of genetic operators is to cause chromosomes created during reproduction to differ from those of their parents. They must be able to create configurations of genes that were never existent before and that are likely to perform well. When genetic operators are used with reproduction plans, the result is a surprisingly

sophisticated set of adaptive plans. The two most commonly used genetic operators are mutation and crossover.

3.6.1Crossover Operators

One of the unique aspects of the work involving genetic algorithms is recombination. In most GA's, recombination is implemented by means of a crossover operator which works on pairs of individuals (or parents) to produce new offspring's by exchanging segments from the parent's genetic material. In general, the number of crossover points has been fixed at a very low constant value of 1 or 2. This is proved in the previous work of both theoretical and empirical nature [HOLL75]. But in recent times there were indications that there are situations in which having a higher number of crossover points is beneficial [SYS89].

3.6.1.1 Crossover Techniques

The simple GA performs crossover by making a single cut at the same location in each of the two parent chromosomes. This cut occurs somewhere between the first gene and the last gene. The cut sections are then exchanged to form two offspring. This method is very simplistic and tends to destroy building blocks that contain widely spaced genes.

Due to this reason many researchers have devised many new crossover techniques often using more than one cut point. The effectiveness of multiple-point crossover was first investigated by DeJong [Djong75] and he found that while 2-point crossover gives an improvement in the performance of GA. With the increase in addition of crossover

points, the search becomes more random because building blocks are more likely to be disrupted. The problem space is thoroughly searched at the expense of highly increased search time.

3.6.1.1.1 2-Point Crossover

Chromosomes are regarded as loops formed by joining the ends together .Two identical points are randomly selected along the length of two parent strings, dividing each into three segments. The first and third segments from the first parent and the second segment from the second parent are recombined to produce the offspring strings. 2-point crossover can be generalized to n-point crossover, where n is the number of crossover points. Researchers generally agree that 2-point crossover is better than 1-point because it samples uniformly across the full length of a chromosome [SDJ91b].

Parent 1	1101	1001	0110
Parent 2	0001	0110	1111
Child 1	1101	0110	0110
Child 2	0001	1001	1111

Figure 3.3. Multipoint –Crossover

3.6.1.1.2 Uniform Crossover

Another form of crossover is the ***n***-point uniform crossover where the number of points ***n*** varies dynamically with each mating [SPEARS91]. In this method a probability P_o characterizes the degree of disruption. A random number is drawn for each parameter

along the string. If the value of this random number is greater than P_o, then the value from parent 1 is used to produce the offspring string. If the random number is less than P_o, parent 2's value is used. The greatest disruption occurs when $P_o = 0.5$.The process is repeated with the parents exchanged to produce the second offspring .A new crossover mask is randomly generated for each pair of parents. Offspring is therefore; contain a mixture of genes from each parent. The number of effective crossing points, while not fixed, will average *l/2* (where *l* is the average length of chromosome).

Parent 1	1101 1001 0110
Parent 2	**0001 0110 1111**
Template	1101 0010 0110
Child 1	1101 **0100** 1111
Child 2	**0001** 1011 **0110**

Figure 3.4 Uniform –crossover

3.6.1.1.3 Crossover Comparisons

There is many debates over which is the best crossover method to use. It was shown that the uniform crossover causes fewer disruptions in long defining length schemas [SYS89]. Syswerda shows that the overall amount of schemata is lower, there by better preserving precious blocks.

Under 2-point crossover the defining length, and not the order, of the schemata determines the likelihood of its disruption. While under uniform crossover, it is based on the order and not its length. Which means that the ordering of genes within a

chromosome is completely irrelevant and eliminates the need for many of the re-ordering operators such as inversion.

An extensive comparison of 1-point, 2-point, multi-point and uniform crossover operators was performed by Eshelman et al [ES93]. Theoretical analysis was performed in terms of positional and distributional bias on several problems. While they found that an 8-point crossover was good on the problems they are tried there was only about a 20% difference in speed between the slowest and fastest techniques. But the theoretical analyses by Spears & DeJong [SPEARS91] shows that both 1 and 2-point crossovers are optimal. They state that due to reduced productivity; 2-point crossover will perform poorly when the population has largely converged.

If two similar chromosomes undergo 2-point crossover then the exchanged segments are likely to be identical causing the offspring to be identical to their parents. But under uniform crossover, this is less likely to happen. A new 2-point crossover is described in [SPEARS91], which will check the offspring's, and if they found to be identical to their parents then the crossover is repeated using two new crossover points. This crossover was found to be slightly better than uniform crossover. But this crossover is best only when there is a large population, and that for small population uniform crossover is best due to the increased disruption that it causes [DJONG90]. Syswerda formulated a new crossover called simulated crossover [SYS92]. Simulated crossover technique treats population of GA as a conditional variable to a probability density function that predicts the chance of generating samples in the problem space. The difference is that by using Simulated crossover technique is that only one child is generated. In creating the child, BSC (Bit simulated crossover) ignores the bit values of

individuals, but uses the statistics of bit values of entire population. But it is found that Simulated crossover's rate of production of schemata is slower than that of explicit crossover and also as the recombination rate is slower for schemata with low selection probability compared to a schemata with high selection probability. But as these differences are small, this crossover can be used as a recombination operator.

3.6.2 Mutation

Mutation occurs with low probability and functions as a background operator. [BOOK82][DJONG85]. It is included to allow for the searching space that may otherwise be precluded by the converging chromosomes as genetic information is discarded during crossover. The exact amount of mutation necessary is somewhat open to debate. Too little and useful alleles that are not currently in the population can never be found while too much causes the GA to degenerate into a random search.

Although crossover is generally seen as the major mechanism for exploring new search space, there are examples in nature where creatures using asexual reproduction have evolved. When Schaffer et al [SCHA89] did a study to determine the optimum parameters for Gas; it was found that mutation played a larger role than previously thought.

In a study by Spears [SPEARS93], crossover and mutation were compared and it was found that each operator contained characteristics not found in the other but that each is simply a form of a more general exploration operator that modifies alleles based on available information. As the population converges, Davis [DAVIS91] found that mutation plays as increasingly important role while the role of crossover diminishes.

Although its probability of use is small and it is seen as nothing more than a background operator, mutation plays a very important part in a GA solution. Adjusting for the optimum GA parameters is difficult since changes in mutation rate will affect performance much more than changes to the crossover parameters [SCHA89].

3.7 Applications of Genetic Algorithms

Genetic algorithms have proved very useful for optimizing highly complex cost functions with large numbers of parameters. In many situations, GA's are able to quickly solve a problem where traditional methods of optimization have little or no success. Genetic Algorithms are increasingly finding application in diverse fields. As they are robust efficient optimization techniques for complex multi-dimensional problems, most of these problems can be characterized as NP-hard and are generally intractable using simple algorithms. Examples of such applications are Task scheduling, Designing VLSI Circuits, Simulations of Biological Evolution, Adaptive Control Systems and Optimization of Network Topologies. GA's can be used to find optimal ways of scheduling a number of tasks under the presence of constraints in such a way that the time taken to perform the tasks is minimized. It is also used in solving routing issues in data and computer networks - minimizing path lengths etc.

One of the classical examples of GA's application is the solving of TSP problem. This is one of the most known problems, and is often called as a *NP-Hard* problem. A salesman must visit n cities, passing through each city only once, beginning from one of

them, which are considered as his base, and returning to it. The cost of the transportation among the cities (whichever combination possible) is given. The program of the journey is requested, that is the order of visiting the cities in such a way that the cost is the minimum.

Let's number the cities from 1 to *n*, and Let City 1 be the city-base of the salesman. Also let's assume that c (i, j) is the visiting cost from i to j. There can be c (i, j)<>c (j, i). Apparently all the possible solutions are *(n-1)!*.Someone could probably determine them systematically, find the cost for each and every one of these solutions and finally keep the one with the minimum cost. These requires at least *(n-1)!* Steps.

3.7.1 GA in optimization and planning: Traveling Salesman Problem

The TSP is interesting not only from a theoretical point of view; many practical applications can be modeled as a travelling salesman problem or as variants of it. For example, pen movement of a plotter, drilling of printed circuit boards, real-world routing of school buses, airlines, delivery trucks and postal carriers. Researchers have tracked TSP to study bio-molecular pathways, to route a computer networks' parallel processing, to advance cryptography, to determine the order of thousands of exposures needed in X-ray crystallography.

In the last two decades an enormous progress has been made with respect to solving travelling salesman problems to optimality, which, of course, is the ultimate goal of every researcher. This progress is only party due to the increasing hardware power of

computers. Above all, it was made possible by the development of mathematical theory and of efficient algorithms

There are strong relations between the constraints of the problem, the representation adopted and the genetic operators that can be used with it. The goal of traveling Salesman Problem is to devise a travel plan or a tour, which minimizes the total distance traveled. TSP is NP-hard it is generally believed cannot be solved in time polynomial. When GA' s applied to very large problems, they fail in two aspects. They scale rather poorly (in terms of time complexity) as the number of cities increases and also the solution quality degrades rapidly. Building a genetic algorithm to solve the TSP requires specifying those elements described in the GA definition.

Chapter 4

New approach for mapping problem

The main objective of this thesis is to investigate the genetic algorithms for designing the logical topology of multihop optical networks. The target logical topology is a De Brujin graph because of its attractive properties such as low diameter, rich connection and simple routing scheme. The logical topology to be designed is represented by a directed graph $G_l(N,A_l)$ where N is the number of nodes and A_l is the set of light paths established over the physical network. Traffic matrix $T= (t_{sd})$ where t_{sd} represents the arrival rate of packets at node s destined for node d. A link $(i,j) \in A_l$, if there is a logical link between nodes i and j.

Since the target logical topology is already known, the topology design is reduced to find an appropriate mapping between the nodes of the physical network and the nodes in the logical topology. In this chapter a new genetic algorithm for the logical topology design problem will be introduced and various control parameters (chromosome representation, the objective function and the stop criteria) that effective the performance of this algorithms will be discussed.

The basic structure of the algorithm follow that of simple genetic algorithm SGA [Goldberg89] and is given in Figure 4-1.In the algorithm P(0) represents the initial population and P(i) is the population in the i^{th} generation.

```
Initialization
        Setup all control parameters
        Create first population, P(0);
Evaluate P(0)
i:= 0;
Repeat
        i:= i+1
        While P(i) is not full
        {
                Select chromosomes from P(i-1)
                Apply crossover and mutation;

        }
```

Figure 4-1 Proposed Genetic Algorithm

4.1 Chromosome Representation

Chromosome representation is a key issue in Genetic Algorithms (GAs). An appropriate representation that stores problem specific information can enhance the performance of the algorithm and expand the search of the problem space. In the proposed algorithm, each possible solution (chromosome) is a mapping between the physical nodes and the logical nodes. Each solution represents a uniquely ordered list of physical nodes, where the positions in the list represents logical node numbers and the corresponding values are the physical nodes, to which these logical nodes are mapped.

Example: -

For a 8-node De Brujin graph (d = 2, k = 3) a chromosome can be represented as follows

Logical Node:	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Label	**(000)**	**(001)**	**(010)**	**(011)**	**(100)**	**(101)**	**(110)**	**(111)**
Physical Node	**C**	**E**	**D**	**H**	**B**	**A**	**G**	**F**

The top row gives the logical node number and the associated label and the second row shows the physical node to which each logical node is mapped. In general numbers are used to represent logical nodes and alphabets to represent the physical nodes.

The logical node to which the physical node is mapped is simply determined by its position of the physical node.

4.2 Initialization

All the control parameters are setup during the initialization phase, including the initial population and population size, the possibilities of mutation and crossover, the crossover strategies to be used. The population size is varied during the experiments depending on the size of the network and the performance of the genetic algorithm.

4.2.1Genetic Parameters

4.2.1.1 Selection

For selection process, roulette selection method is used to select the individuals that are participating in the crossover method with respect to the probability distribution based on fitness values.

4.2.1.2 Crossover:

In this thesis logical topology is implement as a de Brujin graph [SSb88], and in a de Brujin graph of degree *d*, each node has up to *d* predecessors and d successors, i.e., up to 2d nodes adjacent to it. The main aim of this crossover is to preserve *p* adjacent nodes of a given node *n*, where $1 \leq p \leq 2d$. The number adjacent nodes to be preserved are chosen by randomly picking a number from 1 to 2d.But some of the chosen node may contains a cycle, ie, A 8 node logical topology can be represented as 2^3 deBrujin graph and if the chosen node is 000 or 111, then those nodes can have only *d -1* predecessors and *d -1* successors as one of the successors and predecessors are 000 and 111 respectively. So in that scenario *p* should be in the range $1 \leq p \leq 2d-2$.

There by the proposed crossover protects the entire cluster and the unfilled positions in the first chromosome can be filled with genes from the second chromosome.

C_1:

Logical Node:	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Label	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
Physical Node	**C**	**E**	**D**	**H**	**B**	**A**	**G**	**F**

Table: 4-2.1 Chromosome C_1

C_2:

Logical Node:	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Label	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
Physical Node	**C**	**D**	**A**	**E**	**F**	**G**	**H**	**B**

Table: 4-2.2 Chromosome C_2

As an example, let C_1 **and** C_2 be the chromosomes for crossover. The proposed crossover creates the new offspring as follows.

1. Randomly pick a node as the root of the sub-graph. Assume N (001) is chosen.
2. Randomly pick a number *p* from 1 to 2d(i.e, from 1 to 4 as d = 2 in this case). Assume 4 is chosen and p = 3.
3. Node N has two predecessors, 000 and 100, and two successors 010 and 011.Our proposed crossover first protects the predecessors and then successors. There fore the root node (001) and its 3 adjacent nodes including two predecessors (000 and 100) and two successors (010) are filled in the Offspring^1:

Offspring^1:

Logical Node:	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Label	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
Physical Node	**C**	**E**	**D**	*	**B**	*	*	*

4. Fill the vacant positions, marked by "*", with genes from C_2. After getting rid of those nodes(**C, E, D, B**) already in Offspring1, (**A, F, G, H**) are left. Fill the "*" marked places in Offspring1 with (**A, F, G**) as it appears in C_2. Thus, an offspring is created:

Offspring^1:

Logical Node:	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Label	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
Physical Node	**C**	**E**	**D**	**A**	**B**	**F**	**G**	**H**

Similarly applying the same crossover to C_2 and C_1, **Offspring2** can be created

4.2.1.3 Mutation

The mutation operator in this algorithm is a simple swap operator, where two genes in a chromosome are randomly picked and switch their values. This can result in entirely new gene values being added to the gene pool. With these new gene values, the genetic algorithm may be able to arrive at better solution than was previously possible. Mutation is an important part of the genetic search as it helps to prevent the population from stagnating at any local optima. Mutation occurs during evolution according to a user-definable mutation probability. This probability should usually be set fairly low (0.01 is a good first choice). If it is set to high, the search will turn into a primitive random search.

4.3 Objective Function and Fitness Value

In genetic algorithms, an objective function is used to evaluate the fitness or to test the ability of chromosomes to satisfy certain conditions. Here a chromosome represents a mapping between the physical and logical nodes and the objective value is the congestion for that mapping. In this thesis two objective functions are used to evaluate the fitness of the solutions/chromosomes and the main objective is to maximize these objective functions.

1. $\sum T_{ij}(K - l_{ij})$

2. $\sum b_{ij} T_{ij}$

4.3.1 First Objective Function $\sum T_{ij}(K-l_{ij})$:

The first function $\sum T_{ij}\ (K-l_{ij})$ is used to minimize the number of hops to reach a packet from source node to a destination node.

Here T_{ij} Represents the traffic between node ***i*** and ***j***.

K : Represents maximum number of hops needed from any source to destination

l_{ij} : Represents number of hops needed from source node ***i*** to destination node ***j***

To calculate the number of hops to reach from a source node to a destination node the following shortest path algorithm is used. The main to implement this algorithm is to find the minimum number of hops required for a packet to reach from a source node to destination node.

4.3.1.1 Shortest Path Algorithm

Find all successors of u:

$S(u) = \{ w/w=u_1 \ldots u_{k-1}\ i,\ i\ (0 \ldots d-1)\}$.

Calculate the shortest path:

- Calculate the longest matching prefix of v and suffix of u,

  $$u_{k-t-1}\ u_{k-t} .. u_{k-1} = v0\ v_1 \ldots v_{t-1},\ t \in (0 \ldots k-2);$$

- Repeatedly left shift $v_t\ v_{t+1} \ldots v_{k-1}$ to u to create the shortest path
- $S(u) = S(u) - \{u_1 \ldots u_{k-1}\ v_t\}$.

For each path P calculated, check for cycles in P:

- If cycles at the beginning of path P
  - Discard P;

- If cycles in the middle of path P
- Remove cycle to obtain a shorter path P

4.3.2 Second objective function $\sum b_{ij}T_{ij}$:

The main objective is to maximize the above objective function. Here

b_{ij} : Represents the value 0 or 1

1: If there is an edge between node i and j

0: If there is no edge between node i and j

T_{ij} : Represents the traffic between node i and j.

Aim of the first objective function is to minimize the number of hops for a packet to reach from source node to destination node. Since the target topology is an regular graph De Brujin, We are sure about the maximum no of hops a packet can take, will be not more than k.

The second objective function's aim is to place the nodes with higher congestion values close to each other such that the congestion can be minimized.

4.4 Stop Criteria:

Reproduction, crossover and mutation operations and the results of the objective function are the entire GA needs to optimize a solution set. The goal of the GA is to generate a single exceptional individual with genes that would represent the optimal solution to a problem, which is a bit contrary to the workings of evolution, which operates to improve the population as a whole. However, to do this, the GA must generally improve the average quality of the solution set, just as evolution does in the biological world .One important aspect of the GA is its stopping criteria. There are three popular stopping criteria:

- A certain number of generations is reached
- A solution falls within an acceptable fitness margin.
- There have been a number of generations with no improvement in the solution set.

Dependent on the goals of the particular implementation, any one of these may be more viable.

The algorithm implemented in this thesis is a combination of on-line performance and fixed number of generations. Here the concept of convergence is used, which corresponds to the chromosomes in a population become identical or similar to each other. In general a GA can be considered converged if the average fitness of the population is at least 90% of the best fitness, which is called as the 90% rule.

4.5 Crossover Strategies

Selection alone cannot introduce any new individuals into the population. These are generated by genetically inspired operators, of which the best known is *crossover* and *mutation*. Crossover is performed between two selected individuals, called *parents*, by exchanging parts of their genomes to form two new individuals, called *offspring*. This operator tends to enable the evolutionary process to move toward promising regions of the search space. This chapter reviews various crossover operators at present to solve TSP problem, which is basically to find a best order of cities.

4.5.1 Order Crossover

The order crossover was proposed by L.Davis [DAVIS85] for the TSP (Traveling salesman Problem). The main idea of Order Crossover was to create an offspring by choosing a sub-sequence of a tour from one parent and preserving the relative order of cities from the other parent. The following example represents the order crossover.

Let P1 and P2 be the chromosomes for crossover. Since **OCX** does not care about the index and address, P1 and P2 can be simply written as (A B C D E F G H) and (C D A E F G H B)

P1:

Logical Node	0	1	2	3	4	5	6	7
Label	000	001	010	011	100	101	110	111
Physical Node	A	B	C	D	E	F	G	H

P2:

Logical Node	0	1	2	3	4	5	6	7
Label	000	001	010	011	100	101	110	111
Physical Node	C	D	A	E	F	G	H	B

Because the chromosome length is 8, two numbers are randomly picked from 1 to n-1 where n = 8(8 cities), as the two crossover points. Assume 2 and 6 are picked and their positions are marked by "|".

P_1: (A B | C D E F | G H)

P_2: (C D | A E F G | H B)

The offspring are produced in the following way. First the segments between the two cut points are copied in to the offspring:

Offspring1= (? ? | C D E F | ? ?)

Offspring2= (? ? | A E F G | ? ?)

Next starting from the second point of one parent, the nodes from the other parent are copied in the same order, ignoring symbols already present. When reaching the end of the string, continue from the head of the string, The sequence of the nodes in P_2(from the second cut point) is

(H - B - C - D - A - E - F - G)

To fill in the blanks marked by "?" in P_1 , nodes **C D E F** , which are already there, need to be removed .As a result , the remaining sequence is

H B A G

This sequence is placed in Offspring1 from the second cut point:

Offspring1: (A G C D E F H B)

Similarly the second offspring can be created:

Offspring2: (C D A E F G H B)

4.5.2 Union Crossover

Union Crossover was developed by Fox et al [Fox91]. The UX operator take a mutually exclusive sub string from each parent string and then write the elements directly to the offspring string ensuring the precedence relations are preserved. Figure4.5 describes this in detail with example

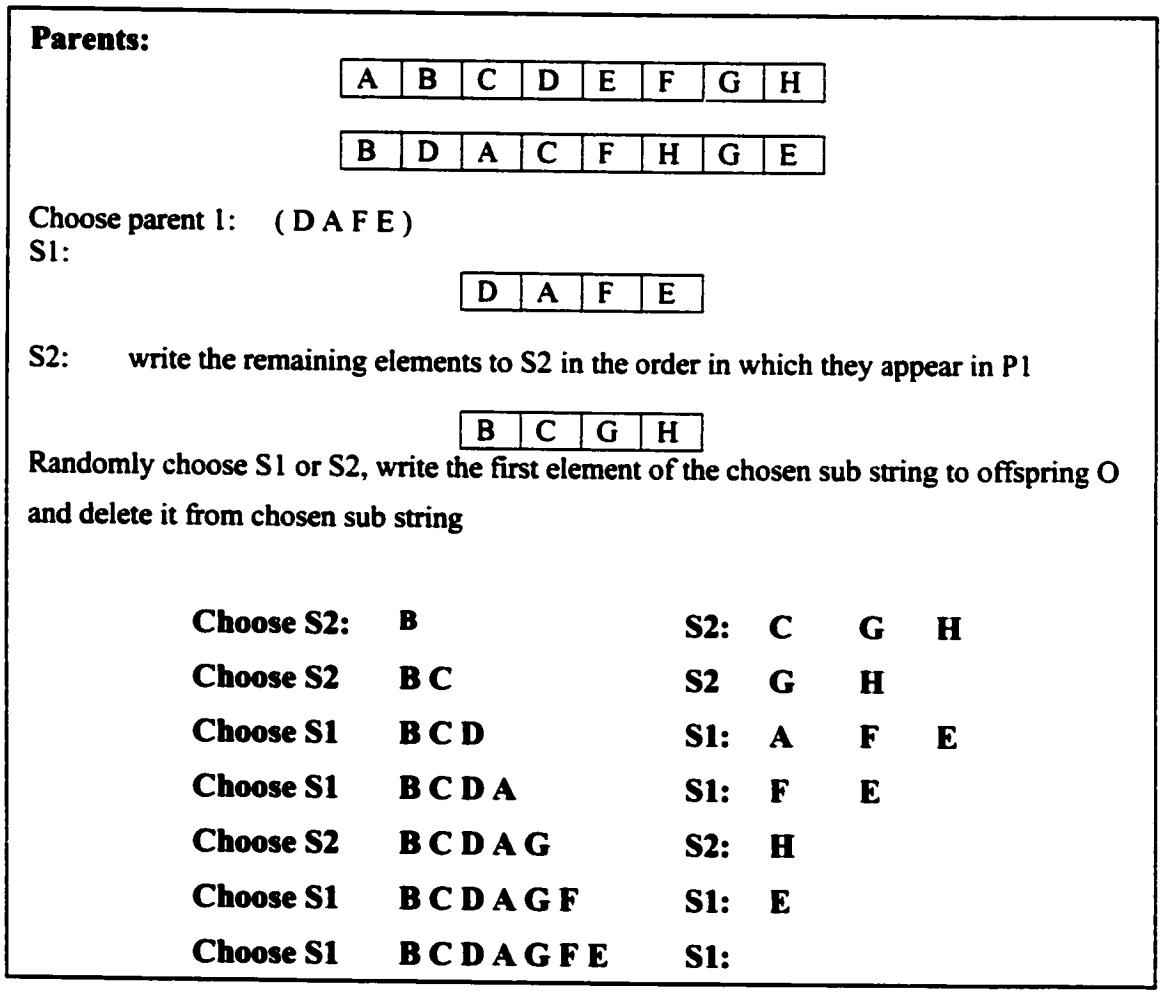


Figure 4.5 Union Crossover

4.5.3 Random Crossover

Random crossover does not care about any special structure of the chromosome representation nor does it care about the problem itself. The idea of RCX comes from uniform crossover, also known as multi-point crossover [Gol89], in SGA. In order to create offspring, RCX randomly picks some positions to exchange genetic material while the rest of the chromosomes remain unchanged. There fore, RCX exchange any element whether it is part of a cluster or not. To demonstrate how it works, the chromosomes P1 and P2 are used again.

P1: (B A C E D G H F)

P2: (C D A E F G H B)

Since the chromosome contains 8 elements, an 8-bit binary string is created. Each bit corresponds to one element. The value of the string is randomly generated. As a result a binary 01101001 is created. Using this string, a mask operation is applied to P1 (figure 4-2), If a bit is "1", the corresponding element stays unchanged in the offspring; otherwise it is removed.

0	1	1	0	1	0	0	1
B	A	C	E	D	G	H	F
			↓				
*	A	C	*	D	*	*	F

Then the vacant positions marked by "*" ar4e filled with genes from P2. The first offspring, Offspring1, is as shown below:

Offspring1 = E A C G D H B F

Applying the same strategy to P2, the second offspring can be created:

Offspring2 = C D A E F G H B

4.5.4 TBX 1 Crossover

The role of the crossover is to recombine information from two good parent solutions in to what we hope are even better "offspring" solutions. The problem is to design a crossover operator that combines characteristics of both "parents" while producing a valid solution to he ordering problem. The TBX operators are designed to break ties between two string locations competing for the same location.

In **TBX1 (Tie breaking crossover 1)** initially with the reference list of elements: a,b,c.., parent solutions are coded using position listing

Parent:

(b d c f a e) → (5 1 3 2 6 4)

(c a e b f d) → (2 4 1 6 3 5)

Randomly choose two crossover points and exchange all the characters between the crossover points as in traditional two-point crossover.

(5 1 | 3 2 6 | 4)

(2 4 | 1 6 3 | 5)

Now generate a crossover map, which is a random ordering of integers from 0 to n-1. Where n is the number of elements. Multiply each character of each string by n and add the corresponding number in the crossover map.

Eg. **(5 0 1 2 4 3)**

(53 6 7 38 22 27)

(17 24 19 14 40 33)

Replace the lowest character in each string by 1, the next lower by 2 etc up to n.

(5 1 2 6 3 4)

(2 4 3 1 6 5)

Map the string back to elements. These are the offspring solutions.

Offspring:

(5 1 2 6 3 4) → (b c e f a d)

(2 4 3 1 6 5) → (d a c b f e)

4.5.5 TBX 2 Crossover

TBX2 crossover is a modification of TBX1 crossover. Here instead of selecting two random crossover points only one-crossover points are selected. The crossover process is implemented as follows.

Parent:

b	d	c	f	A	e	→	5	1	3	2	6	4
c	a	e	b	F	d	→	2	4	1	6	3	5

Randomly choose one initial element x and exchange the characters corresponding to that element.

Eg., x = d => exchange 4th character

5	1	3	6	6	4
2	4	1	2	3	5

Now generate a crossover map, which is a random ordering of integers from 0 to n-1. Where n is the number of elements. Multiply each character of each string by n and add the corresponding number in the crossover map.

Eg.

5	**0**	**1**	**2**	**4**	**3**
(35	**6**	**19**	**38**	**42**	**27)**
(17	**24**	**7**	**14**	**28**	**33)**

Replace the lowest character in each string by 1, the next lower by 2 etc up to n.

5	**1**	**2**	**6**	**4**	**3**
3	**4**	**1**	**2**	**5**	**6**

Map the string back to elements. These are the offspring solutions.

b	**C**	**f**	**e**	**a**	**D**
c	**D**	**a**	**b**	**e**	**F**

Randomly choose the number of neighbor pairs to carry across. Delete this number of x's original neighbors in the parent string.

E.g., r = 1 $\Rightarrow$ Delete b & c in P1

Delete f & c in the P2.

-	**-**	**f**	**e**	**a**	**D**
-	**D**	**a**	**b**	**e**	**-**

Move elements to make r spaces on each side of x leave x fixed in its original position.

-	F	e	a	-	D
-	D	-	a	b	e

Fill in the spaces with x's original neighbor maintaining the original order.

Offspring:

c	F	e	a	b	d
f	D	c	a	b	e

4.5.6. Cluster Crossover

This crossover is similar to the proposed crossover in this thesis (Sub graph crossover). This crossover mainly concentrates on preserving the entire group of nodes that surround the root node. This is mainly due to the fact that to minimize the congestion, two nodes must be placed as close to each other as possible if they have high traffic. In this crossover instead of selecting a selected number of nodes surrounding the root node, we preserve the entire nodes surrounding the root node there by preserving good cluster in the chromosome.

There by the proposed crossover protects the entire cluster and the unfilled positions in the first chromosome can be filled with genes from the second chromosome.

C_1:

Logical Node:	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Label	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
Physical Node	**C**	**E**	**D**	**H**	**B**	**A**	**G**	**F**

Table: 4-2.1 Chromosome C_1

C_2:

Logical Node:	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Label	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
Physical Node	**C**	**D**	**A**	**E**	**F**	**G**	**H**	**B**

Table: 4-2.2 Chromosome C_2

As an example, let **C_1 and C_2** be the chromosomes for crossover. The proposed crossover creates the new offspring as follows.As an example, let **C_1 and C_2** be the chromosomes for crossover. The proposed crossover creates the new offspring as follows.

1. Randomly pick a node as the root of the sub-graph. Assume N (001) is chosen.
2. Node N has two predecessors, 000 and 100, and two successors 010 and 011.This crossover protects both the predecessors and successors. There fore the root node (001) and its 4 adjacent nodes including two predecessors (000 and 100) and one successors (010) are filled in the Offspring^1:

Offspring^1:

Logical Node:	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Label	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
Physical Node	**C**	**E**	**D**	**H**	**B**	*	*	*

3 Fill the vacant positions, marked by "*", with genes from C_2. After getting rid of those nodes(**C, E, D, H,B**) already in Offspring1, (**A, F, G**) are left. Fill the "*" marked places in Offspring1 with (**A, F, G**) as it appears in C_2. Thus, an offspring is created:

Offspring^1:

Logical Node:	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Label	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
Physical Node	**C**	**E**	**D**	**H**	**B**	**A**	**F**	**G**

Similarly applying the same crossover to C_2 and C_1, **Offspring2** can be created

4.6 Conclusion

This chapter proposes a new genetic algorithm to solve the problem of designing an optimum ordering of nodes for the logical topology of an optical network. An outline of its working mechanism, the objective functions and the stopping criteria was explained. At the end a detailed explanation of different crossover strategies that have been investigated are presented. The experimental results are presented and analyzed in Chapter 5.

Chapter 5

Experimentation Results

The main objective of our experiments is to test our algorithm in designing an optimum ordering of nodes for the logical topology of an optical network using the Genetic Algorithms. A number of experiments are performed on different sizes of networks, which are distinct from one another in terms of their physical connectivity, sizes and the traffic matrices used. Experiments in this thesis focus on various key parameters and how they affect the performance of the GA. The key parameters of interest are crossover strategy, number of generations, population size, persentage of mutation and crossover.

5.1 Control parameters

Experiments in this thesis will focus on various parameters that affect the performance of the genetic algorithm. These parameters include the crossover rate, mutation rate, population size, the number of generations and the traffic matrix. In order to provide and maintain good diversity of the population, a larger population size of 300 is also used. The minimal and maximal numbers of generations are 100 and 150.

The initial values of all the key parameters are as follows:

- All the test runs are of 100 runs
- Crossover rate: 0.6
- Mutation rate: 0.05-0.1
- Population: 100-400

Traffic matrix is a randomly generated matrix where, randomly values are assigned between each source and destination pair.

	A	B	C	D	E	F	G	H
A	0	8	6	3	0	1	2	8
B	2	0	9	3	5	4	7	6
C	1	5	0	6	9	8	4	1
D	2	5	8	0	3	9	7	4
E	0	2	5	4	0	8	6	3
F	2	0	1	4	3	0	8	2
G	0	2	8	6	8	6	0	4
H	8	2	0	1	0	3	6	0

Table 5.1 Traffic Matrix for 2^3 De Brujin Graph

Since it is extremely difficult to determine the optimal logical topology, this thesis will evaluate the performance of the GA in terms of following factors,

- Convergence of GA and
- The fitness of the best member with respect to the best member among the possible *n!* members. (*n!* Ordering of nodes where *n* = number of nodes).

5.2 Results for Sub-Graph Crossover

In this thesis we studied the effectiveness of sub graph crossover and cluster crossover strategies and compared their performance in getting the optimal solution from the possible no of generations (Max_gen). The results are then compared using the two objective functions discussed in the chapter 4.At first the optimum ordering of nodes, which with the highest fitness value among the n! ordering of nodes (where n is the no of nodes) is obtained. It is then compared with the optimal ordering obtained from the sub graph crossover and cluster crossover strategies

# Nodes	Pop Size	Best Possible Solution	Fitness	Best Solution	Fitness	Iterations	Time
8	100	1 2 5 3 7 6 4 0	37	2 5 3 7 6 0 4 1	33	100	12sec
	200			1 2 4 0 6 5 3 7	36	100	24sec
	300			1 2 5 3 7 6 4 0	37	100	41sec

Table 5.2.1 Experiment Results for Sub graph crossover using *objective function*[1]

# Nodes	Pop Size	Best Possible Solution	Fitness	Best Solution	Fitness	Iterations	Time
8	100	2 0 3 7 4 1 5 6	48	3 7 4 0 2 5 1 6	47	100	16sec
	200			2 0 3 7 4 1 5 6	48	100	23sec
	300			3 7 4 1 5 6 0 2	47	100	54sec

Table 5.2.2 Experiment Results for Sub graph crossover using *objective function*[2]

5.3 Results for Cluster Crossover

# Nodes	Pop Size	Best Possible Solution	Fitness	Best Solution	Fitness	Iterations	Time
8	100	1 2 5 3 7 6 4 0	37	2 5 3 7 6 4 1 0	35	100	14sec
	200			0 4 1 2 5 3 7 6	36	100	21sec
	300			2 5 3 7 6 4 1 0	35	100	29Sec

Table 5.3.1 Experiment Results for Cluster crossover using *objective function*[1]

# Nodes	Pop Size	Best Possible Solution	Fitness	Best Solution	Fitness	Iterations	Time
8	100	2 0 3 7 4 1 5 6	48	6 5 2 0 3 7 4 1	47	100	24sec
	200			2 0 3 7 4 1 6 5	48	100	27sec
	300			3 7 4 0 2 1 5 6	47	100	33sec

Table 5.3.2 Experiment Results for Cluster crossover using *objective function*^{2}

Each of the above experiments is carried for 8-node network and for 100 iterations. Most of the time the solution fitness are slightly less than or sometimes inferior when compared with the best solution fitness. To obtain a solution with fitness on par with the best solution, elitism is used to preserve the best solution in a generation for the nest generation.

For an 8-node network, both the Sub graph and cluster crossover perform satisfactorily for population sizes of 100 and 200.Most of the time solutions fitness's are about 90% of the best solution. But the results produced are unsatisfactory, as the program never converges. So to eliminate the problem of non-convergence, elitism was introduced. Elitism always retains the best individual in the population, found so far and copied in to the population set for the next generation.

In our next experiment we introduced 10 % elitism such that 10% of the best solutions in a generations are preserved for the next generation

Elitism:

# Nodes	Pop Size	Best Possible Solution	Fitness	Best Solution	Fitness	Iterations	Time
8	100	1 2 5 3 7 6 4 0	37	0 4 1 2 5 3 7 6	36	100	12sec
	200			0 4 1 2 5 3 7 6	36	100	20sec
	300			0 4 1 2 5 3 7 6	36	100	30sec

Table 5.4.1 Experiment Results for Sub Graph crossover using *objective function*[1]

# Nodes	Pop Size	Best Possible Solution	Fitness	Best Solution	Fitness	Iterations	Time
8	100	2 0 3 7 4 1 5 6	48	3 7 4 1 5 6 0 2	47	100	14sec
	200			3 7 4 1 5 6 0 2	47	100	19sec
	300			3 7 4 1 5 6 0 2	47	100	24sec

Table 5.4.2 Experiment Results for Sub Graph crossover using *objective function*[2]

The results found to be encouraging with the introduction of elitism for the population of 100 and 200.It was found that population with size of 100 had a 100% convergence and approximately 95% for the population size of 200.And another significant factor to be noticed is the time it took to get the optimal solution. It was found that it took significantly less time when compared to the non-elitist one.

As the number of nodes increases, ie, from 8-node to 16 and 27 node network, the search space increases from 8! To 16! And 27! . To compare the results obtained from the

algorithm to the best solution among these 16! and 27! ordering of nodes is practically time consuming, as we need to check each of the 16! (20922789888000 solutions) solution's fitness to get the best solution.

The following results will give a brief idea about how the results varied as the number of nodes increase. Results are tested for various values of crossover rate, mutation rate and the time it took to converge.

#Nodes	SIZE	Elitism	Objective Function	Fitness	No of Iterations	Time
8	100	NO	1	41	180~200	17sec
	100	YES	1	41	140~170	16sec
	100	NO	2	46	>200	>30 sec
	100	YES	2	46	>200	>30 sec
	200	NO	1	41	485~500	56sec
	200	YES	1	41	325~400	45sec
	200	NO	2	46	>200	>30 sec
	200	YES	2	46	>200	>30 sec
16	100	NO	1	61	150~250	61sec
	100	YES	1	61	150~250	56sec
	100	NO	2	72	>500	-
	100	YES	2	72	>500	-
	200	NO	1	61	-	-
	200	YES	1	61	-	-
	200	NO	2	72	-	-
	200	YES	2	72	-	-

27	100	NO	1	73	200~250	81sec
	100	YES	1	-	-	-
	100	NO	2	-	-	-
	100	YES	2	-	-	-

From the above results it is evident that the sub graph crossover strategy works well with the objective function[1] and the algorithm is converging after a reasonable number of generations. Elitism is introduced to solve the problem of non-convergence, but it is successful for smaller network like 8-node and it made a negligible effect on the larger networks. Also the crossover rate is increased from 0.7 to 0.9 to increase the rate of convergence in the case of second objective function[2], but it had a little impact on the performance of the algorithm. In the above experiments the algorithm never converged for 16-node and 27-node network for population of 200 and 300. The main reason for the results given above is that, for larger networks, the search space becomes too large that no crossover strategy with such a small population can cover a reasonable amount of the search space within such limited number of generations.

The above results are compared with the results generated by the cluster crossover and it found that the cluster crossover results are comparable with the sub graph crossover and some times better than the previous one.

5.3 Results for Cluster Crossover

The following table gives the brief description of the results generated for 8,16 and 27-node networks.

#Nodes	SIZE	Elitism	Objective Function	Fitness	No of Iterations	Time
8	100	NO	1	41	120 - 150	12sec
	100	YES	1	41	140 - 170	14sec
	100	NO	2	46	200 - 250	44 sec
	100	YES	2	46	200 - 240	38 sec
	200	NO	1	41	485 - 500	56sec
	200	YES	1	41	450 - 480	45sec
	200	NO	2	46	750 - 800	>2 min
	200	YES	2	46	550 - 600	100sec
16	100	NO	1	61	150 - 250	61sec
	100	YES	1	61	150 - 250	56sec
	100	NO	2	72	150 - 250	40sec
	100	YES	2	72	150 - 250	37sec
	200	NO	1	61	-	-
	200	YES	1	61	-	-
	200	NO	2	72	400 - 450	56sec
	200	YES	2	72	700 - 750	137sec
27	100	NO	1	73	200 - 250	81sec
	100	YES	1	-	-	-
	100	NO	2	-	-	-
	100	YES	2	-	-	-

5.4 Conclusions

In this chapter various experiments were conducted by changing the values of parameters that affect the performance of the genetic algorithm. And the results can be summarized as follows.

- For smaller networks like 8-node network both the sub graph crossover and cluster crossover perform well and the results are satisfactory. The algorithm always finds out the better solution in a reasonable amount of time.
- With the increase in the number of nodes in a network, it takes lot of time to get the desired result and some times it never converges to get the result. Even with the introduction of elitism, the results are unsatisfactory. Good genes at the start and larger population sizes may solve this problem.

Chapter 6

Conclusions and Future Work

This chapter outlines the contribution of this thesis and discusses the conclusions that have been reached. A brief outline of directions for the future work is given and also the strategies for the further improvement.

6.1 Conclusions

The genetic algorithm outlined in this thesis meets our objective of designing an optimum ordering of nodes for the logical topology of an optical network using the Genetic Algorithm approach. A genetic algorithm was given to solve the problem of finding an optimum ordering of nodes that is comparable with the best possible solution. The results obtained are encouraging and comparable with the best solutions. For a small network of 8-nodes results are encouraging and results are on par with the best possible solution and it takes a matter of seconds to get the result.

This thesis analyses various parameters that affect the performance of genetic algorithms like population size, elitism, no of generations and the crossover strategies. A few new crossover strategies like sub graph crossover and cluster crossover are introduced and compared with the standard crossovers like OX, TBX1, TBX2, Cluster Crossover, Union Crossover and Random Crossover.

Since this algorithm is tested with smaller network like 8-node network and compared with the best possible solution, it may be necessary to use ***Parallel Genetic Algorithm*** (PGA) to solve the problem for larger networks. Some mathematical optimization techniques like MILP and LP can be used to find the best topology but they fail due to the drawback of generating huge number of constraints and it is practically intractable. Therefore GA's with its attractive features like high flexibility and high efficiency can be used to solve the problem of finding the best results.

6.2 Future Work

The following are some of the topics that need to be explored in improving the algorithm

- The optimum ordering of nodes for the logical topology of an optical network using the Genetic Algorithm approach will need not to be a best ordering in terms of minimizing the congestion. So these solutions should be checked using the routing.

- Since this algorithm is used to solve the problem of smaller networks, we need large populations to solve the larger networks like 27–node or 32-node. There fore we need a different approach like PGA ***(Parallel Genetic Algorithm)*** to solve the above problem.

- The target logical topology in this thesis is De Brujin graph, which will support a regular topologies which can be expressed as d^k, where d^k is the total number of

nodes in the network. To solve the problem of designing a logical topology for an irregular graph we need a regular topology capable of supporting an arbitrary number of nodes such as GEMNET.

7. References

[Acam87] Acampora. S, *A Multichannel Multihop Local lightwave Network, Proc. IEEE GLOBECOM '87*, pp. 14591467, Nov. 1987.

[AH96] Ali Amiri and Hasan Pirkul, *Routing and capacity assignment in back bone communication networks.*

[BAC93] Bäck. T. *Optimal Mutation Rates in Genetic Search.* In ICGA5, pp. 2-8, 1993.

[BAC96] Bäck. T. *Evolutionary Algorithms in Theory and Practice - Evolution Strategies, Evolutionary Programming, Genetic Algorithms.* New York, Oxford: Oxford University Press, 1996.

[Bak85] Baker. J.E. *Adaptive selection methods for genetic algorithms.* In John J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, pages 101--111. Texas Instruments, Inc. and Naval Research Laboratory, Lawrence Erlbaum Associates, 1985.

[BALUJA 94] Baluja. S *Population Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning* (1994).

[BENN99] Prügel-Bennett. A. *The Mixing Rate of Different Crossover Operators*, Proceeding due in spring 2001.

[BFGS93] Spears. W.M, De Jong. K.A, Back.T, David Fogel, and Hugo de Garis. *An overview of evolutionary computation.* In Proceedings of the European Conference on Machine Learning, pages 442--459, 1993. 15

[BOOK82] Booker.L.B. *Improving Search in Genetic Algorithms.* In L. Davis, editor, Genetic Algorithms and Simulated Annealing. Lawrence Erlbaum, 1982.

[BRUN93] Ralf Bruns. *Direct chromosome representation and advanced genetic operators for production scheduling.* In Proceedings of the Fourth International Conference on Genetic Algorithms, 1993.

[BS 93] Bäck. T. and Schwefel. H.-P *An overview of evolutionary algorithms for parameter optimization. Evolutionary Computation*, 1(1), pp. 1-23, 1993.

[BS99] Banerjee. V.J.S and Shah. S, *Regular multihop logical topologies for lightwave networks*, in Technical Report 97-1, Dept. of ECE, Stevens Institute for Technology, Hoboken, NJ, 1999. IEEE Communications Surveys.

[CARLO97] Carlo. M. *Mutation Rates as Adaptations.* Massachusetts Institute of Technology Cambridge, MA 02139.1997.

[CARSE93] Carse.B, Terence C. Fogarty, and Alistair. Munro. *Evolutionary Learning in Computational Ecologies: An Application to Adaptive Distributed Routing in Telecommunication Networks.*

[CES89] Caruana. R.A, Eshelmann, L. A. and Schaffer, J. D. *Representation and hidden bias II: Eliminating defining length bias in genetic search via shuffle crossover.* In Eleventh International Joint Conference on Artificial Intelligence, Sridharan, N. S. (Ed.), vol. 1, pp. 750-755, San Mateo, California, USA: Morgan Kaufmann Publishers, 1989.

[CLAUS 97] Hillermeier. C, Weber. D. *Optimal routing in private networks using genetic algorithms.* ISS'97 World Telecommunications Congress. Global Network Evolution: Convergence or Collision?'. Proceedings, p. 2 vol., 523-31 vol.1

[COLI96] Coli. M, Gennuso. G and Palazzari. P. *A new crossover operator for Genetic Algorithms*. Proceedings of the IEEE International Conference on Evolutionary Computation ICEC'96, May 20-22 1996, Nagoya (Japan)

[DAVIS87] Davis. L *Genetic Algorithms and Simulated Annealing*, San Mateo: Morgan Kaufmann, 1987, pages 61--73.

[DAVIS85] Davis. L, Applying *Adaptive Algorithms to epistatis Domains*, Proceedings of the International Joint Conference on Artificial Intelligence, 162-164.1985.

[DAVIS89] Davis. L. *Adapting operator probabilities in genetic algorithms*. Proceedings of the Third International Conference on Genetic Algorithms, 60-69.1989

[Davis91] Davis. L. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, N.Y

[Davis93] Davis. L, David Orvosh, Anthony Cox, and Yuping Qiu. *A genetic algorithm for survivable network design*. In Proceedings of the Fifth International Conference on Genetic Algorithms, pages 408--415, Champaign, IL, 1993

[DJONG75] DeJong. K.A. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975.

[DJONG90] DeJong. K.A and Spears. W. M. *An analysis of the interacting roles of population size and crossover in genetic algorithms*. In H.-P. Schwefel and R. M¨anner, editors, Parallel Problem Solving from Nature, pages 38--47. Springer-Verlag, 1990.

[DJONG93] De Jong. K. A. (1993). *Genetic algorithms are NOT function optimizers*. In Foundations of Genetic Algorithms 2, D. Whitley (Ed.). San Mateo, CA: Morgan Kaufmann.

[DJONG93] De Jong. K.A and Sarma. J. *Generation gaps revisited.* In L. D. Whitley, editor, *f*oundations of Genetic Algorithms *2*, pages 19-28, San Mateo, 1993. Morgan Kaufmann.

[EAST93] Fred. F. Easton and Nashat Mansour. *A Distributed Genetic Algorithm for Employee Staffing and Scheduling Problems.* In International Conference on Genetic Algorithms 1993, pages 360--367, 1993.

[EIBEN97] Eiben. A.E. (1997). *Multi-parent recombination.* In T. Back, D. Fogel & Z. Michalewicz (Eds.), Handbook of Evolutionary Algorithms (pp. 25-33). IOP Publishing Ltd. and Oxford University Press.

[ERR94] Eiben. A.E, P-E. Rau'e, and Zs. Ruttkay. *Genetic algorithms with multi-parent recombination.* In Parallel Problem Solving from Nature - 3, LNCS 866, pages 78--87. Springer-Verlag, 1994.

[ES93] Eshelman. L. J. and J. D. Schaffer (1993). *Crossover's niche.* In S.Forrest, ed., Fifth Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, pp. 9--14.

[EST97] Eiben. A.E, Sprinkhuizen-Kuyper, I., and Thijssen, B. (1997). *Competing crossovers in an adaptive GA framework.* Submitted for publication.

[Fox91] Fox, B.R and McMohan, M.B *Genetic operators for sequencing problems.* In foundations of Genetic Algorithms.pp284-300.Kaufmann (1991).

[GAB99] Gabriela Ochoa 1999.*The Multiple Roles of Recombination in Gas Centre for the Study of Evolution Centre for Computational Neuroscience and Robotics* COGS - The University of Sussex.

[GAZEN99] Gazen. C. and Ersoy. C. *Genetic algorithms for designing multihop lightwave network topologies*, Artificial Intelligence in Engineering, Elsevier Publishers, Vol.13, Issue 3, pp. 211-221, July 1999.

[GGR93] Gerstel. O, Green. P.E. and Ramaswami. R. *Architecture for an optical network layer.* IBM T.J Watson research center, NY, USA.

[GOLD89] Goldberg. D. E. *Genetic Algorithms in Search*, Optimization, and Machine Learning. Addison-Wesley, Reading, Massachusetts, 1989.

[GOLDD93] Goldberg. D. E, Deb, K., & Thierens, D. (1993). *Toward a better understanding of mixing in genetic algorithms*. Society of Instrument and Control Engineers Journal, 32(1), 10--16.

[GOLDT93] San Mateo. C.A, Morgan Kaufmann. Thierens. D and Goldberg, D. E. *Mixing in genetic algorithms.* In Forrest, S. (Ed.), Proceedings of the Fifth International Conference on Genetic Algorithms (pp. 38--45).

[GOMEZ97] M de las Mercedes Gomez-Albarran, Anan M Fernandez-Pampillon-Cesteros and Juan Manuel Sanchez-Perez, *A routing strategy based on genetic algorithm, Microelectronics Journal 28(1997)*

[GOO95] Goonatilake. S, Treleaven. P. *Intelligent Systems for Finance and Business*, Wiley, Chichester. (1995).

[Green92] Green. P.E. Fiber-*Optic Networks*. Prentice Hall, 1992.

[GS97] Sushil. J. Louis and Gong Li ,*Genetic Algorithms with Memory for Traveling Salesman* Dept. of Computer Science University of Nevada, Reno, NV 89557 sushil@cs.unr.eduli_g@cs.unr.edu

[HJBS00] Haque. A, Jackel. A, Bandyopadhyay. S and Sengupta. A. *Efficient Heuristic for Designing the Topology of Multi-hop Optical Networks*, Photonics-2000, Dec. 18-20, in Calcutta, India.

[HK88] Michael G. Hluchyj and Mark J. Karol. *Shuffle Net: An application of generalized perfect shuffles to multihop lightwave networks*. In Proceedings of IEEE INFOCOM '88, pages 379--390, New Orleans, Louisiana, March 1988.

[HKM95] Inki Hong, Andrew. B. Kahng, and Byung. Ro Moon. *Exploiting synergies of multiple crossovers: initial studies*. In ICEC'95 [859], pages 245-250.

[HL99] Georges Harik, Fernando Lobo, A parameter-less genetic algorithm, (IlliGAL Report No. 99009).

[HLG98] Harik. G, Lobo. F.G. and Goldberg. D. E. *The compact genetic algorithm*, in Proceedings of the IEEE Conference on Evolutionary Computation, pp. 523-528, (1998).

[HOLL75] Holland. John H. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press.

[HOLL92] Holland. John H, *Genetic Algorithms*, Scientific American July 1992

[KAL94] Kalyanmoy Deb. *Genetic Algorithm in Search and Optimization: The Technique and Applications*.

[KIDW93] Kidwell. M. *Using genetic algorithms to schedule distributed tasks on a bus-based system*. In Proceedings of the Fifth International Conference on Genetic Algorithms, pages 368--374, July 1993.

[KOZA92] Koza. J. *Genetic programming on the natural programming of Computers by means of Natural Selection*.

[LGDT93] Larra~naga. P, Gra~na. M, D'Anjou. A. and Torrealdea. F.J. *Genetics algorithms elitist probabilistic of degree 1, a generalization of simulated annealing*, in Proceedings of the Third Congress of the Italian Association for Artificial Intelligence (1993).

[LISEIB96] Joanna. L. and Eiben. A. E. *A Multi-Sexual Genetic Algorithm for Multiobjective Optimization*. In Toshio Fukuda and Takeshi Furuhashi, editors, Proceedings of the 1996 International Conference on Evolutionary Computation, pages 59--64, Nagoya, Japan, 1996. IEEE.

[MERZ97] Merz. P and Freisleben. B. *Genetic Local Search for the TSP: New Results*. In Proceedings of the 1997 IEEE International Conference on Evolutionary Computation, pages 159--164, IEEE Press 1997.

[MF97] Merz. P and Freisleben. B. *Genetic Local Search for the TSP: New Results*. In Thomas Baeck, Zbigniew Michalewicz, and Xin Yao, editors, Proceedings of IEEE International Conference on Evolutionary Computation, pages 159--164, 1997.

[MICH91] Michalewicz. Z and Janikow. C. *Handling constraints in genetic algorithms*. In R. Belew and L. Booker, editors, Genetic Algorithms, pages 151--157, 1991.

[MICH94] Michalewicz. Z. *Genetic Algorithms + Data Structures = Evolution* Programs, Second, Extended Edition, 1994. Springer-Verlag.

[MS93] Mitchell. M and Forrest. S (1993). *Genetic Algorithms and Artificial Life*. Santa Fe Institute. Working paper 93-11-072.

[MS94] Moore. S.J and Sinclair. M.C *Design of Routing Tables for a Survivable Military Communications networks using Genetic Algorithms*.

[MSV93b] Mühlenbein. H. and Schlierkamp-Voosen D. *Analysis of Selection, Mutation and Recombination in Genetic Algorithms*. Technical Report 93-24, GMD, 1993.

[MUK97] Mukherjee. B. *Optical Communication Networks*, McGrawHill, 1997.

[MV93] Heinz. Muhlenbein and Dirk Schlierkamp-Voosen. *Predictive models for the breeder genetic algorithm: Continuous parameter optimization.* Evolutionary Computation, 1(1):25--49, 1993.

[MV93] M¨uhlenbein. H and Schlierkamp-Voosen, D. (1993). *Optimal Interaction of Mutation and Crossover in the Breeder Genetic Algorithm* (1993)

[PC95] Poon. P.W and Carter, J.*N Genetic Algorithm Crossover Operators for Ordering Applications*, 1995.

[PHILIP90] Husbands. P. *Genetic Algorithms for Scheduling.* AISB Quarterly, No. 89.

[RALF96] Ralf Salomon. *The influence of different coding schemes on the computational complexity of genetic algorithms in function optimization.* In Voigt et al. [163], pages 227--235.

[ROJER00] Rogers. A and Prügel-Bennett,A. *Evolving populations with overlapping generations.* Theoretical Population Biology, 57(2):121-129, 2000.

[ROJERS99] Rogers. A and Prügel-Bennett,A. *Genetic drift in genetic algorithm selection schemes. IEEE Transactions on Evolutionary Computation*, 3(4): 298-303, 1999.

[RS96] Ramaswami. R, Sivarajan. Kumar. N. *Design of Logical Topologies for Wavelength Routed Optical Networks*, IEEE J. on selected areas in Communications, Vol. 14,No. 5, June 1996.

[RS98] Ramaswami. R, Sivarajan. Kumar. N. *Optical Networks: a practical Perspective*, Morgan Kaufman Publishers, Inc.Sanfransisco, California, 1998.

[RUDOLPH94] Rudolph. G. *Convergence of non-elitist strategies*. In Michalewicz, Z., Schaffer, J. D., Schwefel, H.-P., Fogel, D. B., Kitano, H., editors, Proceedings of the First IEEE International Conference on Evolutionary Computation, pages 63-66, IEEE Press. (1994).

[SCHA89] Schaffer. J.D, Caruna, R.A, Eshelman L.J., and Das, R. *A study of control parameters affecting online performance of genetic algorithms for function optimization*. In J.D. Schaffer, editor, Proceedings of the Third International Conference on Genetic Algorithms, pages 51--60. Morgan Kaufmann, 1989.

[SCHA93] Schaffer. C. *Over fitting avoidance as bias*. Machine Learning, 10, 153--178. (1993).

[SDJ91a] Spears. W.M. and De Jong, K. A. On the Virtues of Parameterized Uniform Cross over. In [ICGA4], pp. 230-236, 1991.

[SDJ91b] Spears. W.M. and De Jong, K. A. An Analysis of Multi-Point Cross over in [FGA1], pp. 301-315, 1991.

[SDJ91c] De Jong. K.A, Spears, W.M. *An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms* (1990) kdejong@aic.gmu.edu

[SGD93] Schultz. Alan C, Grefenstette, J. J., and De Jong, K. A. (1993) *Test and Evaluation by Genetic Algorithms*, IEEE Expert, v8, n5,9-14, October 1993.

[SINCLAIR92] Sinclair. A. *Improved bounds for mixing rates of Markov chains and multicommodity flow*. Combinatorics, Prob., Comput. 1, 351--370. (1992).

[SINCLAIR93] Sinclair. A. *Algorithms for Random Generation and Counting: A Markov Chain Approach*. Birkhauser, Boston-Basel-Berlin, 1993.

[SPEARS91] Spears. W. M. and Anand, V. *A study of crossover operators in genetic programming*. Proceedings of the Sixth International Symposium on Methodologies for Intelligent Systems (pp. 409-418). Charlotte, NC: Springer-Verlag. (1991).

[SPEARS91] Spears, W. M. and De Jong, K.A (1991a). *An analysis of multi-point crossover.*

[SPEARS92] Spears. W.M. *Crossover or Mutation* In Whitley (ed.) Foundations of Genetic Algorithms-2. 221237.

[SPEARS93] Spears. W.M. *Adaptive Crossover in a Genetic Algorithm Naval Research Laboratory.*

[SPEARS95] Spears. W. M. *Adaptive Crossover in a Genetic Algorithm* In Proceedings of the 5th Conference on Evolutionary Programming, J. R. McDonnell, R. G. Reynolds and D. B. Fogel (eds.), Cambridge: MIT Press, pp 367-384. (1995).

[SPEARS98] Spears. W.M. *The Role of Mutation and Recombination in Evolutionary Algorithms*. PhD thesis, George Mason University, Fairfax, Virginia. (1998).

[SR91] Sivarajan. K, Ramaswani. R. *Multihop light wave networks based on De Brujin Graphs*. Proc IEEE INFOCOM, 1991, pp. 1001–1011.

[SSB88]. Sengupta. A, Sen. A and Bandyopadhyay. S, *Fault-tolerant Distributed System Design*, IEEE Trans on circuits and Systems, Vol.35, No. 2, Feb 1988.

[SVS99] Subrata. Banerjee, Vivek. Jain, Sanjay. Shah, *Regular Multihop Logical Topologies for Light wave Networks.* IEEE Communications Survey First Quarter 1999.Vol. 2, No. 1.

[SYS89] Syswerda. G. *Uniform crossover in genetic algorithms* [ICGA3], pp. 2-9, and 1989.

[TANG97] Ko. K.T, Tang. K.S, Chan. C.Y, Man. K.F, Kwong, S. *Using Genetic Algorithms to Design Mesh Networks*. Computer 30 (1997) 56-61

[THIE93] Dirk. Thierens, Goldberg. D. E, *Mixing in Genetic Algorithms*. Proceedings of the Fifth International Conference on Genetic Algorithms, 1993.

[VA94] Voigt. H.M. and Anheyer.T. *Modal Mutations in Evolutionary Algorithms*. [ICGA4] Vol. I, pp. 88-92, 1994.

[VIGN91] Vignaux. G. A. and Michalewicz. Z *A genetic algorithm for the linear transportation problem*. IEEE Transactions on Systems.

[ZA95] Zhang. Z, Acampora. A. S, *A Heuristic Wavelength Assignment Algorithm for Multihop WDM Networks with Wavelength Routing and Wavelength Re-use*, IEEE/ACM Transactions on Networking, June 1995,Vol.3, pp.281-288.

Appendix A Definitions

Allele

Each gene occupies a specific character location within the chromosome string. Each gene position may take a character value called an allele.

Building Block

A pattern of genes in a contiguous section of a chromosome.

Chromosome

A data structure consisting of a character string of coded task parameters.

Convergence

Tendency of members of a population to be the same. A gene is said to have converged when 90% of the chromosomes in the population all contain the same allele for that gene. A population is said to have converged when all gene converged.

Crossover:

Crossover is the procedure by which two chromosomes mate to create a new offspring chromosome. This means that the offspring string is a copy of parent 1 up to a randomly chosen point, and a copy of parent 2 from that point onwards.

Elitist:

GA, which always retains the best individual in the population, found so far (Tournament selection is naturally elitist).

Evolution Strategy:

A search technique, where the next point to search is given by adding gaussian random noise to the current search point.

Fitness Function:

Function that evaluates a member of a population.

Fitness: fitness is a measure of how good a solution it codes. Fitness is calculated by a fitness function.

Generation:

An iteration of the measurement of fitness and the creation of a new population by means of genetic operations.

Genetic Algorithm (GA):

Model of machine learning that uses a genetic/evolutionary metaphor.

Genetic Operator:

An operator in a genetic algorithm or genetic programming, which acts upon the chromosome to produce a new individual. Example operators are mutation and crossover.

Genetic Programming (GP):

Genetic Algorithms applied to programs. Genetic Programming is more expressive than fixed-length character string GA's, though GA's are likely to be more efficient for some classes of problems.

Genotype:

"Physiological Team" in which a gene can make a maximum contribution to fitness by elaborating its chemical "gene product" in the needed quantity and at the appropriate stage of development.

Mutation:

Arbitrary change to representation, often at random.

Phenotype:

Product of the interaction of all genes.

Premature Convergence:

When a Genetic Algorithm's population converges to something that is not the solution you wanted.

Reproduction:

The genetic operation that causes an exact copy of the genetic representation of an individual to be made in the population.

Roulette wheel selection:

Roulette wheel selection is a way of picking out a string from among a group of strings (a population).

Solution:

A solution is coded by a string or chromosome. The words string and chromosome are used interchangeably.

Tournament Selection:

A mechanism for choosing individuals from a population. Groups are selected at random from the population and the best (normally only one, but possibly more) is chosen.

Vita Auctoris

Sridhar Tadicherla obtained a Bachelor of Technology degree from **R.E.C. Warangal, Kakatiya University** India in 1995.He is a candidate for a master's degree in science at the University of Windsor and hopes to graduate in 2001.