1-1-2007

Use of regular topology in logical topology design.

Chun-Hsien Vic Ho
University of Windsor

Use of Regular Topology in Logical Topology Design

by

Chun-Hsien Vic Ho

A Thesis
Submitted to the Faculty of Graduate Studies and Research
through Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada

2007

Your file Votre référence
ISBN: 978-0-494-35044-7
Our file Notre référence
ISBN: 978-0-494-35044-7

ABSTRACT

With the increases in the demand for high-speed data communication, new approached to design logical topologies for large networks are required to satisfy the user requirements. The problem of designing an optimal logical topology for WDM networks can be formulated as a mathematical optimization problem. The amount of time required for this approach is unacceptably large even for moderate sized networks. In multi processor networks, regular topologies have been investigated widely. In order to solve the problem in a reasonable amount of time, we propose to use regular topology coupled with the genetic algorithm to find a "good" logical topology for wavelength routed WDM networks. This research looks at a number of approaches to design the logical topologies and determine whether regular topologies have promise in wavelength routed WDM networks.

DEDICATION

to all the people that I love...

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to all those who gave me the opportunity to complete this thesis. I want to thank my supervisor Dr. S. Bandyopadhyay for his guidance and constant support. I have furthermore to thank Dr. D. Kao for her valuable advice. I also would like to thank Dr. D Wu and Dr. X. Yuan for being my thesis committee.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Copper wires have been replaced by optical fibers as the communication medium in computer communication because of the huge bandwidth, low signal attenuation and low distortion in optical fibers. Optical networks are interconnections of computers connected by optical fibers and provide the following features to the users.

- Protocol Transparency: The underlying network protocols do not need to be aware.
- Reliable Communication: There are fault avoidance algorithms available to guarantee low error rate.
- High Speed of Transmission: The huge capacity of optical fibers can be efficiently utilized.

Wavelength Division Multiplexing (WDM) is a technology to transmit multiple optical signals using different carrier wavelengths on a single fiber. As demand for higher transmission bandwidth increases today, optical networks coupled with WDM technologies represent most promising candidates for the solutions to this problem.

1.1 Motivation

In optical networks, the speed at which optical signals may be communicated is far greater than the speed at which data can be processed by electronic circuits. However, optical devices are much more expensive compared to electronic devices so it is important to optimize the use of optical network resources. In order to do so, a good optical network design is required. In general, optical network design problems can be decomposed into the following four subproblems [4]:

- Logical Topology Design Subproblem: Decide which node pairs in the network should be able to communicate directly with each other in optical domain. In other word, the problem is to determine which lightpaths should be created.
- Lightpath Routing Subproblem: Implement the lightpaths on the physical topology by selecting which physical link(s) should be included in the route used by each lightpaths.
- Wavelength Assignment Subproblem: Assign the wavelengths to the lightpaths without contradicting the restriction on the wavelength assignment subproblem.
- Traffic Routing Subproblem: Decide which logical path(s) should be used for each data stream so that the total payload of each edge in the logical path never exceeds the capacity of a lightpath.

Since the number of carrier wavelength that may be used on an optical fiber is now 200, in laboratory settings and increasing fast, this thesis has ignored the issues of routing and wavelength assignment (RWA) and focused on designing the optimal logical topologies of optical networks.

Different logical topologies can be set up on the same physical topology, but each logical topology has different performance characteristics. An important research area is to determine the optimum logical topology for a given optical network with the specified physical topology and expected traffic demand for each node pair. In general, the objective of logical topology design is to design a network which is as fast as possible (performance optimization) and requires the least amount of optical hardware (cost optimization). The following two factors are usually used to evaluate the performance of an optical network.

- *Congestion*: The maximum amount of data that is carried by any logical edge. It is desirable to have a lower value of congestion so that the cost of electronic hardware is reduced.
- *Delay*: The amount of time that is taken by a signal to be sent from the source node to the destination node. To ensure higher throughput, the value of delay should be as low as possible.

In this thesis, delay has not been considered.

1.2 Existing Approaches

The logical topology design problems mentioned in the previous section can be formulated as optimization problems aimed at maximizing the network throughput or other performance measures of interests. Some early researchers have solved the problems using Mixed Integer Linear formulations (MILP) [14] [24]. Since the MILP formulations are NP-complete, the actual time for solving the problems can be unacceptable large even for moderate sized networks. This fact has motivated the development of heuristic approaches for finding a "good" solution instead of an optimum solution in the reasonable amount of time [4]. The logical topologies can be represented by directed graphs, and those topologies derived by heuristic approaches are generally arbitrary graphs.

Another approach investigated in logical topology design is to consider a class of topologies called, regular graphs[1] as the logical topologies of optical networks [20]. This approach is attractive because of the low diameter of regular graphs. However, there is a major problem in directly using regular graphs as the logical topologies of optical networks since the number of vertices in a regular graph is restricted by some well-defined formula. In other word, there is very little flexibility with respect to the size of networks.

1.3 Proposed Approach

In order to solve this problem, this thesis proposed to implement a scalable topology, henceforth called the *scalable de Bruijn graph* based on the de Bruijn graph [13]. In the scalable de Bruijn graph, there is considerable flexibility in the number of nodes while retaining the advantages of de Bruijn graph. Since the in-degree and the out-degree of each node in a scalable de Bruijn graph are not exactly the same, it is called *almost regular* or *nearly regular* topology. Given the size *n* of a network, the interconnection, using the scalable de Bruijn graph, is completely defined. However,

[1] A definition of regular graphs and the properties relevant to this research appears in Chapter 2.

there is total flexibility with respect to mapping the physical nodes of the network to the logical nodes of the scalable de Bruijn graph. By selecting this scalable topology as the target logical topology, the logical topology design problem is reduced to that of finding an appropriate mapping between the physical nodes of the network and the logical nodes of the scalable de Bruijn graph. The process of mapping is accomplished using the genetic algorithm which attempts to minimize the maximum congestion of the network.

The problem is presented as follows. Given the traffic demand of an optical network, find the proper mapping and the traffic routing for the scalable de Bruijn graph such that the maximum congestion of the network is minimized. FIGURE 1 shows an example of traffic matrix which represents the traffic demand of all source destination node pairs, and FIGURE 2 is a scalable de Bruijn graph with 7 nodes.

$$T = \lambda_{sd} = \left[\begin{array}{cccccccc} & A & B & C & D & E & F & G \\ A & 0 & 6 & 48 & 24 & 33 & 42 & 45 \\ B & 12 & 0 & 45 & 30 & 3 & 30 & 18 \\ C & 39 & 9 & 0 & 24 & 45 & 15 & 27 \\ D & 27 & 48 & 15 & 0 & 42 & 18 & 15 \\ E & 33 & 15 & 33 & 3 & 0 & 42 & 36 \\ F & 48 & 9 & 45 & 12 & 39 & 0 & 33 \\ G & 36 & 21 & 36 & 21 & 15 & 12 & 0 \end{array}\right]$$

FIGURE 1: TRAFFIC MATRIX

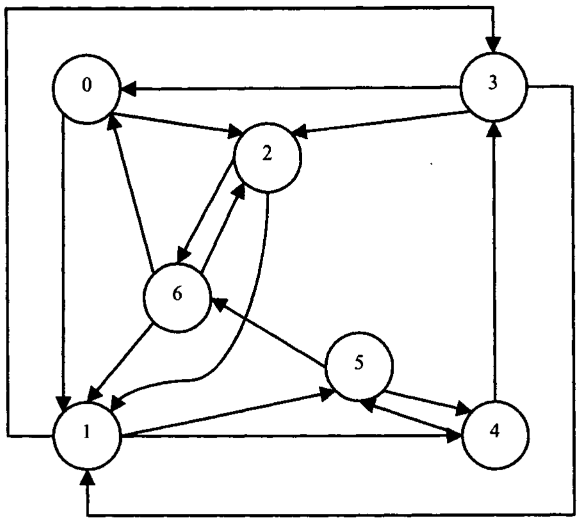

FIGURE 2: A SCALABLE DE BRUIJN GRAPH WITH 7 NODES

The logical nodes of the logical topology are represented by numbers, and the physical nodes of physical topology are represented by capital characters. In a 7-node topology, there are 7 factorial different ways of mapping these physical nodes to logical nodes. The following is an example corresponding to a 7-node network.

Logical Node	0	1	2	3	4	5	6
Physical Node	*B*	*D*	*C*	*A*	*F*	*E*	*G*

1.4 Objective

This research uses the scalable de Bruijn graph coupled with the genetic algorithm to design logical topologies. The main objective of this thesis is to compare the performance of our approach to the optimum approach and heuristic approach. In order to so, the following steps are taken.

1. Design networks of various sizes and construct the scalable de Bruijn graph corresponding to them.

2. Determine an optimal mapping of the physical nodes of the network to the logical nodes of the scalable de Bruijn graph using the genetic algorithm.
3. Determine optimal traffic routing on the logical topology obtained in (2) using CPLEX
4. Repeat the study using Heuristic Logical Topology Design Algorithm (HLDA) for the logical topology design.
5. Repeat the study using Mixed Integer Linear formulations (MILP) for the logical topology design.

1.5 Organization

The rest of this thesis is organized as follows. Chapter 2 reviews some background information relevant to optical networks and genetic algorithms. Chapter 3 presents the reported research work in detail including the use of the genetic algorithm to find the optimal mapping mentioned above. Chapter 4 gives the experimental results regarding the performance of the proposed approach. Lastly, Chapter 5 summarizes the concepts introduced in this report and the conclusion.

CHAPTER II

REVIEW OF LITERATURE

2.1 Introduction to Optical Networks

Optical network is a telecommunication system using optical fibers as communication medium to communicate and share resources. Because of the huge capacity of optical fibers, a key advantage of optical network is speed. Using current technology, the transmission bandwidth of optical networks can be up to 50 tera-bits per second (Tbps). In other words, it is theoretically possible to send 50×10^{12} bits per second using a single fiber. When an optical network uses Wavelength Division Multiplexing technologies (transmitting multiple optical signals on a single fiber), optical networks have become even more cost effective.

2.1.1 Major Components of Optical Networks

An optical network consists of three major components: *optical fibers*, *optical routers* and *end-nodes*. In an optical network, every router node connects to several input and several output fibers. Each of these fibers is a very thin glass cylinder, and is used to carry multiple incoming or outgoing optical signals. The functionality of an optical router is to direct each incoming optical signal to an appropriate outgoing fiber. Another important component of an optical network is an end-node. An end-node is typically a computer – a possible source or destination of data.

2.1.2 Terminology

Physical Topology: The physical topology of an optical network defines the interconnection of the major physical components on the network. FIGURE 3 shows the physical topology of an optical network.

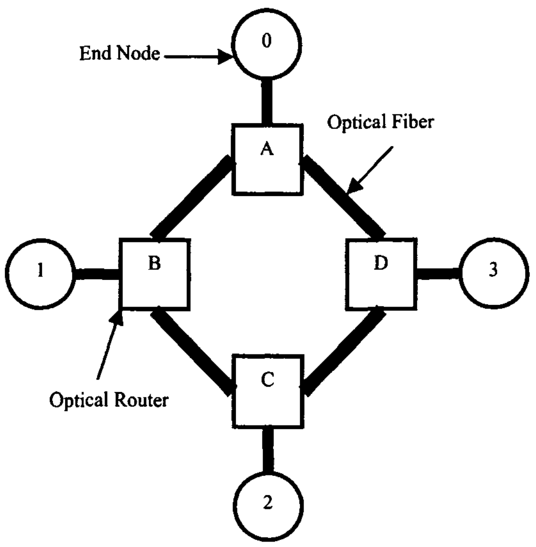

FIGURE 3: PHYSICAL TOPOLOGY

Lightpath: A lightpath is an optical connection from one end-node to another. It is used to carry data in the form of encoded optical signals. Such a lightpath always starts from an end-node, traverses a number of fibers and router nodes and ends in another end-node. Therefore, the lightpath can be visualized as a direct optical connection between two end-nodes. There is an important restriction when determining the route used by lightpaths: Two lightpaths using the same fiber must have different wavelengths. FIGURE 4 illustrates this restriction.

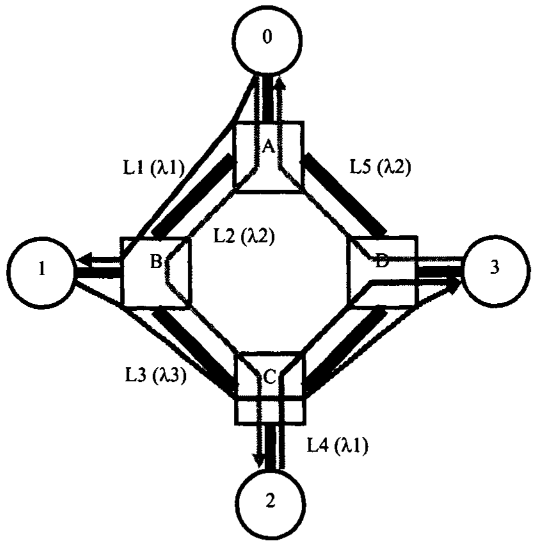

FIGURE 4: LIGHTPATHS WITH ASSIGNED WAVELENGTHS ON A PHYSICAL TOPOLOGY

In this example, three wavelengths $\lambda_1, \lambda_2, \lambda_3$ are available on each optical fibers. Lightpath L_1 and L_2 must have different wavelengths since they both use the same fiber (from A to B). Similarly lightpaths L_3 and L_4 must have different wavelengths since they both use the same fiber (from C to D).

Logical Topology: A logical topology, also called a virtual topology by some researchers, of an optical network is a directed graph which represents how lightpaths connect the end-nodes. The edges in this graph are the lightpaths and the vertices are the end-node of the physical topology. FIGURE 6 is an example of logical topology based on the lightpaths in FIGURE 5.

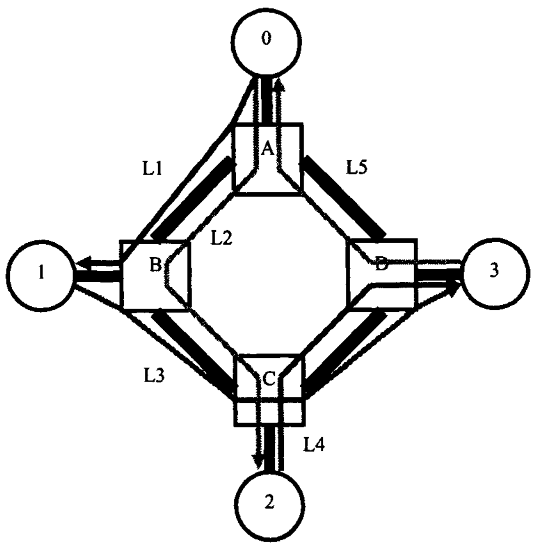

FIGURE 5: LIGHTPATHS

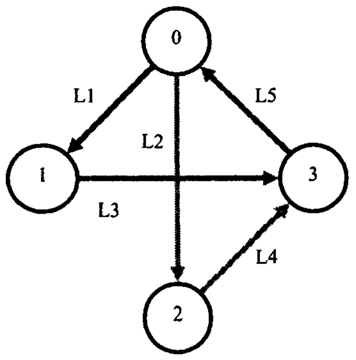

FIGURE 6: LOGICAL TOPOLOGY

2.1.3 Nearly Regular Topologies

Topologies for data communication may be broadly categorized as regular or irregular. The topologies with fixed and simple structural properties are called *regular topologies* such as the de Bruijn graph [23] [27] [29] [30], the Kautz graph [22], the ShuffleNet [6], 2-dimensional mesh [8], the multimesh [3] and the hypercube [1]. The typical property of a regular topology is that it has the same in-degree and out-degree where the in-degree (or out-degree) is defined as the number of incoming (or outgoing) links in the graph. The topologies which have no such structural properties are characterized as irregular (The graph does not has the same in-degree and out-degree). Irregular logical topologies are usually derived by mixed integer linear formulations or by heuristic approaches. In regular topologies, research has shown that de Bruijn graph is better in handling large size networks than the ShuffleNet [26].

A de Bruijn graph with d^k vertices is denoted by $B(d,k)$ where d is in-degree and out-degree, and k is the diameter. The diameter is defined as the longest length of the shortest path between any pair of nodes in the graph. FIGURE 7 is an example of de Bruijn graph with 8 vertices.

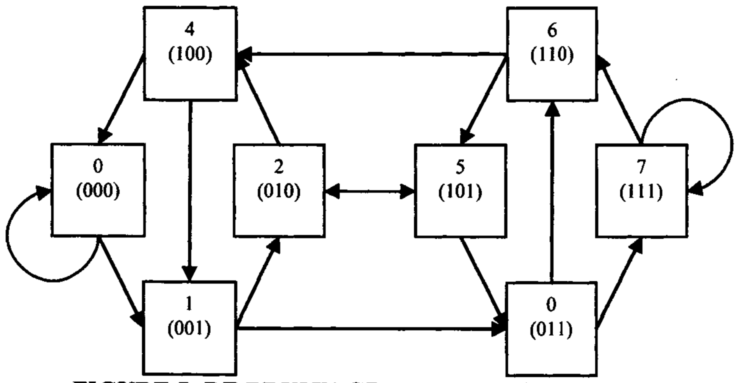

FIGURE 7: DE BRUIJN GRAPH WITH 8 VERTICES

This graph is attractive because of the following reasons.

- The length of the shortest path between any pair of nodes is short (low diameter).
- The routing algorithms are easy to define (simple routing algorithms).
- The routing tables are not involved.

However, de Bruijn graphs are not directly usable for logical topology design since the number of vertices in the graphs must be expressible as d^k. [13] presents a scalable de Bruijn graph to solve the scalability problems while retaining the advantages of the de Bruijn graphs, but it is not strictly regular. FIGURE 8 shows a scalable de Bruijn graph with 7 vertices.

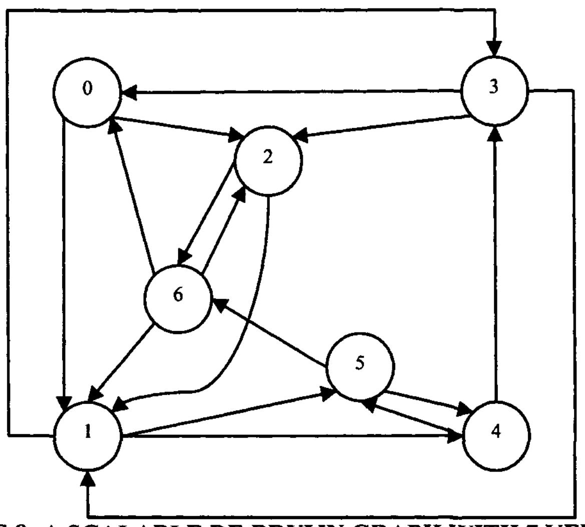

FIGURE 8: A SCALABLE DE BRUIJN GRAPH WITH 7 VERTICES

2.1.4 Traffic Routing

The traffic between the nodes in the network can be represented by an $N \times N$ matrix called the *traffic matrix*, where N is the number of nodes in the networks. Each element in the traffic matrix represents the traffic flow from the source node (row) to destination node (column). The element on the first row and second column in FIGURE 9 represents there are 30 units of data sending from node 1 to node 2, and FIGURE 10 visualizes the traffic matrix in FIGURE 9.

$$T = \lambda_{sd} = \begin{bmatrix} 0 & 30 & 6 & 0 \\ 0 & 0 & 0 & 39 \\ 0 & 0 & 0 & 6 \\ 18 & 0 & 0 & 0 \end{bmatrix}$$

FIGURE 9: TRAFFIC MATRIX OF A 4-NODE NETWORK

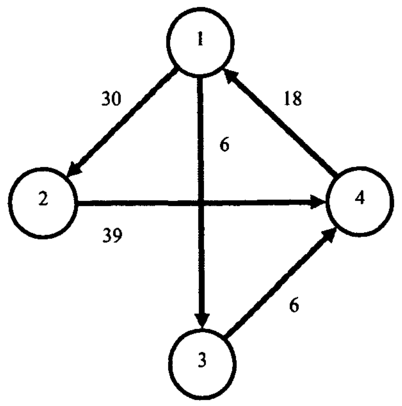

FIGURE 10: TRAFFIC DEMAND OF A 4-NODE NETWORK

The following example explains how traffic routing is done on a given logical topology in FIGURE 11 by sending 50 units of data from node 1 to node 4 where this given topology includes some existing traffic flow.

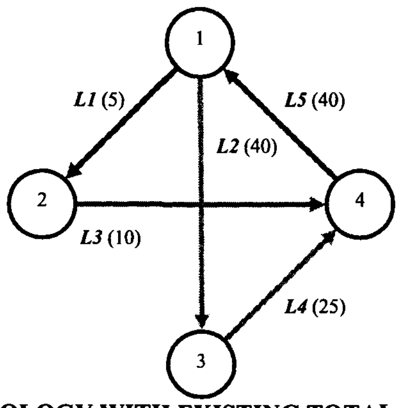

FIGURE 11: LOGICAL TOPOLOGY WITH EXISTING TOTAL PAYLOAD ON EACH EDGE

There are two paths to send the data from node 1 to node 4 as shown in FIGURE 12. One is from node 1 through node 2 and to node 4. The other one is from node 1 through node 3 and then to node 4.

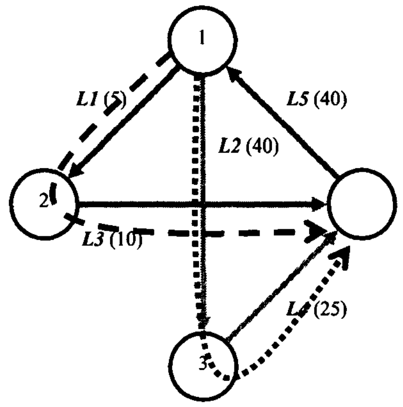

FIGURE 12: TWO PATHS FROM NODE 1 TO NODE 4

There are many ways to separate the data and then send it using these two paths. The following are the two possible solutions, and FIGURE 13 represent the updated amount of data carried by the logical edge corresponding to these solutions.

Solution 1:	*Solution* 2:
25 *units on* $1 \to 2 \to 4$	40 *units on* $1 \to 2 \to 4$
25 *units on* $1 \to 3 \to 4$	10 *units on* $1 \to 3 \to 4$

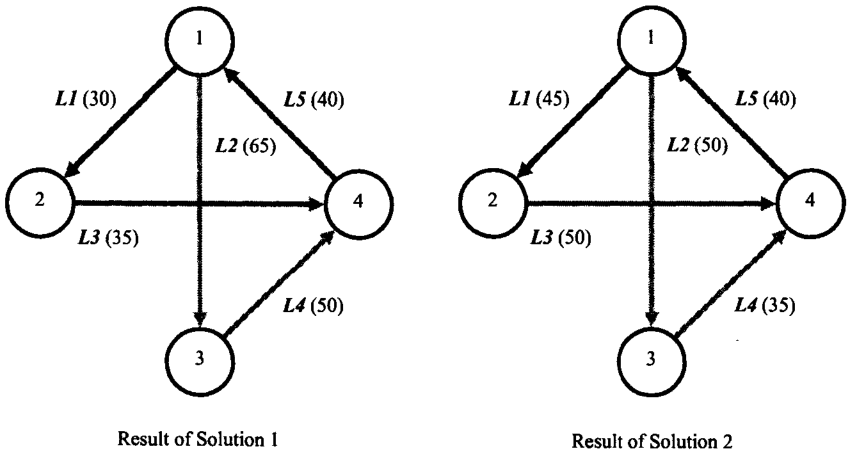

FIGURE 13: UPDATED TOTAL PAYLOAD ON EACH EDGE

As shown in FIGURE 13, the maximum congestion of the solution 1 is 65, and the maximum congestion of the solution 2 is 50. Therefore, solution 2 is better than the solution 1.

2.2 Heuristic Logical Topology Design Algorithm

In order to make the logical topology design problem tractable, the research in [4] proposed the *Heuristic Logical Topology Design Algorithm* (HLDA) to design the logical topology for practical networks. In this heuristic, it is assumed that each of the transmitters and receivers may be tuned to any desired carrier wavelengths. However, as mentioned previously, this thesis ignored the issue of routing and wavelength assignment. Therefore, a simplified version of HLDA is given in FIGURE 14.

```
$HLDA\_NO\_RWA()$
{
Initialization:
    Specified the capacity of the lightpath, $C_{lightpath}$
    Specified the number of the transmitter, $T_x$;
    Specified the number of the receiver, $R_x$;
Select the highest traffic entry from the traffic matrix, $t_{\max} = \max\{t_{sd} \mid t_{sd} \in T\}$;
Repeat
    Compute the number of lightpath starting from node $s$, $CT_x$
    Compute the number of lightpath ending at node $d$, $CR_x$
    If $CT_x \geq T_x$
        Set $t_{sd} = 0$ in the traffic matrix;
        Go to ***Label***;
    If $CR_x \geq R_x$
        Set $t_{sd} = 0$ in the traffic matrix;
        Go to ***Label***;

    Create a logical edge from node $s$ to node $d$;
    Set $t_{sd} = t_{sd} - C_{lightpath}$ *or* 0 which ever is greater in the traffic matrix;
***Label:***
    Select the highest traffic entry from the traffic matrix, $t_{\max} = \max\{t_{sd} \mid t_{sd} \in T\}$;
Until there is no non-zero entry in $T$
}
```

FIGURE 14: HLDA WITHOUT RWA

2.3 Overview of Genetic Algorithms

Genetic algorithms (GA) are randomized search methods based on Evolutionary Theory (Charles Darwin, 1809 – 1882) which may be used to solve optimization problems. Since Holland [7] developed and applied this idea into computer simulations, genetic algorithms have been successfully used in widespread applications in business, scientific and engineering etc.

When simulating genetic algorithms, candidate solutions (a set of possible solutions) to the given problems are represented by a *population* of individuals. Each *individual* is represented by a chromosome. A *chromosome* is defined by an ordered list of parameters; each parameter represents a feature of the individual and is called a *gene.* The value of a gene is called an *allele.*

According to the principles of natural selection and survival of the fittest, genetic algorithms are able to generate solutions to real world problems. The evolution usually starts from a population of randomly generated individuals. The fitness of every individual in the population is evaluated by the fitness function. The algorithms generate a new population based on their fitness. The new population is then used in the next iteration, and each successive population is called a *generation.* Genetic algorithms normally terminate at one of the following two states.

- *Convergence*: A satisfactory fitness level has been reached for the population. In this state, individuals in the population become alike to each other.
- A maximum number of generations have been generated. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

FIGURE 15 outlines the skeleton of genetic algorithms, and FIGURE 16 illustrates the generation procedures of genetic algorithm where P (*T*) is the population in the Tth generation.

```
SGA()
{
    Randomly generate the initial population $P(T)$ where $T = 0$;
    Evaluate $P(T)$;
    Repeat
        Generate offspring population $P(T+1)$ from $P(T)$;
        $T = T + 1$;
        Evaluate $P(T)$;
    Until termination criteria are satisfied
}
```

FIGURE 15: ABSTRACT VIEW OF GENETIC ALGORITHM

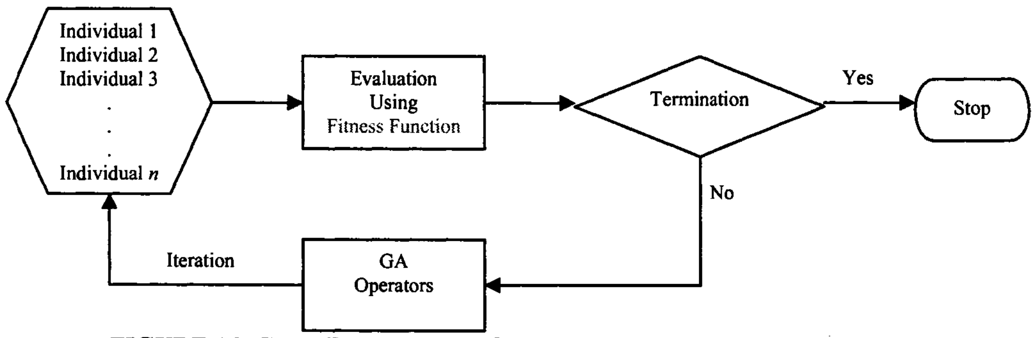

FIGURE 16: GENERATION PROCEDURE OF GENETIC ALGORITHM

In above figures, GA operators usually include the following three operations:

- *Selection (Reproduction)*: To decide which individual will have greater chance to be selected and reproduced for crossover and mutation in the next generation.
- *Crossover*: For individuals to exchange useful information with each other in order to produce better, same or worse offspring in terms of fitness.
- *Mutation*: For individuals to change one or more properties in that individual. In the genetic algorithm, if there is only crossover and no mutation, it is easy to fall into local maximum. But if there is only mutation and no crossover, the process of evolution will be slow.

2.3.1 Simple Genetic Algorithm

Simple Genetic Algorithm (SGA) is the simplest form of genetic algorithm consisting of the following components: data representation, initialization (control variables), evaluation, genetic operators and termination criterion. The following example illustrates how SGA works.

Consider the problem of maximizing the function $f(x) = x^2$, where x is an integer between -31 and 31.

1. Data Representation: The parameter x can be represented as a binary signed integer of length 6 where leftmost bit represents the sign of the binary integer. For examples, positive integer 17 can be represented by 010001, and negative integer 18 can be represented by 110010.
2. Initialization: (control variables): In this step, the values of the control variables need to be defined. These control variables include the probability of crossover, mutation and the size of population. In addition, the initial population (first generation) is randomly generated. The following are the control variables that we used for this problem.
   - Probability of Crossover: $P_c = 100\%$
   - Probability of Mutation: $P_m = 1\%$
   - Population Size: $P_{size} = 4$
   - Initial Population: $P(0) = \{001101, 110001, 100010, 011100\}$
3. Evaluation: Assign the fitness value to each individual in a population using the fitness function where the fitness function for this problem is $f(x) = x^2$.

Individual No.	*Chromosome*	*Signed Integer*	*Fitness Value*	Pr*obability of Selection*
		x	$f(x)=x^2$	$P_s=\frac{f_i}{\sum f}$
1	001101	13	169	$169/862=0.20=20\%$
2	110001	-17	289	$289/862=0.34=34\%$
3	100010	-2	4	$4/862=0.00=0\%$
4	010100	20	400	$400/862=0.46=46\%$
Sum			862	
Average			215.5	
Max			400	

4. GA Operator:

➢ Selection (Reproduction): Calculate the probability of selection for every individual according to their fitness value where the probability of selection is defined by $P_s=\frac{f_i}{\sum f}$. After calculating the probability of selection, copy the individuals from the current generation $G(T)$ to the next generation $G(T+1)$. When selecting the individuals to copy, we use a roulette wheel with slots sized according to each individual's probability of selection. FIGURE 17 is an example of roulette wheel for this problem.

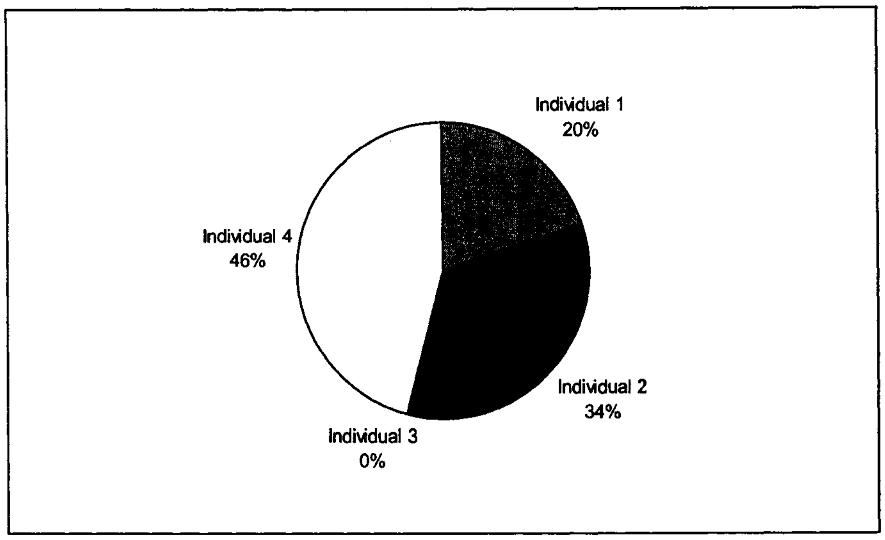

FIGURE 17: ROULETTE WHEEL

The following is one of the possible results after selection & reproduction. Since individual no. 4 in $G(T)$ has greater probability of selection, it has better chance to be selected in $G(T+1)$ more than once. In other word, individual no. 3 in $G(T)$ might not be selected at all in $G(T+1)$ because of its low probability of selection.

$G(T)$ Individual No.	Chromosome		$G(T+1)$ Individual No.	Chromosome
1	001101		1	010100
2	110001	$\xrightarrow{Selection \,\&\, Re\,production}$	2	110001
3	100010		3	010100
4	010100		4	001101

➢ Crossover: Apply crossover on the new generation $G(T+1)$ by picking up two individuals from $G(T+1)$ to exchange their information. Individual no. 1 and 2 in $G(T+1)$ are picked to illustrate the procedure of crossover.

$$\begin{array}{ccc} \textit{Parents} & & \textit{Offspring} \\ 0101|00 & \Rightarrow & 0101|01 \\ 1100|01 & & 1100|00 \end{array}$$

Randomly select a number between 1 to the maximum length of the binary string which is 6. This selected number represents the position of the crossover point. Assume number 4 is selected. The chromosome of the two selected individuals exchanges their genetic information from the crossover point to create the offspring. Thus, 0101 (the first part of chromosome 010100) is combined with 01 (the second part of chromosome 110000) to create offspring 010101. Similarly, 1100 (the first part of chromosome 110001) is combined with 00 (the second part of chromosome 010100) to create 110000.

- Mutation: According to the probability of mutation, a number of individuals in $G(T+1)$ mutate. Assume individual no. 1 in $G(T+1)$ mutates, and number 5 is selected for the mutation point (range from 1 to the maximum length of the binary string).

$$\begin{array}{ccc} \textit{Parents} & & \textit{Offspring} \\ 0\ 1\ 0\ 1\ \underline{\underline{0}}\ 0 & \rightarrow & 0\ 1\ 0\ 1\ \underline{\underline{1}}\ 0 \end{array}$$

In computer simulation, mutation is simply to flip the binary bit from 0 to 1 or 1 to 0. Therefore, 010100 will change to 010110.

5. Termination Criteria: When termination criteria have been archived, the program stops. Otherwise, the program goes to the next iteration with $G(T+1)$. Ideally, chromosome 011111 or chromosome 111111 should be found as the solution to this problem.

2.4 Power of Genetic Algorithms

The power of genetic algorithms comes from the fact that the techniques provide unique flexibility, simplicity and robustness to deal with a wide range of difficult problems including NP-hard problems. Genetic algorithms do not guarantee to find the global optimum solution to a problem, but they are usually good at finding acceptable good solutions within reasonable amount of time. They are different from the traditional optimization and search methods in four ways including data representation, parallel search, black box search and probabilistic transition rules [5].

2.4.1 Data Representation

Unlike the other optimization and search methods, genetic algorithms work with a coding of the parameter set instead of dealing with the objective functions and their parameters directly. In genetic algorithms, the natural parameter set of the optimization problem is required to be coded as a string of finite length over some finite alphabet. As an example, consider a black box with 6 on-off switches in FIGURE 18.

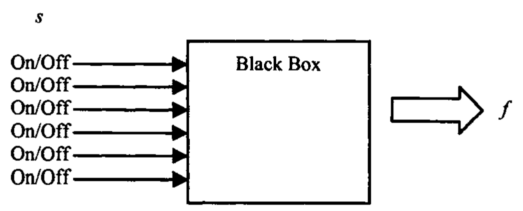

FIGURE 18: A BLACK BOX WITH 5 ON-OFF SWITCHES

For every setting of the six switches s, there is an output signal f. This relationship can be represented by a mathematical function $f = f(s)$ where s represents a setting of the six switches. The objective of this problem is to maximize the value f. In genetic algorithm, s can be coded as a 6-bit binary string where 1 represents on and 0 represents off.

Consider another example that is used in Section 2.3.1 for maximizing the function $f(x) = x^2$. With genetic algorithms, parameter x also can be coded as a 6-bit binary string where 0 and 1 represent other meanings. This problem is not any different from the black box problem in the abstract level except the objective function. Both problems can be solved using the same mechanism in genetic algorithms. Therefore, the advantage of data representation in genetic algorithms is that genetic algorithm can exploit coding similarities in a very general way because of the abstraction of data representation for the problem and operation on the coding level [5].

2.4.2 Parallel Search

Many optimization and search methods start from a single point in the search space and move to the next available one using some transition rules. This kind of point-to-point methods is dangerous since they are easy falling into local maximal area especially in multimodal (many-peaked) search spaces. One characteristic of genetic algorithm is that it searches from a population of points simultaneously (not a single point) and climbs many peaks in parallel. This characteristic generally reduces the chance of finding a false peak.

For an example, consider the black box problem that is used in Section 2.4.1. Other optimization and search methods might start with only one set of switch setting (i.e.

010101) and generate a new trail switch setting using some transition rules until the solution is found. With genetic algorithm, it might randomly generate the initial population of switch setting where we assume the size of population $n = 4$

010101, 101010, 000011, 111100

After the initialization, genetic algorithms generate successful candidates using selection, crossover and mutation until the stop criteria has been reached. By working from a population of well-adapted diversity instead of a single point, genetic algorithms find safety in numbers [5].

2.4.3 Black Box Search

Many optimization and search methods are unable to work properly without auxiliary information. For an example, gradient techniques need information that can be calculated analytically or numerically in order to climb the current peak. Other techniques such as greedy techniques of combinatorial optimization require the information of the tabular parameters.

Compared to the search scheme above, genetic algorithm is a black box search. It can perform effective search on any problem as long as the data can be represented (coded) properly and the objective function is provided to compute payoff (fitness) values that are associated with individual string. In other word, genetic algorithms do not need to be aware the problem domain. This characteristic makes genetic algorithms to be a more canonical method than many other optimization and search methods [5].

2.4.4 Probabilistic Transition Rules

Instead of directly using deterministic transition rules, genetic algorithms use probabilistic transition rules as tools to guide a search toward the improved point in the search space [5]. As discussion in Section 2.3.1, most genetic operators including selection, crossover and mutation are performed based on the probabilities. The use of probabilistic transition rules in genetic algorithms is not as simple as randomized walk. Genetic algorithms can efficiently exploit the historical information to help on searching the new point in the search space with expected improvement since they are based on the Evolutionary Theory.

2.4.5 Comparison from General Perspective

Genetic algorithms have advantages over other search methods based on those four differences discussed in the previous sections. However, certain limitations do exist in genetic algorithms as mentioned previously. Genetic algorithms do not guarantee to find the global optimum nor the proper convergence in arbitrary problems. Generally, a customized scheme would outperform genetic algorithms on specific problems. The major advantage of genetic algorithms is that they can find acceptable good solutions within reasonable amount of time without accessing auxiliary information or being customized specifically for the problems. As a result, genetic algorithms provide flexibility, simplicity and robustness for arbitrary problems. FIGURE 19 taken from [5], illustrates the performance of genetic algorithms compared with other search methods.

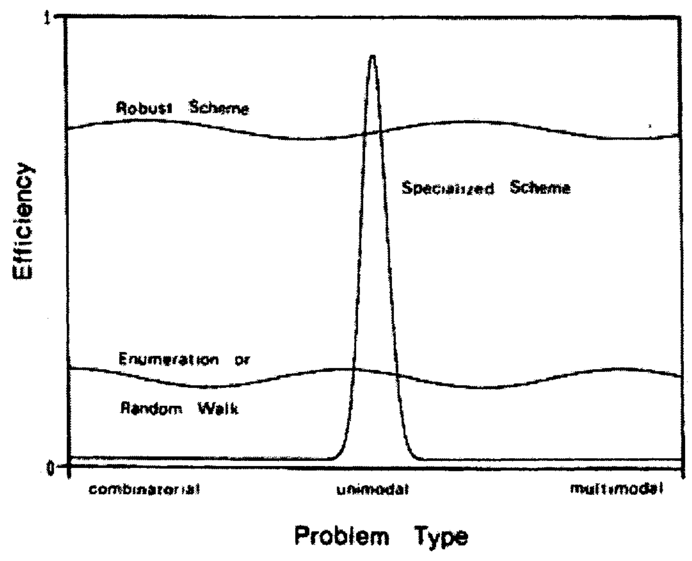

FIGURE 19: GENERAL COMPARISON

Genetic algorithms work well across different problem domains. Some traditional techniques might outperform genetic algorithms, but they only work well in a narrow problem domain. Enumerative schemes and random walks both work less efficiently than the genetic algorithms.

CHAPTER III

DESIGN AND METHODOLOGY

As explained in Section 1.2, determining the optimal logical topology has been proved that it is a NP-complete problem [4], and heuristics such as the HLDA are used to find a logical topology for practical networks [4]. It is also well-known that the routing problem for non-bifurcated traffic grooming is also a NP-complete problem [4]. Other investigators at Windsor have studied the logical topology design and traffic routing on logical topologies using MILP formulations [12]. The objective of this thesis is to study whether the compute-intensive task of logical topology design may be simplified by considering regular topologies. Given a list of requests for communication, this research compares two existing approaches and a new approach in logical topology design to determine whether the regular topology have promise as logical topologies. FIGURE 20 illustrates the major components of work in this research.

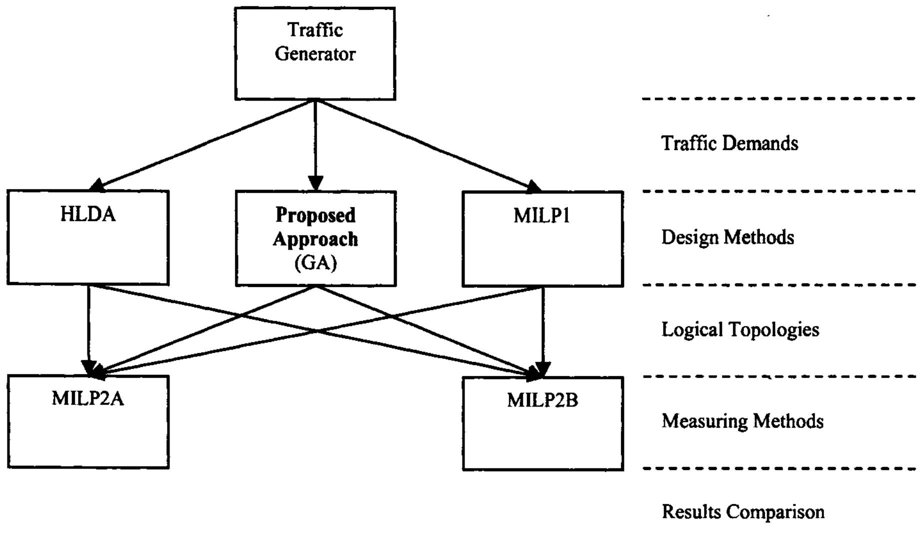

FIGURE 20: MAJOR COMPONENTS OF THE RESEARCH

The traffic generator in FIGURE 20 generates a list of traffic demands to be handled by the network under consideration using a random number generator. For convenience, this list is in the form of a traffic matrix described in Section 2.1.4.

For generating the logical topology, the following approaches have been studied:

- HLDA: A well-known heuristic for generating the logical topology described in Section 2.2.
- MILP1: A Mixed Integer Linear Program formulation developed in [12] to design an optimal logical topology[2].
- Proposed Approach: The genetic algorithm based approach to find the optimum logical topology based on the idea of scalable de Bruijn graph in Section 2.1.3

For traffic routing, the following models have been used after the logical topology is defined.

- Model A: Congestion minimization problem has been discussed in Section 2.1 and solved using MILP2A[3].

2 MILP1 has been described in Appendix A.

- Model B: Traffic maximization problem has been discussed in Section 2.1 and solved using MILP2B[4].

3.1 Proposed Genetic Algorithm

This thesis investigates the use the scalable de Bruijn graph as the target logical topology because of its attractive properties including low diameter, rich interconnection, simple routing scheme and the flexibility with respect to the size of the networks. By selecting this graph, the logical topology design problem is reduced to finding an appropriate mapping between the physical nodes of the network and the logical nodes of the scalable de Bruijn graph which attempts to minimize the maximum congestion of the network. This section introduces a customized genetic algorithm to process the mapping between the physical nodes and logical nodes. The basic structure of the proposed genetic algorithm is based on the simple genetic algorithm [5] and is given in FIGURE 21. The detail of the proposed algorithm will be discussed in the following subsections.

---

[3] MILP2A has been described in Appendix B.
[4] MILP2B has been described in Appendix C.

```
PROPOSED_GA()
{
    Initialization:
        Setup all the control variables;
        Compute the selected "good" paths for all source destination pairs;
        Compute the list of adjacent node for each node
        Randomly generate the initial population $P(T)$ where $T = 0$;
    Evaluate $P(T)$;
    Repeat
        If converge
            Return success;
        Else
            Generate offspring population $P(T+1)$ from $P(T)$:
                Select from $P(T)$ to reproduce $P(T+1)$;
                Apply crossover and/or mutation on $P(T+1)$;
            $T = T + 1$;
            Evaluate $P(T)$;
    Until a maximum number of generations have been generated
    Return failure;
}
```

FIGURE 21: PROPOSED GENETIC ALGORITHM

3.2 Data Representation

In this research, each potential solution to the logical topology design problem is a possible mapping between the physical nodes and the logical nodes. This mapping can be represented by a chromosome[5]. A chromosome is defined by an ordered list where the position in the list (gene) represents the logical node number and the corresponding value (allele) represents the physical node number that the logical node is mapped.

As an example, the following chromosome corresponds to a 7-node network which is also one of the possible mappings corresponding to FIGURE 2.

Logical Node	0	1	2	3	4	5	6
Physical Node	*B*	*D*	*C*	*A*	*F*	*E*	*G*

The first row gives the logical node number, and it is always presented in the same order as shown above. The second row shows the physical node number that the logical node is mapped. This example shows that node $B, D, C, \ldots$ will be assigned logical nodes 0, 1, 2 ... Since the logical topology with 7 nodes using the scalable de Bruijn graph as shown in FIGURE 8 has an edge from node 0 to node 1, there is an edge representing a lightpath between node B and nod D. Since a chromosome is uniquely identified by an ordered list, the mapping between the logical nodes and the physical nodes is simply determined by the position of the physical nodes in the list. Therefore, the list $(B\ D\ C\ A\ F\ E\ G)$

[5] Terminology related to the genetic algorithm has been explained in Section 2.3.

represents the same chromosome as the one given above. In order to avoid confusion, the rest of this thesis uses numbers to represent the logical nodes and alphabets to represent the physical nodes. However, both the logical nodes and physical nodes are represented by integers when actually implementing the chromosome.

3.3 Initialization

During the initialization phase, the following steps are taken.

1. Setup all the control variables, including the population size, the probabilities of crossover and mutation. During the experiments, the control variables are varied depending on the performance of the proposed genetic algorithm.
2. Compute "good" paths for all source destination pairs. This step is needed for the evaluation strategy and will be discussed in Section 3.4.
3. Compute adjacent node list for each node. This step is needed for the crossover strategy discussed in Section 3.6.2.
4. Randomly generate initial population.

Since steps 1 and 4 are straight forward, no further explanation is needed.

As soon as all the required steps are taken in the initialization phase, the algorithm will perform the evaluation on the initial population using the objective function and generate the offspring population until the termination criteria are satisfied which will be discussed in Section 3.5, 3.6 and 3.7.

3.4 Selected "Good" Paths

Since it is not feasible to consider all possible logical paths in a dense graph such as the scalable de Bruijn graph considered here, the approach adopted in this research is to determine a relatively small number of "good" paths and examine only those paths when routing the traffic. A brief overview of the process of selection of good paths is given below. Details are given in Appendix D.

In the optical network research, the shortest path from a source to a destination is usually a good candidate for the route of a lightpath. If the shortest path routing is used, the total number of wavelengths used in setting up the lightpath is the smallest possible. However, the shortest path is not necessarily the optimal choice for the route of a lightpath in the presence of other lightpaths. For an example, if the shortest paths of all lightpaths use the same fiber, all wavelengths on that fiber will be used up quickly and many lightpaths will not be feasible. In other word, if a relatively long route is used for a lightpath, the number of wavelengths used to set up the lightpath and hence the amount of resources devoted to the lightpath becomes large. Therefore, it is desirable to use routes that are "as short as possible" and yet allow RWA for all the lightpaths to be set up.

The process adopted in this research is to generate a list L of paths containing k paths. These paths should be as short as possible and each path should be ideally disjoint from all other paths. The number of edge disjoint paths in a graph between a specified source and a specified destination is limited by the connectivity of the graph [4]. It is not possible to have only edge-disjoint paths in L. The heuristic has been used to find the good paths are given in FIGURE 22.

```
GOOD_PATH()
{
        Initialization:
                Initialize the list of path, $L = \{(shortest\ path)\}$;
                Initialize the counter for the number of path, $i = 0$;
                Specified the maximum number of path, $k$;
                Specified the increase in the length of path, $n$;
        Compute the length of the shortest path, $m$
        Generate a list of path, $P$        // The paths in P have length between m and m+n
        Repeat
                Find a path $x$ in $P$ which is not in $L$ where $x$ has minimum number of
                overlap edges with all the paths exist in $L$;
                Add the path $x$ to $L$;
                $i = i + 1$;
        Until $i = k$
        Return $L$;
}
```

FIGURE 22: SELECTED "GOOD" PATH ALGORITHM

3.5 Evaluation

There are two functions used for the evaluation of a chromosome: the *objective function* and the *fitness function*. The purpose of the objective function is to compute the objective values of individuals. In this thesis, each individual is a permutation – an ordered list of numbers, denoting the logical nodes. Each chromosome, an individual in the genetic algorithm, represents a mapping of the physical node into the logical node number. The objective value for the individual is the congestion for that mapping. The value of the congestion[6] for this individual depends on the routing strategy used in the model. A greedy heuristic is used in our objective function as follows.

The communication with the highest traffic is considered first. There are only up to *k* selected "good" paths (as discussed in Section 3.4) to be considered when routing each communication. The path with the least traffic and having the shortest length is considered first. This is a greedy heuristic so that any earlier routing decisions for the communication will not be changed and hence will affect later decisions.

In order to reduce the complexity of the problem, this thesis has not considered the communication between all source destination pairs. It has only considered the communication having a traffic value higher than a predefined threshold value. The value of the threshold is usually set to be between 10% and 30% of the maximal traffic. This thesis has set the threshold value to be 10% of the maximal traffic. FIGURE 23 gives a brief outline of the objective function in the proposed algorithm.

[6] Congestion is already defined in Section 1.1.

```
$OBJECTIVE\_FUNCTION()$
{
$P$ = selected "good" path for all source destination pairs;
Initialize the value of congestion, $congestion = 0$;
Initialize the traffic load of each link, $\lambda_{ij} = 0$;
Setup the threshold, $threshold = T_m$;
Select the highest traffic entry from the traffic matrix, $t_{\max} = \max\{t_{sd} \mid t_{sd} \in T\}$;
Repeat
    Select a path $P_{sd}^k$ in $P$;
    // $P_{sd}^k$ is a path from s to d with the least traffic and having shortest length
    For each link $E_{ij}$ in $P_{sd}^k$
        $\lambda_{ij} = \lambda_{ij} + t_{sd}$;
    Set $t_{sd} = -t_{sd}$ in the traffic matrix;
    Select the highest traffic entry from the traffic matrix, $t_{\max} = \max\{t_{sd} \mid t_{sd} \in T\}$;
Until $t_{\max} < T_m$
$congestion = \max\{\lambda_{ij}\}$;
}
```

FIGURE 23: OBJECTIVE FUNCTION OF THE PROPOSED GA

The following example illustrates how objective function has been computed. FIGURE 24 gives a chromosome C_1 corresponding to a 7-node network in FIGURE 25 and TABLE 1 shows input traffic demand to be considered.

Logical Node	0	1	2	3	4	5	6
Physical Node	*B*	*D*	*C*	*A*	*F*	*E*	*G*

FIGURE 24: CHROMOSOME C_1

	A	*B*	*C*	*D*	*E*	*F*	*G*
A	0	3	30	12	24	3	33
B	30	0	27	42	6	27	36
C	45	36	0	27	24	3	15
D	36	3	3	0	21	21	15
E	9	9	48	27	0	6	48
F	6	6	21	30	48	0	12
G	36	15	24	24	39	48	0

TABLE 1: INPUT TRAFFIC DEMAND

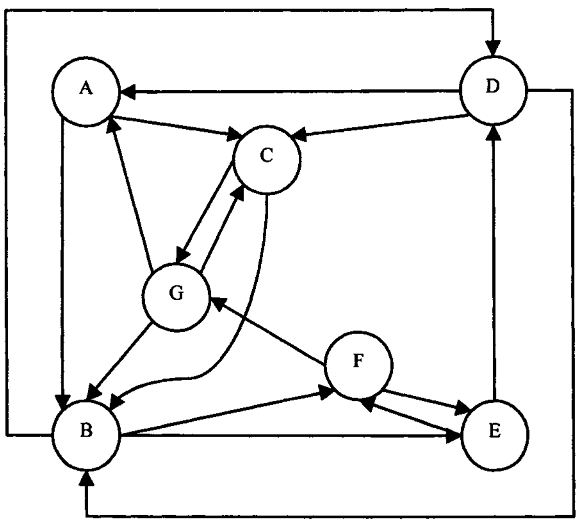

FIGURE 25: TARGET TOPOLOGY

1. Initialization:
   - P = selected "good" path for all source destination pairs
   - Initialize the traffic load of each link, $\lambda_{ij} = 0$
   - Initialize the congestion, *congestion* = 0
   - Set the threshold, $T_m = 30$
2. Select the highest traffic entry from the traffic matrix, $t_{\max} = t_{EC} = 48$

3. Check if $t_{\max} < T_m$

    - Since $t_{\max} > T_m$, the process continues.

4. Select a path P_{sd}^k in P:

    - $P = \{P_{EC}^1, P_{EC}^2, P_{EC}^3\}$ where $P_{EC}^1 = E \to D \to C$, $P_{EC}^2 = E \to F \to G \to C$ and $P_{EC}^3 = E \to D \to A \to C$
    - Since the maximum traffic load on the link in $P_{EC}^1, P_{EC}^2, P_{EC}^3$ are all equal to 0, select the shortest path P_{EC}^1

5. Update the traffic load on each link E_{ij} in the path P_{EC}^1

    - Set $E_{ED} = E_{ED} + t_{EC} = 0 + 48 = 48$
    - Set $E_{DC} = E_{DC} + t_{EC} = 0 + 48 = 48$

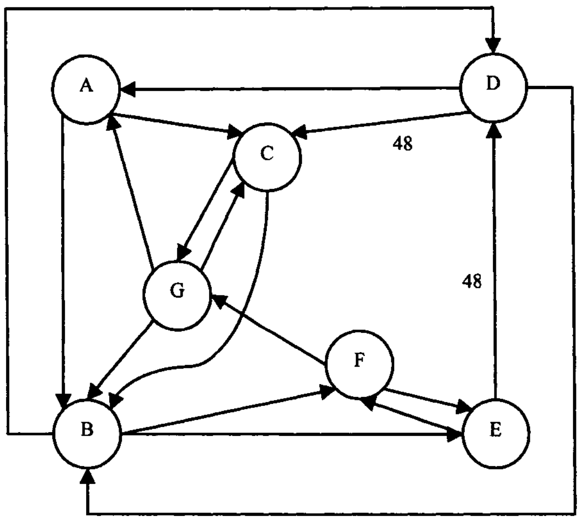

FIGURE 26: TARGET TOPOLOGY AFTER PROCESSING t_{EC}

6. Set $t_{EC} = -t_{EC}$ in the traffic matrix to indicate the traffic entry t_{EC} has been processed.

- Since $t_{EC} = 48$, set $t_{EC} = -48$

7. Repeat from the step 2
   - Select the highest traffic entry from the traffic matrix, $t_{\max} = t_{EG} = 48$
8. Check if $t_{\max} < T_m$
   - Since $t_{\max} > T_m$, the process continues.
9. Select a path P_{sd}^k in P:
   - $P = \{P_{EG}^1, P_{EG}^2, P_{EG}^3\}$ where $P_{EG}^1 = E \to F \to G$, $P_{EG}^2 = E \to D \to C \to G$ and $P_{EG}^3 = E \to D \to A \to C \to G$
   - Since the maximum traffic load on the link in $P_{EG}^1, P_{EG}^2, P_{EG}^3$ corresponds to 0, 96 and 48, select the path P_{EG}^1
10. Update the traffic load on each link E_{ij} in the path P_{EG}^1
    - Set $E_{EF} = E_{EF} + t_{EG} = 0 + 48 = 48$
    - Set $E_{FG} = E_{FG} + t_{EG} = 0 + 48 = 48$
11. Set $t_{EG} = -t_{EG}$ in the traffic matrix to indicate the traffic entry t_{EG} has been processed.
    - Since $t_{EG} = 48$, set $t_{EG} = -48$
12. Repeat from the step 2 until all the traffic entries in the traffic matrix which are greater than the threshold value have been processed.
13. Set congestion (objective value) equal to the maximum value of the traffic load on the link of the target topology.

The other function used for the evaluation is the fitness function. The objective of the proposed genetic algorithm is to find a mapping between the physical nodes of the network and the logical nodes of the scalable de Bruijn graph which has the ***minimum*** congestion value. However, a general genetic algorithm typically attempts to find a mapping which has the ***maximum*** fitness value. Therefore, a function is required to convert the congestion value (objective value) to a non-negative fitness value. FIGURE 27 shows the fitness function that converts the congestion value of the chromosome x to the corresponding fitness value where the variable $MaxCongestion$ is the maximum congestion so far during the execution.

```
if $Congestion(x) > MaxCongestion$
    $Fitness(x) = 0$
else
    $Fitness(x) = MaxCongestion - Congestion(x)$
```

FIGURE 27: FITNESS FUNCTION OF THE PROPOSED GA

Initially $MaxCongestion$ is set to be $zero$. After computing the congestion value of whole population for each generation, $MaxCongestion$ is set to be the highest congestion found so far considering all generations. This step of updating $MaxCongestion$ is done before converting the congestion values to fitness values using the Fitness Function.

3.6 Genetic Operators

GA operation usually includes the following three operators.

- selection & reproduction
- crossover

- mutation

This section will explain how to incorporate these three operators into the proposed algorithm.

3.6.1 Selection (Reproduction)

The selection strategy in the proposed genetic algorithm is the combination of *proportional selection* and *elitist selection* [25]. In proportional selection, the probability of being selected for an individual is equal to the fitness value of the individual divided by the sum of the fitness value in a population. It is the same as the selection strategy used in SGA. For elitist selection, a certain number of individuals with the highest fitness value in every generation automatically enter the next generation $G(T+1)$ without going through the selection (reproduction) and mating procedures. This approach is also known as pre-selection [5]. With pre-selection procedure, the individual(s) having the best objective value in the current generation maintain unchanged in the next generation. In other world, the maximum fitness value will never decrease from one generation to the next.

3.6.2 Crossover

Since the objective of the proposed algorithm is to minimize the congestion of a network, two nodes should be placed as close to each other as possible if they have large traffic. Therefore, the most important structure that needs to be preserved is a node x and its adjacent nodes. This collection of a node x and its adjacent nodes will be termed a *cluster* with center x. FIGURE 28 shows an example of a cluster with center E.

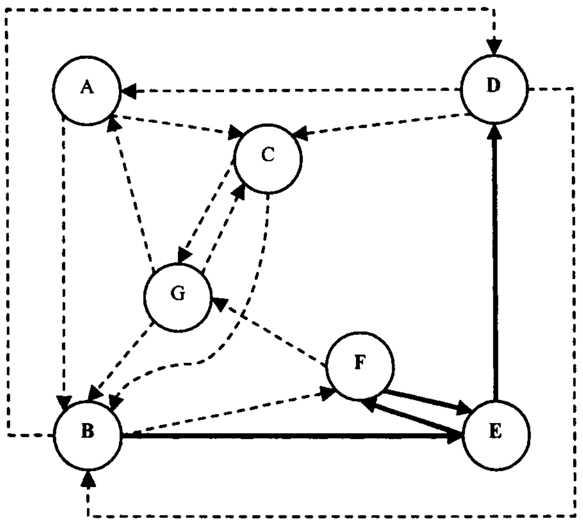

FIGURE 28: AN EXAMPLE OF CLUSTER

The proposed algorithm uses *cluster crossover* strategy to perform the crossover operation. In cluster crossover strategy, it tries to preserve the good cluster in a chromosome. The following two chromosomes C_1 and C_2 are used to illustrate the cluster crossover.

Chromosome C_1

Logical Node	0	1	2	3	4	5	6
Physical Node	*B*	*D*	*C*	*A*	*F*	*E*	*G*

Chromosome C_2

Logical Node	0	1	2	3	4	5	6
Physical Node	*A*	*D*	*C*	*E*	*F*	*B*	*G*

1. Assume the node E is randomly selected as the center of the cluster.
2. Protect all the nodes are adjacent to the node E. Therefore, the temporary offspring of chromosome C_1 and C_2 would be as the following.

Temporary Offspring of Chromosome C_1

Logical Node	0	1	2	3	4	5	6
Physical Node	*B*	*D*	?	?	*F*	*E*	?

Temporary Offspring of Chromosome C_2

Logical Node	0	1	2	3	4	5	6
Physical Node	?	*D*	?	*E*	*F*	*B*	?

3. Fill the blanks marked by "?" in the temporary offspring of chromosome C_1 using the physical node numbers in chromosome C_2 that do not appear in the temporary offspring of chromosome C_1. When doing so, these blanks will be replaced by physical node numbers in the same order as those appearing in C_2. In other words, the first question mark will be replaced by A, the second by B and the third by C. Therefore, the offspring of chromosome C_1 would be as the following.

Offspring of Chromosome C_1

Logical Node	0	1	2	3	4	5	6
Physical Node	*B*	*D*	*A*	*C*	*F*	*E*	*G*

4. Apply the similar strategy as step 3 on the temporary offspring of chromosome C_2 to create offspring of chromosome C_2 as the following.

Offspring of Chromosome C_2

Logical Node	0	1	2	3	4	5	6
Physical Node	*C*	*D*	*A*	*E*	*F*	*B*	*G*

3.6.3 Mutation

The mutation operator in the proposed algorithm is a swap operator. It simply switches two genes in a chromosome where these two genes are randomly selected. The following chromosome, corresponding to a 7-node network, is used as an example to illustrate the mutation procedure.

Logical Node	0	$\underline{\underline{1}}$	2	3	$\underline{\underline{4}}$	5	6
Physical Node	*B*	$\underline{\underline{D}}$	*C*	*A*	$\underline{\underline{F}}$	*E*	*G*

Assume the genes in position 1 and 4 are selected, the new chromosome after a mutation will be as the following.

Logical Node	0	$\underline{\underline{1}}$	2	3	$\underline{\underline{4}}$	5	6
Physical Node	*B*	$\underline{\underline{F}}$	*C*	*A*	$\underline{\underline{D}}$	*E*	*G*

The probability of mutation in the proposed algorithm is set to be 1%. Since the purpose of mutation is to prevent the algorithm falling into local optimal, the algorithm increases mutation rate to 51% (suggested by [5]) for one generation when the algorithm suspects early convergence.

3.7 Termination Criteria

The proposed genetic algorithm will terminate when one of the following two situations occurs:

- Convergence: The chromosomes of individuals in a population become identical or similar to each other.
- Fixed Number of Generations: A maximum number of generations have been generated.

When the algorithm converges, the average fitness value $(AvgFitness)$ for a population must be at least 90% of the best fitness value $(MaxFitness)$, and it is called *90%-rule*. However, it does not necessary mean that the chromosome become similar to each other for multimodal problems since there might be a possibility that half of the population are one individual with best fitness value and the other half are other individual with best fitness value if there are more than one individuals with best fitness value. According to 90%-rule, the algorithm is considered to be convergent in this case, but the chromosomes of individuals in a population are not identical or similar. Although the mapping problem presented in this thesis is a multimodal problem, convergence actually doe not mean anything as long as a good solution can be found. However, the principle of the genetic algorithm, the survival of the fitness, dictates that the genetic algorithm should terminate when converge. Therefore, the proposed algorithm includes it as one of the stopping criteria.

Early convergence is another situation should be considered and prevented. This may result in a situation where the program may not find a good solution. In order to minimize the possibility of this situation, the program has to go through a certain number of generations $(MinGen)$ before it terminates. If the algorithm converges before undergoing a certain number of generations, it increases the probability of mutation (P_m) in one generation. If the program does not converge after another certain number of generation $(MaxGen)$, the program stops and is considered to be a failure. The outline of the termination criteria used in the proposed algorithm is given in FIGURE 29.

```
$if(Gen < MinGen) and (AvgFitness \geq 0.9 \times MaxFitness)$
    $P_m = P_m + 0.5$
$if(MinGen \leq Gen \leq MaxGen) and (AvgFitness \geq 0.9 \times MaxFitness)$
    $return\ succeeds$
$if(Gen > MaxGen)$
    $return\ failure$
```

FIGURE 29: TERMINATION CRITERIA

CHAPTER IV

ANALYSIS OF RESULTS

This chapter investigates the proposed approach for logical topology design using a series of experiments. Since the proposed approach uses regular topology, coupled with the genetic algorithm, the investigation can be separated into the following two phases:

- Phase I: Investigate the robustness of the proposed genetic algorithm.
- Phase II: Investigate the performance of the regular topology in logical topology design.

These experiments have been described in the subsections below.

4.1 Testing Environment and Control Variables

The proposed genetic algorithm has been implemented using the *C* language and has been tested on the *luna* server. TABLE 2 gives the system specification of the *luna* server. Unless specified, all test results presented in this chapter represent the average of 5 runs.

OS Name	Solaris 9
OS Manufacture	Sun Microsystems
System Name	luna.cs.uwindsor.ca
System Manufacture	Sun Microsystems
System Model	V880
Processor	8 CPUs
Total Physical Memory	16 GB
Ethernet	Gigabit Ethernet

TABLE 2: SYSTEM SPECIFICATION

As mentioned in Chapter 3, there are a number of important control variables in the proposed genetic algorithm. These variables include the population size, the probabilities of crossover and mutation. However, this thesis does not investigate how these variables affect the performance of the proposed genetic algorithm. The effect of modifying these variables have been discussed in [15] for a similar problem. This investigation focuses on comparing the proposed approach to the two other existing approaches for logical topology design. TABLE 3 gives the list of the values of the control variables for the proposed genetic algorithm. These values were mainly determined from the experiments reported in [15].

Threshold, $T_m = 0.1$
Probability of Pre-selection, $P_{pre-s} = 0.04$
Probability of Crossover, $P_c = 1.0$
Probability of Mutation, $P_m = 0.01$
Probability of Mutation (Early Converge), $P_m = 0.51$
Minimum Number of Iteration, $MinGen = 200$
Maximum Number of Iteration, $MaxGen = 1000$
Selected Path (Increased Length, No of Path) = (2, 3)

TABLE 3: A LIST OF CONTROL VARIABLES

4.2 Robustness of the Proposed Genetic Algorithm

In order to investigate the robustness of the proposed genetic algorithm, the resulting topologies of the proposed genetic algorithm have been compared with the topologies generated by the exhaustive search (in terms of the solution time and the minimum congestion of the resulting topologies). Since there are $N!$ different mappings between the physical nodes of the network and the logical nodes of the scalable de Bruijn graph, it is not feasible to find the optimal topologies in the practical sized networks. Therefore, this experiment has been conducted for the small sized network only. In this phase of the experiments, scalable de Bruijn graphs with 6 and 7 vertices have been tested. Other parameters for this experiment are listed in TABLE 3.

Additionally, this experiment also tests the genetic algorithm with different population sizes to determine the suitable population size for the later experiments. TABLE 4**Error! Reference source not found.** gives the average solution time of the compared algorithms, and the average minimum congestion of the resulting topologies.

Network Size (No. of Node)	Exhaustive Search		Genetic Algorithm			
			Population Size (50)		Population Size (100)	
	Solution Time	Minimum Congestion	Solution Time	Minimum Congestion	Solution Time	Minimum Congestion
6	0.053	114.6	0.806	115.2	1.671	115.2
7	0.626	133.2	1.324	133.8	2.719	134.4

TABLE 4: ROBUSTNESS – AVERAGE SOLUTION TIME (SEC) AND AVERAGE MINIMUM CONGESTION

As seen from the above tables, the genetic algorithm is able to generate the logical topologies which are competitive with the topologies found by exhaustive search. Besides the quality of the solutions, the solution time of the genetic algorithm is also acceptable when comparing to the exhaustive search.

4.3 Performance of the Proposed Approach

Ideally, it is desirable to measure the performance of the proposed approach on the basis of the global optimum. However, it is also very difficult to determine the optimal logical topologies for most networks. Therefore, HLDA and MILP formulation are implemented and the results of these two approaches have been used as benchmarks. This thesis used the following three factors to evaluate the performance of the proposed approach.

- The time to find the logical topologies.
- The minimum congestion of the resulting topologies.
- The number of traffic request that can be handled.

TABLE 5 is the average solution time of the approaches; TABLE 6 gives the average minimum congestion of the resulting topologies generated by the approaches, and TABLE 7 shows the average number of traffic request that can be handled by the logical topologies generated by the approaches.

Compared Approaches	Network Size (No. of Node)				
	7	10	12	14	21
HLDA	0.000001	0.000001	0.000001	0.000001	0.010000
MILP1	2.804000	7200.000000	7200.000000	7200.000000	-
Proposed GA	2.780000	10.530000	14.518000	25.972000	238.848000

TABLE 5: PERFORMANCE – AVERAGE SOLUTION TIME (SEC)

Compared Methods	Network Size (No. of Node)				
	7	10	12	14	21
HLDA	121.50	307.60	300.60	362.40	573.00
MILP1	118.20	259.20	264.00	372.00	-
Proposed GA	135.00	312.60	317.40	408.00	608.40

TABLE 6: PERFORMANCE – AVERAGE MINIMUM CONGESTION

Compared Methods	Network Size (No. of Node)				
	7	10	12	14	21
HLDA	1014.00	1960.20	2901.60	4753.80	8974.80
MILP1	1050.60	2095.80	3006.60	4721.40	-
Proposed GA	948.00	1835.20	2562.60	4646.40	8596.20

TABLE 7: PERFORMANCE – AVERAGE NUMBER OF TRAFFIC REQUEST THAT CAN BE HANDLED

Although the MILP-based approach generates better logical topologies in most networks, it takes more time to find the solutions and fails to find one in the larger networks. HLDA takes the least amount of time to find solutions. The proposed method is able to find a "good" solution in the reasonable amount time. FIGURE 30, FIGURE 31 and FIGURE 32 present the data to the readers in a better way which are with respect to TABLE 5, TABLE 6 and TABLE 7.

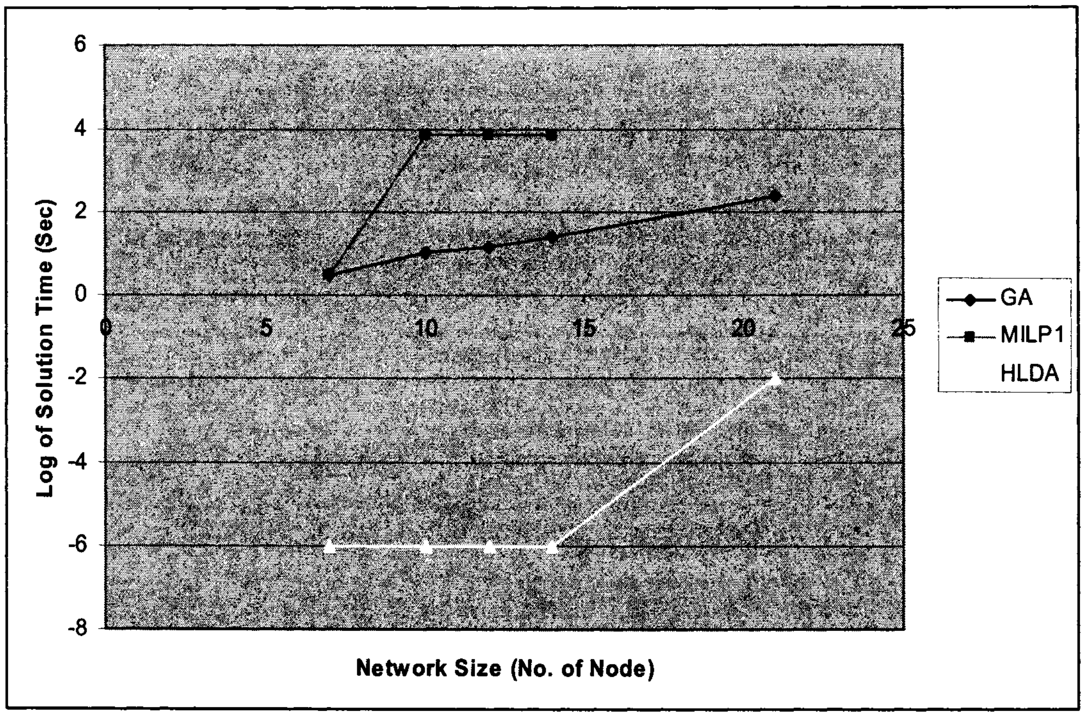

FIGURE 30: PERFORMANCE – LOG OF AVERAGE SOLUTION TIME

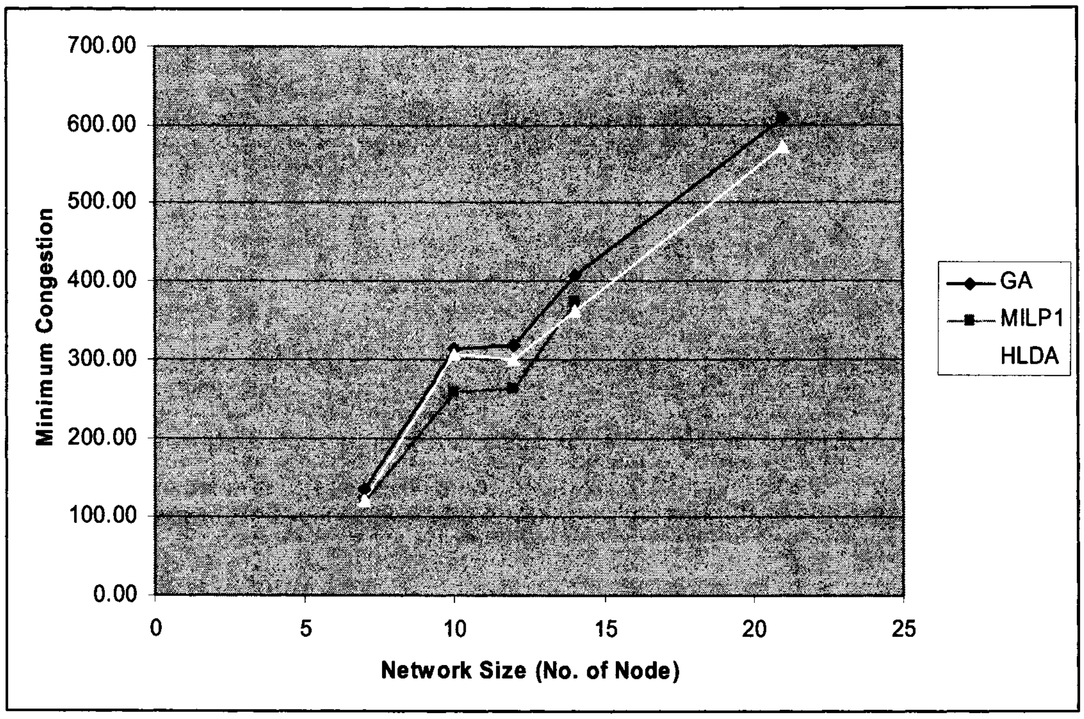

FIGURE 31: PERFORMANCE – AVERAGE MINIMUM CONGESTION

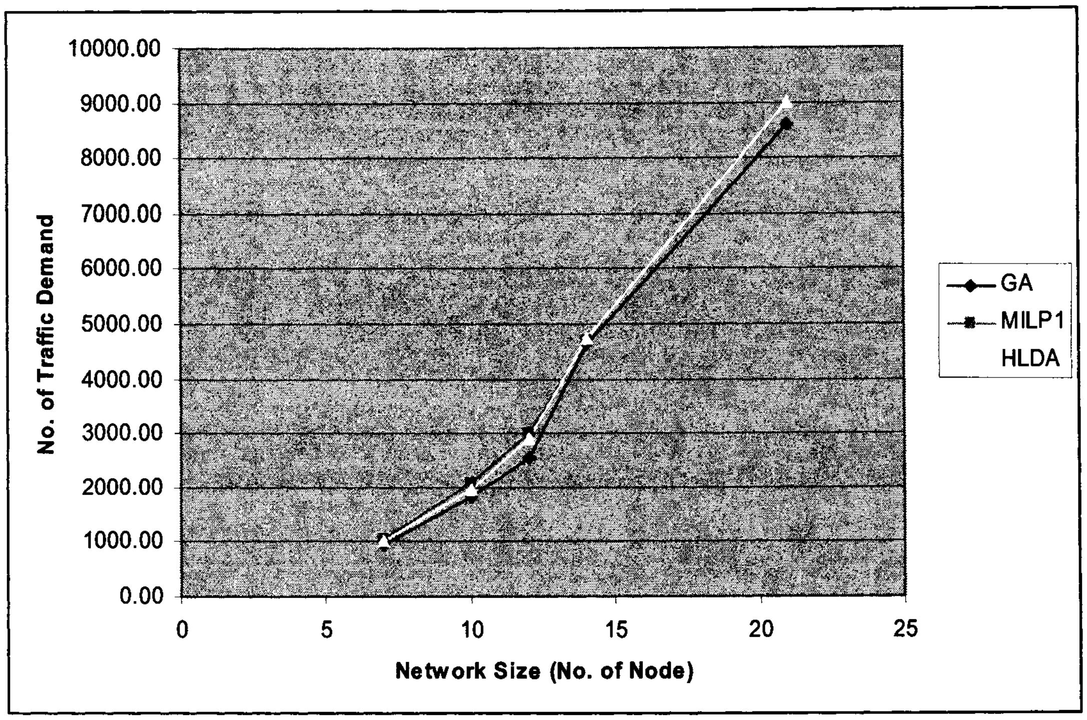

FIGURE 32: PERFORMANCE – AVERAGE NUMBER OF TRAFFIC REQUEST CAN BE HANDLED

In this series of experiments, HLDA performs very well. However, in some cases, the topologies generated by HLDA are failed when applying the models that are used to determine the minimum congestion. TABLE 8 shows how many times (in percentage) HLDA failed when applying MILP2A and MILP2B.

Applied Model	Network Size (No. of Node)				
	7	10	12	14	21
MILP2A	20%	0%	0%	0%	60%
MILP2B	0%	0%	0%	0%	0%

TABLE 8: FAILING RATE OF HLDA

4.4 Statistical Analysis

This thesis uses the 95% *confidence interval* (C.I.) to analyze the statistical significance of the experimental results. This interval is specified by an upper bound (U) and a lower bound (L). In other word, this confidence interval means that if large scale experiments are carried out, 95% of the time, the mean will lie within the interval U and L. The confidence interval is calculated using Equation (1).

$$95\%\ C.I. = \bar{x} \pm 1.96 \times \frac{s}{\sqrt{n}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (1)$$

Where,

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \text{, the mean of the samples} \quad \ldots\ldots\ldots\ldots \quad (2)$$

$$s = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2} \text{, the standard deviation} \quad \ldots\ldots\ldots\ldots \quad (3)$$

$$n = 1,2,3\ldots\infty \text{, the size of the samples} \quad \ldots\ldots\ldots\ldots \quad (4)$$

TABLE 9 and TABLE 10 both represent the confidence interval for the minimum congestion of the logical topologies, and TABLE 11 shows the confidence interval for the number of traffic request that can be handled by the logical topologies.

Network Size (No. of Node)	**Proposed Approach (Population Size = 50) Vs. Exhaust Search**		**Proposed Approach (Population Size = 100 Vs. Exhaust Search**	
	Lower Bound	**Upper Bound**	**Lower Bound**	**Upper Bound**
6	1.00	1.01	1.00	1.02
7	1.00	1.01	1.00	1.02

TABLE 9: 95% CONFIDENCE INTERVAL FOR THE MINIMUM CONGESTION OF THE LOGICAL TOPOLOGIES

Network Size (No. of Node)	**Proposed Approach Vs. HLDA**		**Proposed Approach Vs. MILP1**	
	Lower Bound	**Upper Bound**	**Lower Bound**	**Upper Bound**
7	1.05	1.19	1.05	1.21
10	0.88	1.21	1.15	1.27
12	1.00	1.12	1.15	1.26
14	1.08	1.17	1.00	1.21
21	1.02	1.08	-	-

TABLE 10: 95% CONFIDENCE INTERVAL FOR THE MINIMUM CONGESTION OF THE LOGICAL TOPOLOGIES

Network Size (No. of Node)	**Proposed Approach Vs. HLDA**		**Proposed Approach Vs. MILP1**	
	Lower Bound	**Upper Bound**	**Lower Bound**	**Upper Bound**
7	0.52	1.37	0.54	1.28
10	0.88	1.00	0.89	0.93
12	0.86	0.90	0.84	0.87
14	0.97	0.98	0.97	1.00
21	0.93	0.97	-	-

TABLE 11: 95% CONFIDENCE INTERVAL FOR THE NUMBER OF TRAFFIC REQUEST THAT CAN BE HANDLED BY THE LOGICAL TOPOLOGIES

As an example, in TABLE 11, for a 7-node network and the proposed approach vs. HLDA, the 95% confidence interval is between 0.52 and 1.37 for the proposed approach vs. MILP formulation. This means that if many experiments are carried out the mean of the results is computed, in 95% of the experiments, the ratio (in terms of the number of traffic request can be handled by the topologies) will not be less than 0.52 and will not be more than 1.37.

CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

This chapter summarizes the contribution of this research and discusses the conclusions that haven been reached. It also outlines some directions for future work.

5.1 Conclusions

This thesis investigates the logical topology design problem in optical WDM networks. It ignores the issues of routing and wavelength assignment and has focused on designing the optimal logical topologies of the networks. In order to solve the problem in a reasonable amount of time, a regular topology was used, in conjunction with a customized genetic algorithm, to find a “good” logical topology. The objective of the genetic algorithm was to find an appropriate mapping between the physical nodes of the network and the logical nodes of the scalable de Bruijn graph to minimize the congestion, the traffic on the logical edge carrying the maximum traffic. This research also looks at a number of approaches to design the logical topologies to determine whether regular topologies have promise.

During this research, a series of experiments was conducted under restricted conditions. The experimental result shows that the proposed genetic algorithm is robust since the minimum congestion of the topologies generated by the proposed genetic algorithm is only slightly different compared to the exhaustive search. The result also shows that the proposed approach generates the logical topologies whose performance, in terms of the minimum congestion of the network and the number of traffic request that can be handled, is somewhat less than the other approaches investigated in this thesis. However, the proposed approach is able to find a "good" logical topology within a reasonable amount of time.

5.2 Future Work

This section outlines some potential research directions related to logical topology design for further investigation.

- Regular Topologies: Although the result of the proposed approach is not better than other approaches, the result of exhaustive search is also not better than other approaches. Therefore the Genetic algorithm reported in this thesis is good. The limitation of the approach is likely to be the choice of the regular topology. This investigation only looked at the scalable de Bruijn graphs. It is possible that some other regular graph is better than the scalable de Bruijn graph.
- Dynamic Traffic: Another potential study in the area of logical topology design is the application of regular topologies for handling dynamic traffic. In the research reported in this thesis, the requests are specified using a traffic matrix so that the traffic requests are known in advance and are fixed in time. In the case where the requests for communication are arriving at random, and the previous history of calls has no relationship to future requests for communication, having a low diameter logical topology has distinct advantages since it guarantees that there is at least one logical paths that has relatively few edges. This scenario is more representative of internet traffic. It is possible that the application of regular

topology may be defended better for dynamic traffic. This needs to be investigated in detail in the future.

APPENDICES

APPENDIX A

MILP1

This appendix contains the details of the MILP1 formulation that is taken from [12]. The objective of MILP1 formulation is to design a logical topology which attempts to minimize the total weighted hop count corresponding to the logical paths used to route each traffic request. In other word, this approach tries to minimize the total amount of optical resources used to accommodate a given set of traffic requests. The formulation of MILP1 and the notations[7] used for this formulation are given below.

Notation used

- V_L : Set of end-nodes in the network.
- P : Set of potential lightpaths to be considered for inclusion in the network.
- Q : Set of all traffic requests.
- T_X^i : Number of transmitters at end-node *i*.
- R_X^i : Number of receivers at end-node *i*.

[7] Some of the notations described here will be used in Appendix B and C.

- t_q : Data communication rate for traffic request q using OC-n notation.
- s_q : Source node of traffic request q.
- d_q : Destination node of traffic request q.
- $o(p)$: Originating node of lightpath.
- $l(p)$: Terminating node of lightpath.
- u_p : Capacity of lightpath p using OC-n notation.
- $\lambda_{\max}$: The maximum amount of traffic, using the OC-n notation, on any lightpath.
- b_p : Binary variable defined as follows:

$$b_p = \begin{cases} 1 & \textit{if lightpath } p \in P \textit{ is selected to constitute an edge in the logical topology}, \\ 0 & \textit{otherwise.} \end{cases}$$

- $f_{p,q}$: Binary variable defined as follows:

$$f_{p,q} = \begin{cases} 1 & \textit{if request } q \textit{ is routed over lightpath } p, \\ 0 & \textit{otherwise.} \end{cases}$$

- y_q : Binary variable defined as follows:

$$y_q = \begin{cases} 1 & \textit{if request } q \textit{ is not blocked}, \\ 0 & \textit{otherwise.} \end{cases}$$

Formulation for MILP1

Objective:

$$\textbf{Minimize} \sum_{p \in P} \sum_{q \in Q} f_{p,q} t_q \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (1)$$

Subject to:

a) Flow constraints:

$$\sum_{p:o(p)=i} f_{p,q} - \sum_{p:l(p)=i} f_{p,q} = \begin{cases} 1 & \text{if } i = s_q, \\ -1 & \text{if } i = d_q, \\ 0 & \text{otherwise.} \end{cases} \quad \ldots\ldots\ldots\ldots\ldots \quad (2)$$

Constraint (2) has to be repeated for all $q \in Q$ and for all $i \in V_L$.

b) Capacity constraint for each lightpath:

$$\sum_{q \in Q} f_{p,q} t_q \le u_p . b_p, \quad \forall p \in P \quad \ldots\ldots\ldots\ldots\ldots \quad (3)$$

c) Transceiver constraints at each node:

$$\sum_{p:o(p)=i} b_p \le T_X^i, \quad \forall i \in V_L \quad \ldots\ldots\ldots\ldots\ldots \quad (4)$$

$$\sum_{p:l(p)=i} b_p \le R_X^i, \quad \forall i \in V_L \quad \ldots\ldots\ldots\ldots\ldots \quad (5)$$

APPENDIX B

MILP2A

This appendix contains the details of the MILP2A formulation that is taken from [12]. This formulation is for the traffic routing problem. The objective of MILP2A formulation is to minimize the maximum load on any given lightpath where the logical topology is already specified. In other word, it minimizes the congestion of the network. The formulation of MILP2A and the notation used for this formulation are given below.

Notation used

Refer to Appendix A.

Formulation for MILP2A

Objective:

$$\textbf{Minimize } \lambda_{\max} \quad \dots\dots\dots\dots \quad (6)$$

Subject to:

a) Flow constraints:

$$\sum_{p:o(p)=i} f_{p,q} - \sum_{p:l(p)=i} f_{p,q} = \begin{cases} 1 & if\ i = s_q, \\ -1 & if\ i = d_q, \\ 0 & otherwise. \end{cases} \quad \dots\dots\dots\dots \quad (7)$$

b) Compute total demand on a lightpath:

$$\sum_q f_{p,q} t_q \le \lambda_{\max}, \quad \forall p \in E_L \quad \dots\dots\dots\dots \quad (8)$$

c) Capacity constraint for each lightpath:

$$\lambda_{\max} \le u_p, \quad \forall p \in E_L \quad \dots\dots\dots\dots \quad (9)$$

APPENDIX C

MILP2B

This appendix contains the details of the MILP2B formulation that is taken from [12]. This formulation is for the traffic routing problem. The objective of MILP2B formulation is to maximize the weighted sum of requests that may be handled by the network where the logical topology is already specified. The formulation of MILP2B and the notation used for this formulation are given below.

Notation used

Refer to Appendix A.

Formulation for MILP2B

Objective:

$$\textbf{Maximize} \sum_{q} y_q \cdot t_q \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (10)$$

Subject to:

d) Flow constraints:

$$\sum_{p:o(p)=i} f_{p,q} - \sum_{p:l(p)=i} f_{p,q} = \begin{cases} y_q & \textit{if } i = s_{q,} \\ -y_q & \textit{if } i = d_{q,} \\ 0 & \textit{otherwise.} \end{cases} \qquad \ldots\ldots\ldots\ldots \quad (11)$$

e) Blocked request constraints:

$$f_{p,q} \le y_q, \quad \forall p \in P \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (12)$$

f) Capacity constraint for each lightpath:

$$\sum_{q} f_{p,q} \cdot t_q \le u_p \quad \forall p \in E_L \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (13)$$

APPENDIX D

Selected "Good" Path

This appendix contains the details of the procedure used to find "good" paths from source s to destination d. This uses the following tasks:

1. Task 1 is to find the shortest path for the specified source-destination pair (s, d). Let m be the length of this path.
2. Task 2 is to find a set P of all paths from s to d having length between m and $m + x$ where x is a specified constant.
3. Task 3 is the selected k paths from s to d.

These tasks have been described below.

Task 1:

This task is entirely based on a well-known search algorithm called *Breadth First Search*. Breadth First Search is a simple search strategy. In this strategy, the root node is expanded first then all the nodes generated by the root node are expanded next, and then their successors, and so on. It can be implemented with a FIFO-queue function. In the other words, the newly generated states are put at the end of the queue. Breadth First Search uses a search graph to represent a problem space. In the search graph, nodes (vertices) represent search states and arcs (edges) represent operators that can be used to reach the other state. A search tree is a special case of a search graph with no cycles. Breadth First Search has time complexity of $O(b^d)$ and space complexity of $O(b^d)$ where b is branching or average branching factor and d is depth of tree. TABLE 12 gives the general description of the shortest path algorithm.

```
SHORTEST_PATH (logical_topology, source_node, destination_node)
{
    ENQUEUE (STATE (source_node)) to state_list
    while (state_list is not empty) do
        STATE (x) = DEQUEUE (state_list)
        if x is destination_node then
            sol_state = PREPARE_SOL (STATE (x))
            return sol_state
        else
            for each $n \in$ Adjacent Nodes of (x)
                // Check if there is a "loop" in the logical topology
                if n have not been visited in the current path
                    ENQUEUE (STATE (n)) to state_list
                end if
            end for
        end if
    end while
    return failure
}
```

TABLE 12: SHORTEST PATH ALGORITHM

Task 2:

The algorithm for this task, selected "good" path algorithm, is a customized version of the shortest path algorithm. The difference between the selected "good" path algorithm and the shortest path algorithm is that this algorithm does not only search for the shortest path. It searches the paths that have lengths up to $m + x$ where m is the length of the shortest path and x is the increased length of the path. TABLE 13 gives the general description of the selected "good" path algorithm.

```
ALL_PATH_OF_LENGTH_BETWEEN_MN (logical_topology, source_node,
destination_node, increased_length, input_path)
{
    // The Increase in the Length of Path
    l = increased_length
    if input_path is not empty then
        m = length of input_path
        ENQUEUE (STATE (source_node)) to state_list
        while (state_list is not empty and path_length is less or equal to $m + l$) do
            STATE (x) = DEQUEUE (state_list)
            if x is destination_node then
            sol_state = PREPARE_SOL (STATE (x))
            else
                for each $n \in$ Adjacent Nodes of (x)
                    // Check if there is a "loop" in the logical topology
                    if n have not been visited in the current path
                        ENQUEUE (STATE (n)) to state_list
                    end if
                end for
            end if
        end while
    else
        return failure
    end if
}
```

TABLE 13: SELECTED "GOOD" PATH ALGORITHM

Task 3:

The selected *k* “good” path algorithm is based on the selected “good” path algorithm. The modification of this algorithm is to restrict the number of the paths by the user input *k*.

REFERENCES

[1] L. Bhuyan and D. Agrawal. Generalized hypercube and hyperbus structures for a computer network. *IEEE Transactions on Computers*, C-33(4): 323-333, April 1984.

[2] F. Busetti. Genetic algorithms overview. Documentation available at *http://www.geocities.com/francorbusetti/gaweb.pdf*

[3] D. Das, M. De and B. Sinha. A new network topology with multiple meshes. IEEE Transactions on Computers, 48(5): 536-551, May 1999.

[4] R. Dutta and G. Rouskas. A survey of virtual topology design algorithms for wavelength routed optical networks. *Optical Networks Magazine*, 1(1): 73-89, January 2000.

[5] D. Goldberg. *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley Publishing Company, 1989.

[6] M. Hluchyj and M. Karol. Shuffle Net: an application of generalized perfect shuffles to multihop lightwave networks. *Journal of Lightwave Technology*, 9(10): 1386-1397, October 1991.

[7] J. Holland. Adaption in Neural and Artificial Systems. *University of Michigan Press*, 1975.

[8] K. Hwang and F. Briggs. *Computer architecture and parallel processing*. McGraw-Hill Companies, 1984.

[9] J. Iness, S. Banerjee, and B. Mukherjee. GEMNET: a generalized, shuffle-exchange-based, regular, scalable, modular, multihop, WDM lightwave network *IEEE/ACM Transactions on Networking*, 3(4): 470-476, August 1995.

[10] M. Ilyas and H. Mouftah. *The Handbook of Optical Communication Networks*. CRC Press, 2003.

[11] A. Jaekel, A. Bari and S. Bandyopadhyay. Strategies for traffic grooming over logical topologies. *ISPA*, 2007.

[12] A. Jaekel, A. Bari, Y. Chen and S. Bandyopadhyay. New techniques for efficient traffic grooming in WDM mesh networks. *ICCCN*, 2007.

[13] A. Jaekel, S. Bandyopadhyay, S. Roychoudhury and A. Sengupta. A scalable logical topology for optical networks. *Journal of High Speed Networks*, 11(2): 79-87, September 2002.

[14] R. Krishnaswamy and K. Sivarajan. Design of logical topologies: a linear formulation for wavelength-routed optical networks with no wavelength changers. *IEEE/ACM Transactions on Networking*, 9(2): 186-198, April 2001.

[15] Z. Liu. Mapping physical topology with logical topology using genetic algorithm. M.Sc. Thesis, University of Windsor, June 2001.

[16] Z. Liu, A. Jaekel, and S. Bandyopadhyay. A genetic algorithm for optimization of logical topologies in optical networks. *IEEE IPDPS*, pages: 202-209, April 2002.

[17] K. Man, K. Tang and S. Kwong. Genetic algorithms: concepts and applications. *IEEE Transactions on Industrial Electronics*, 43(5): 519-534, October 1996.

[18] B. Mukherjee. WDM-based local lightwave networks part I: Single-hop systems. *IEEE Network*, 6(3): 12-27, May 1992.

[19] B. Mukherjee. WDM-based local lightwave networks part II: Multihop systems. *IEEE Network*, 6(4): 20-32, July 1992.

[20] B. Mukherjee. *Optical Communication Networks*. McGraw-Hill, 1997.

[21] B. Mukherjee. WDM optical communication networks: progress and challenges. *IEEE Journal on Selected Areas in Communications*, 18(10): 1810-1824, October 2000.

[22] G. Panchapakesan and A. Sengupta. On a lightwave network topology using Kautz digraphs. *IEEE Transactions on Computers*, 48(10): 1131–1137, October 1999.

[23] D. Pradhan and S. Reddy. A fault-tolerant communication architecture for distributed systems. *IEEE Transactions on Computers*, C-31(9): 863-870, September 1982.

[24] R. Ramaswami and K. Sivarajan. Design of logical topologies for wavelength-routed optical networks. *IEEE Journal on Selected Areas in Communications*, 14(5): 840-851, June 1996.

[25] Z. Shi, J. Cui, D. Tao and Y.Zhou. Comparison of steady state and elitist selection genetic algorithms. *2004 International Conference on Intelligent Mechatronics and Automation*, pages 495-499, August 2004.

[26] K. Sivarajan and R. Ramaswami. Multihop lightwave networks based on de Bruijn graphs. *IEEE INFOCOM*, volume 3, pages 1001-1011, April 1991.

[27] K. Sivarajan and R. Ramaswami. Lightwave networks based on de Bruijn graphs. *IEEE/ACM Transactions on Networking*, 2(1): 70-79, February 1994.

[28] M. Srinivas and L. Patnaik. Genetic algorithms: a survey. *Computer*, 27(6): 17-26, June 1994.

[29] M. Sridhar and C. Raghavendra. Fault-tolerant networks based on the de Bruijn graph. IEEE Transactions on Computers, 40(10): 1167-1174, October 1991.

[30] A. Sengupta, A. Sen and S. Bandyopadhyay. Fault-tolerant distributed system design. *IEEE Transactions on Circuits and Systems*, 35(2): 168-172, February 1988.

VITA AUCTORIS

Name:	Chun-Hsien Vic Ho
Year of Birth:	1979
Place of Birth:	Kaohsiung, Taiwan

Education:

Master of Science in Computer Science (2004 – 2007)

University of Windsor, Windsor, ON

Bachelor of Computer Science (2002 – 2003)

University of Windsor, Windsor, ON

Bachelor of Applied Science in Electrical Engineering (1998 – 2003)

University of Windsor, Windsor, ON

Related Experience:

Solution Developer (2004 – 2005)

Faculty of Arts and Social Sciences, University of Windsor, Windsor, ON