2006

Logical topology design for WDM networks using tabu search.

Ashutosh Sood

University of Windsor

Logical Topology Design for WDM Networks using Tabu Search

by

Ashutosh Sood

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

Windsor, Ontario, Canada

Your file Votre référence
ISBN: 978-0-494-17036-6
Our file Notre référence
ISBN: 978-0-494-17036-6

ABSTRACT

In this thesis the problem of Logical Topology Design for WDM Networks has been considered. Optical networks form the backbone of our communication needs, so logical topology design forms a critical part of network design. The physical topology consists of nodes and fiber optic links. Lightpaths are set up on physical topology, which represent a optical connection between two end nodes. The maximum total traffic on a logical link gives us the congestion of the network. The tabu search meta-heuristic opens interesting avenue to expedite finding the optimal logical topology. For a given physical topology, and traffic pattern our objective is to optimize logical topology, using tabu search so as to minimize the network congestion.

To my parents and my wife

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincere gratitude to my supervisor, Dr. Subir Bandyopadhyay and Dr. Yash Aneja for their continuous encouragement and intensive support throughout my graduate studies. This work could not be achieved without their astute guidance and constant encouragement. I would also like to thank Dr. Arunita Jaekel for her valuable suggestions. Also I would like to thank all the committee members, Dr. Scott Goodwin, Dr. Fazle Baki and Dr. Akshai Aggarwal for their valuable time and comments.

Finally I would like to thank my parents, my wife, my brothers and my friends. Their love and care have always been with me during these years. Their trust and encouragement pulled me through hard times.

Last but not least, I would like to thank everybody for offering the help for this thesis work.

Table of Contents

List of Tables

List of Diagrams

CHAPTER 1

Introduction

Optical networking and *Wavelength Division Multiplexing* (WDM) have emerged as the technology of choice for efficiently exploiting the large bandwidth offered by optical fibers. In WDM networks it is possible to combine the data from different users and send it on a single *optical fiber*. Each fiber allows multiple signals on distinct wavelengths for transmission. The ability of WDM networks to send multiple data streams on different wavelengths allows it to make use of the available high bandwidth. A lot of research has been carried out in various areas of optical networking and WDM networks.

1.1 Fundamentals of WDM networks

In an optical network the sources or the destinations of data transmissions are known as *end-nodes*. Another important component of optical networks is the *optical router*. Each router has a number of incoming fibers and a number of outgoing fibers. A router can route an incoming data stream to any outgoing fiber. The *physical network topology* is conveniently depicted as a graph G where the end-nodes and the routers are nodes of the graph G and each fiber, defining a connection from an end-node or an optical router to another end-node or an optical router is an edge in G. A *lightpath* is an optical connection from one end-node to another that is used to carry data in the form of encoded optical signals. A directed graph representation showing the lightpaths connecting one end-node to another is the *logical topology* of the network. A path through a logical topology is known as a *logical path* [Su 06]. Given a logical topology, routing determines which logical paths are used to communicate data for a given (source, destination) pair and how much data is carried by each selected logical path. The *traffic requirement* for a (source, destination) pair is defined as the amount of date to be routed from the source end-node to the destination end-node. An important metric for a logical topology is the *congestion* for the logical topology, defined as the traffic on the logical edge carrying the maximum traffic.

1.2 The Problem Investigated

The optimal logical topology design problem in WDM networks is as follows:

> *Given the traffic requirements between each end-node in the network, the logical topology design problem is to find the logical topology such that the congestion in the network is as low as possible.*

In this thesis the problem of optimal logical topology design using the tabu search is investigated. *Tabu search* is a meta-heuristic search procedure [GL 97] used to guide other lower level heuristics to escape the trap of local optima. In a variety of problem settings, tabu search has found solutions superior to the best previously obtained by alternative methods [Gl 90]. Tabu search procedure makes a systematic use of the memory, keeping track of not only the local information but also information related to the search process. Both short-term and long-term memory is employed to guide the search process. This has been explained in much more detail in Chapter 2.

1.3 Motivation for this Investigation

It is possible to specify the optimal logical topology design, using the standard MILP (mixed integer linear programming) formulation. A solution using the MILP makes it possible to obtain an exact solution for the problem. But the MILP solution involves a large number of binary variables (integer variables whose values are restricted to 0 or 1), and is expensive to solve even for small networks. The alternative of using heuristics is efficient in terms of time, but in a solution using a heuristic, the quality of the solution is unknown.

The optimal logical topology design problem may be simplified into two sub-problems:

- The logical topology design and
- Determining an optimal routing over the logical topology

In this thesis, it is proposed to solve the logical topology design problem using tabu search procedure in order to obtain a near optimal logical topology. The idea is that, when properly designed, tactical and strategic moves of a tabu search is known to be effective in overcoming the well-known problem of getting trapped in a local optimum. Even when the logical topology is known, routing over the logical topology is known to be a time consuming process. This study expedites the process of determining the optimal routing over a logical topology by using the algorithm proposed by Yumei [Lu 04] which gives an approximate solution in a time efficient manner with known approximation bounds.

1.4 Thesis Outline

In this thesis, a tabu search approach to solve the problem of logical topology design has been presented. Two different approaches for assigning tabu status to the moves have been considered. Also the use of two different strategic moves has been employed to escape the trap of local optimality.

In chapter 2 a review of literature on WDM networks, logical topology design problems, and tabu search has been presented. In chapter 3 the algorithm to solve the problem of logical topology design has been described in detail. The chapter 4 describes how the algorithm has been implemented and the experimental results obtained have been enumerated. Of the various approaches experimented the approach using tabu type of “deleting a logical edge or adding a logical edge” and strategic move of “deleting edge with maximum traffic and adding another logical edge” with tabu tenure of 7 for 10 or 14 node networks is found to be most promising.

The chapter 5 concludes the thesis with a critical analysis and suggestions for the future work.

CHAPTER 2

Review of Literature

The exponential growth of Internet has fundamentally changed the way the computer and telecommunication industry is understood. One of the major factors is the relentless increase in the bandwidth requirement for the networks. This again is fuelled by the ever increasing base of Internet users. At the same time, businesses today rely on high speed networks to conduct their business. Practical realization of universal, high-speed wide area networks has been possible largely due to advances in optical networking technology. An optical network provides numerous benefits; foremost among them are high bandwidth, low electromagnetic interference, better security and low crosstalk.

2.1 All-Optical Networks

In optical networks the technology at the physical layer is that of fiber-optic cable. All-optical networks refer to the class of networks where the information path between source node and destination node remains entirely optical [Gr 91]. In this type of network there is no optical-to-electrical conversion of data along the path traversed. Such a network offers protocol transparency, apart from its high speed of operation [Mu 97]. All-optical networks offer the distinct advantage of high bit-rate up to 50 tera-bits per second using a single fiber. One technique for accessing the huge bandwidth available in an optical fiber is Wavelength Division Multiplexing (WDM) [Mu 97].

2.2 Components of WDM Network

2.2.1 Optical Fiber

Optical Fibers are essentially very thin glass cylinders or filaments, which carry signals in the form of light. The optical fiber is made of silica core which has a refractive index of μ_1. The core is enveloped by silica cladding of a lower refractive index μ_2. The refractive Index of a material is the ratio of speed of light in vacuum to speed of light in the medium (in this case glass core or cladding). Optical fibers have evolved over the years in a variety of ways to accommodate both the changing requirements of customer community and the technological challenges that emerged as the demand for bandwidth climbed. Today it is theoretically possible to send 50 tera-bits per second using single fiber. With such high bandwidth availability, optical fiber is a natural choice to build high speed networks.

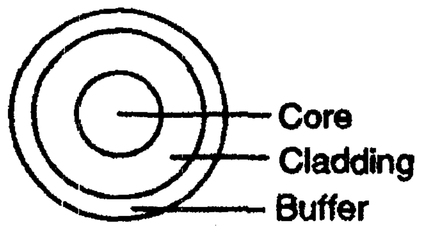


Figure 2.2.1.1 Optical fiber Cross-section

In Figure 2.2.2.1 above at the centre of the fiber is glass filled medium called core. The core is surrounded by cladding also made of glass. The glass cladding is surrounded by a protective material known as buffer.

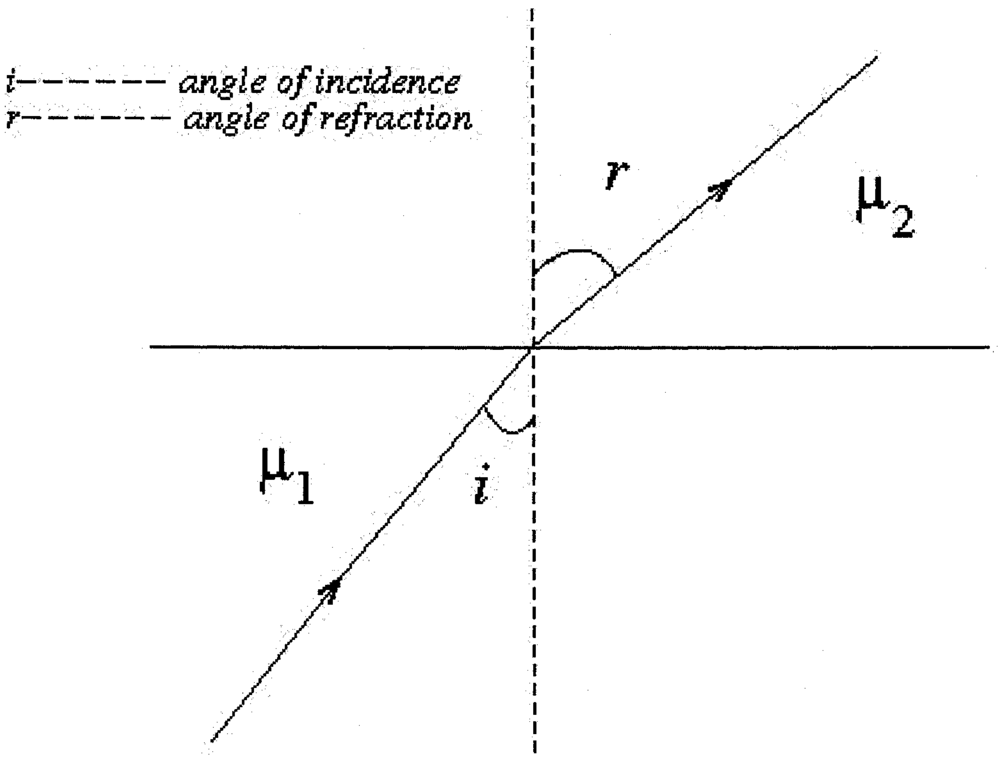


Figure 2.2.1.2 Refraction

As light passes from a medium with refractive index μ_1 to a medium with a lower refractive index μ_2, there is a change in speed of light and it bends as a consequence, the degree by which it bends is determined by two factors:

1) The difference of refractive index between the two media and

2) The angle at which light strikes the medium (angle of incidence).

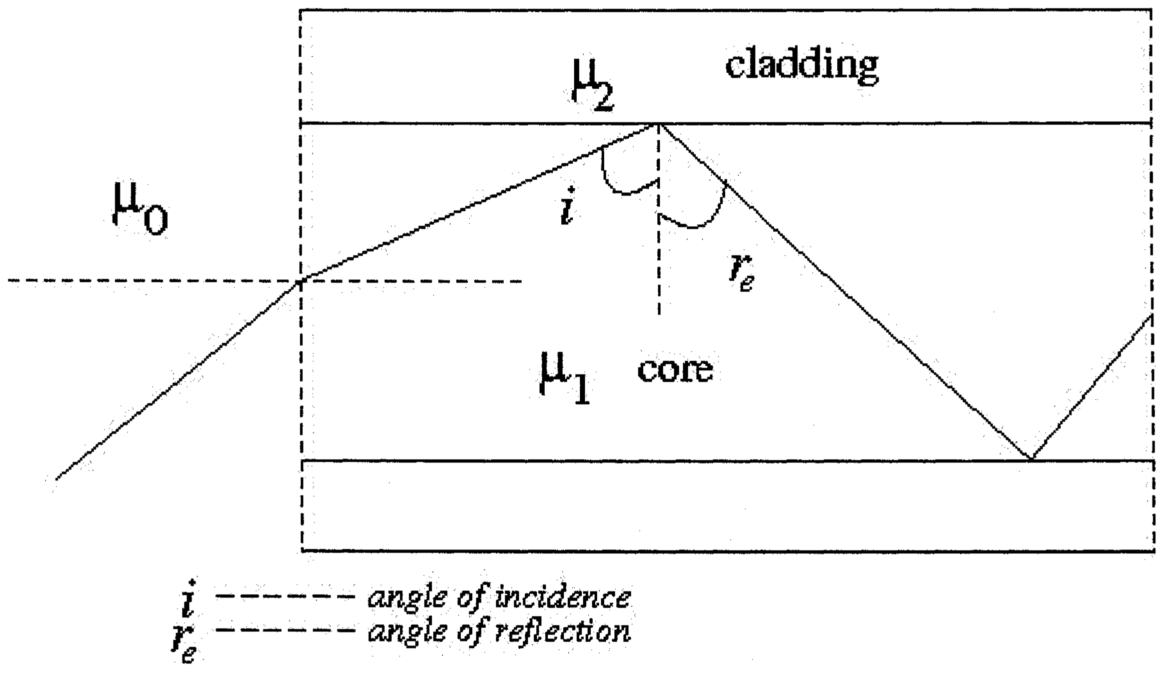

μ_2
cladding
μ_0
i
r_e
μ_1 core
i ------ angle of incidence
r_e ------ angle of reflection

Figure 2.2.1.3 Total Internal Reflection

If the angle of incidence is steep enough (greater than the critical angle $\sin^{-1} \mu_2/\mu_1$), the light instead of going out of the medium having a higher refractive index is reflected back into the medium, this phenomenon known as total internal reflection, forms the basis of optical transmission.

2.2.2 Wavelength Division Multiplexing

Wavelength Division Multiplexing (WDM) technology uses multiple optical signals on the same fiber. The use of WDM allows the utilization of large bandwidth inherent in the optical fiber [DR00]. The vast optical bandwidth of a fiber is carved up into smaller-capacity channels [BMS 94]. In WDM networks, the available bandwidth of the fiber can be visualized as a set of channels.

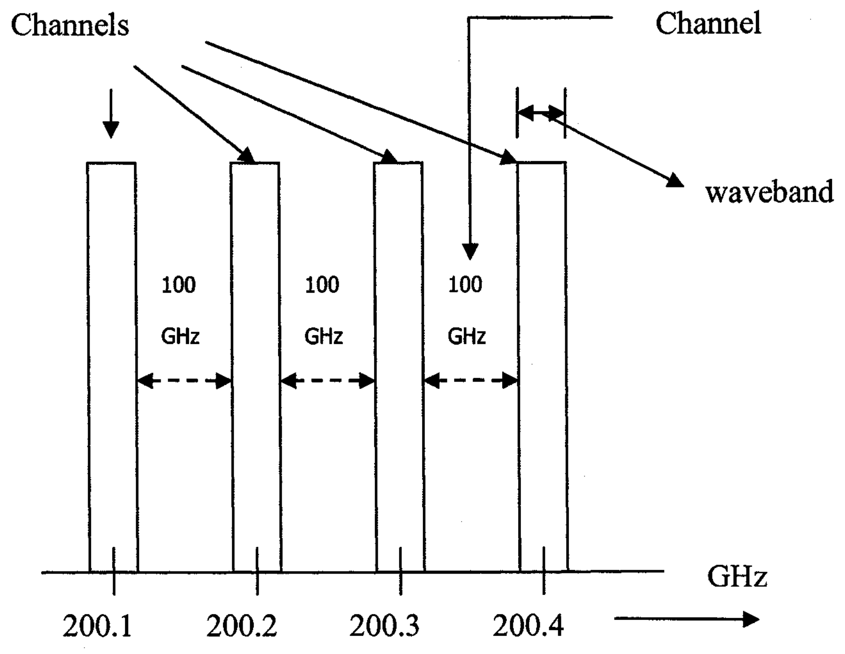
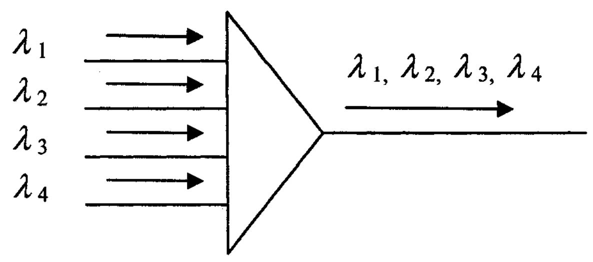


Figure 2.2.2.1 Channel Spacing

The bandwidth is split into multiple channels, each channel having its own waveband, a range of wavelengths. A single fiber thus is capable of carrying different data streams independently on different channels.

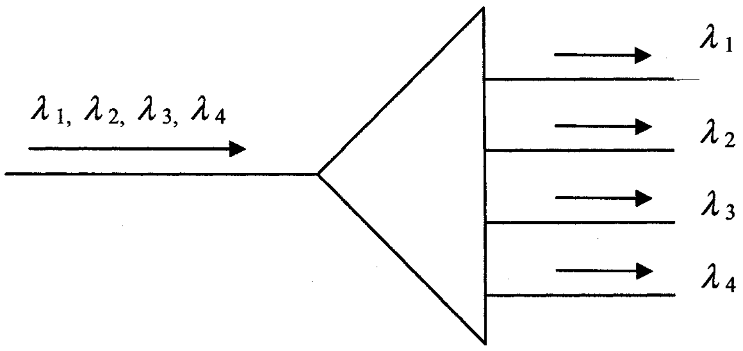
The use of multiple data streams on the same fiber is made possible by use of multiplexers and de-multiplexers. Different optical signals, each having different wavelengths are combined to be transmitted on a single fiber using Multiplexer.



Wavelength

Figure 2.2.2.2 Multiplexer

Figure 2.2.2.2 illustrates a typical wavelength multiplexer used in optical networks. The multiplexer combines four different carrier wavelengths λ_1, λ_2, λ_3 and λ_4, to be transmitted across the fiber.



DeMultiplexer

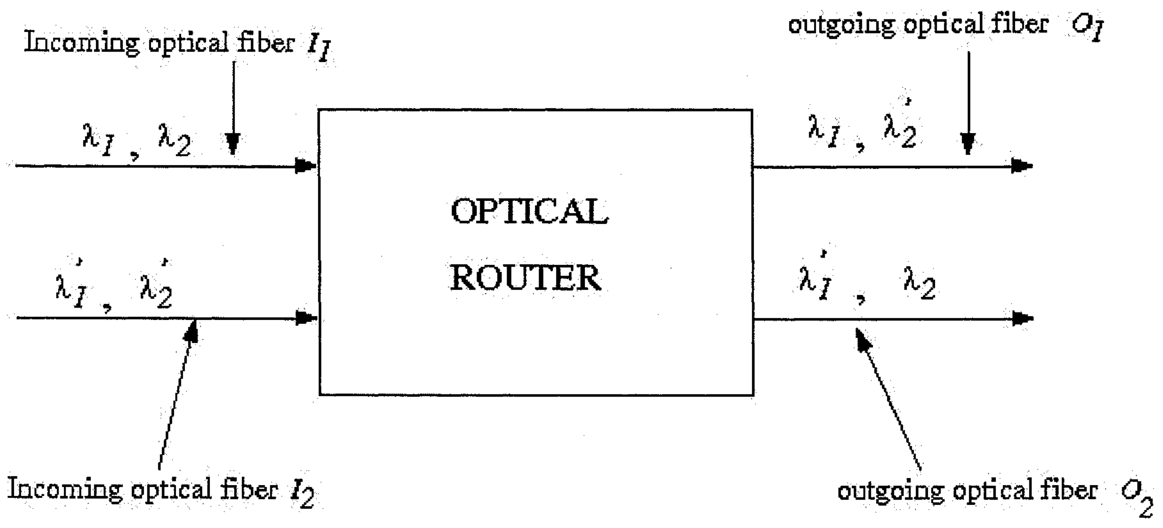
Figure 2.2.2.3 DeMultiplexer

Demultiplexers, on the other hand, splits the different optical signals carried by an incoming fiber into separate outputs, each carrying an optical signal at a distinct carrier wavelength. By using multiplexers (MUX) and demultiplexers (DeMUX), the system is able to achieve this simultaneous transmission without experiencing significant interference between the information channels [RS 02].

WDM Networks offer significant advantages viz.[Su 06]

- *Low signal attenuation* - As signal propagates through fibers, the signal strength goes down at a low rate (0.2db/km). This means that the number of optical amplifiers needed is relatively low.
- *Low signal distortion* - As signal is sent along a fiber optic network, the signal degrades, with respect to shape and phase. Signal regenerators are needed to restore the shape and timing. Low signal distortion means that signal regeneration is needed infrequently.
- *Low power requirement*
- *Low material usage*
- *Small space requirements*
- *Low cost*

Any possible source or destination of data is known as an end-node in optical networks [RS 02]. An optical router is a device used to route optical signals. It has a number of inputs, each carrying a number of optical signals and a number of outputs, again each carrying a number of optical signals. The optical router determines which incoming signal is routed to which outgoing fiber. The optical router routes optical signal coming in at an input port via the input optical fiber to appropriate output port onto the outgoing optical fiber, depending on the wavelength of the optical signal.



Incoming optical fiber I_1
λ_1, λ_2
λ_1', λ_2'
Incoming optical fiber I_2
OPTICAL ROUTER
outgoing optical fiber O_1
λ_1, λ_2'
λ_1', λ_2
outgoing optical fiber O_2

Figure 2.2.2.4 Optical Router

In the Figure 2.2.2.4 it can be seen that incoming optical fiber I_1 is carrying wavelengths λ_1 and λ_2 and the optical router routes the wavelength λ_1 to outgoing fiber O_1 but wavelength λ_2 is routed to outgoing optical fiber O_2.

2.2.3 Physical Topology

The interconnection of computers, optical routers and optical fibers define the physical network topology.

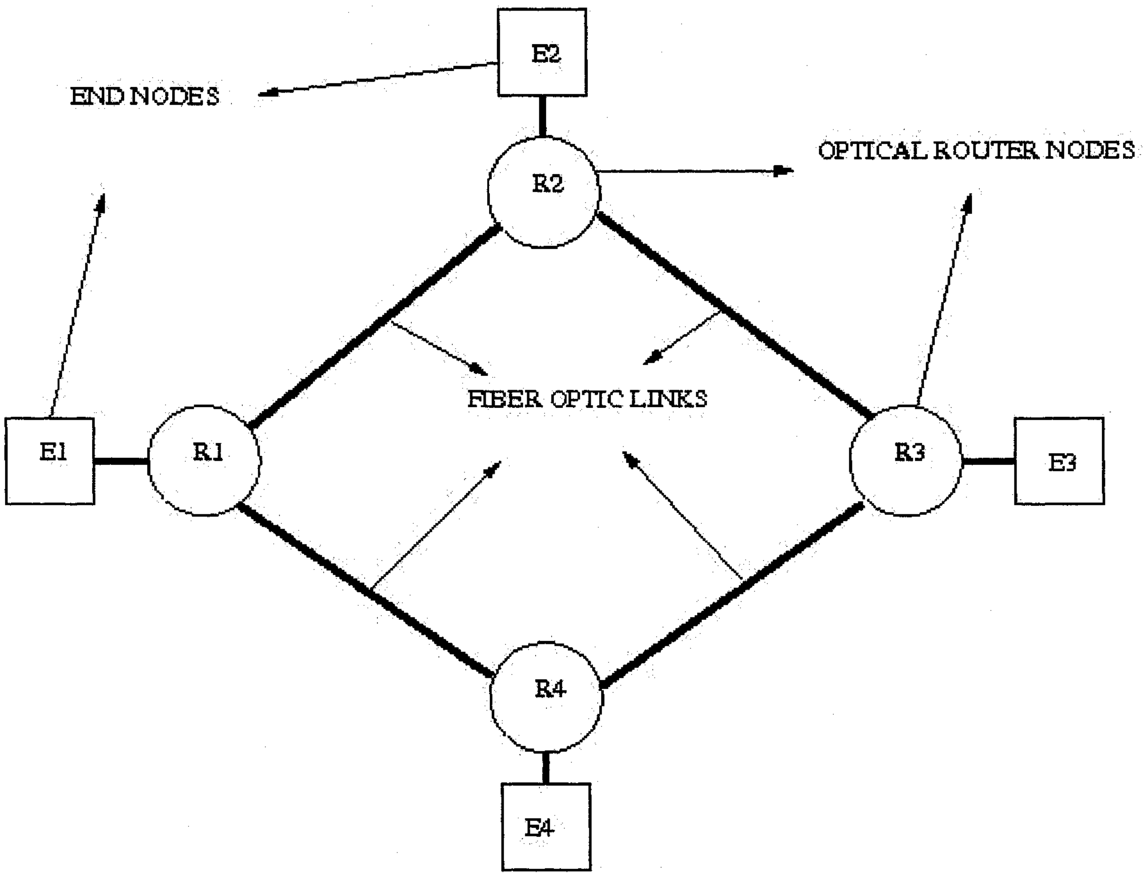


Figure 2.2.3.1 Physical Topology

Figure 2.2.3.1 shows one physical topology where the circles represent optical router nodes and the black solid lines connecting the circles represent optical fibers. The physical topology shown above is a highly simplified representation of how the optical routers and end-nodes are connected with each others using optical fibers.

2.2.4 Lightpath

A lightpath is an all-optical transmission path between two network nodes, implemented by the allocation of the same wavelength throughout the path [CFZ 96]. A lightpath may be viewed as an end-to-end, all optical communication channels from a source node *s* to a destination node *d*. If the wavelength continuity constraint is satisfied, the same channel must be used by a lightpath on every fiber in its path from the source to its destination.

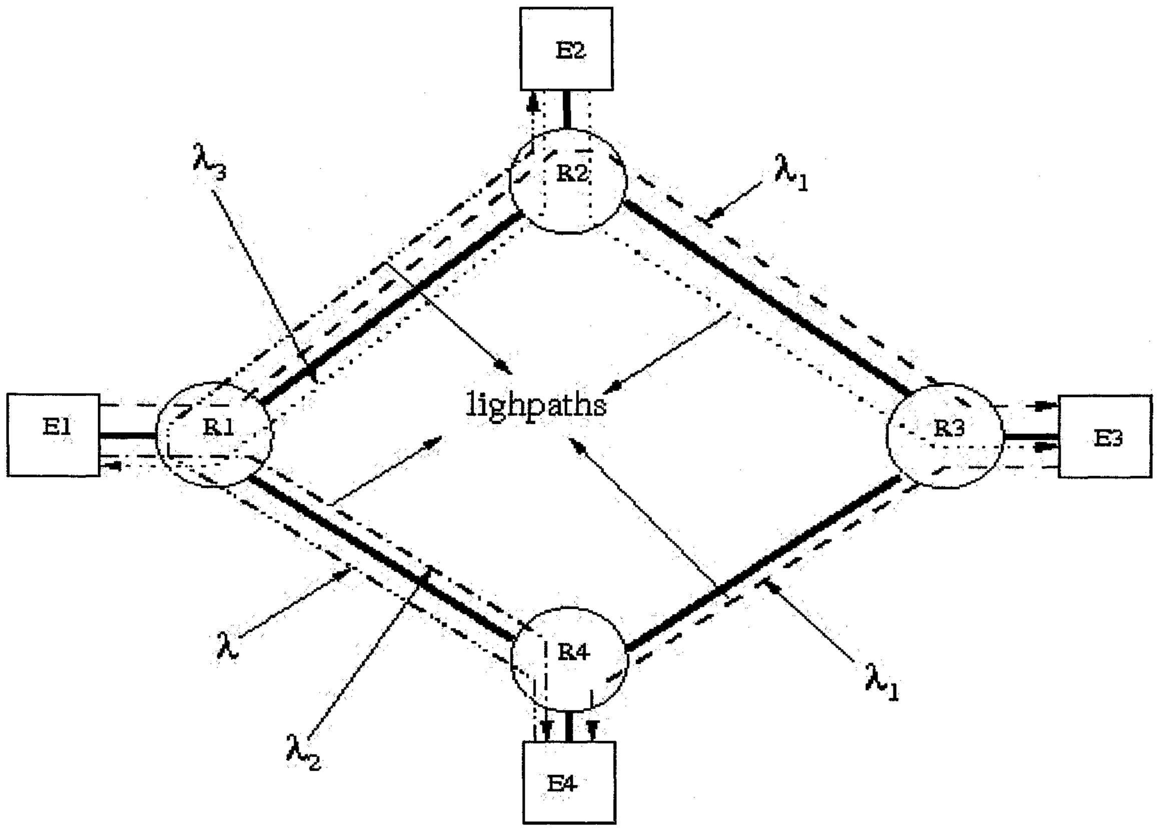


Figure 2.2.4.1 Lightpaths

As seen in the figure above the dashed or dotted connectors form the lightpaths using different available wavelengths. A lightpath from end-node *s* to end-node *d* using a carrier wavelength λ may be set up if there is a path *P* from end-node *s* to end-node *d* such that the wavelength λ is not used by any existing lightpath using any fiber in the path *P*. As seen in the figure 2.2.4.1 there is a lightpath with wavelength $\lambda 1$ from end node E1 to end node E3. Similarly there also exists a lightpath with wavelength λ from end node E4 to end node E2. The lightpaths in a WDM Network determine which end nodes communicate directly with other end nodes. Once the lightpaths are setup, it is no longer necessary to consider the physical topology to determine a communication strategy.

2.2.5 Logical Topology

The logical topology of a wavelength routed network is a directed graph where the computers are the nodes and the lightpaths are the directed edges. A logical topology represents communication channels between source and destination nodes. The logical topology defines an optical layer on which network can be built. Corresponding to figure 2.2.5.1 logical topology can be represented as shown below

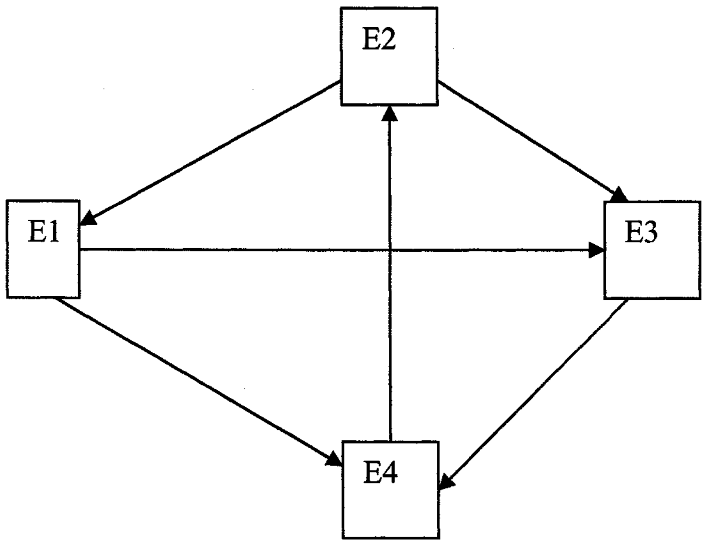


Figure 2.2.5.1 Logical Topology

The directed graph shown above in Figure 2.2.5.1 is called a logical topology. Fig 2.2.5.1 shows the various lightpaths available and the end-nodes that are connected using these lightpaths. Lightpath forms the basis of data communication for optical networks. As shown in Figure 2.2.5.1, end-node E2 can directly send data to node E3 using the lightpath E2 to E3. This path is not the only available path since communication from E2 to E3 can also take place via end-node E1 i.e. E2→E1→E3.

Generally there may be multiple paths available for communication between any two pairs of end-nodes. To communicate from a source end-node to a destination end-node, one or more appropriate paths need to be picked. When data is sent using a lightpath having one or more intermediate end-nodes; conversion of data from optical to electronic form takes place. For example if the path E2→E1→E3 is used, data is converted from electronic form to optical form at end-node E2 and transmitted to end-node E1. At end-node E1 the data is extracted and converted to electronic form, again the data is converted from electronic to optical form, possibly using a different channel, to be transmitted to end-node E3. At end-node E3, the data is again converted back to electronic form.

2.3 WDM Networks

A WDM network consists of routing nodes, interconnected by point-to-point fiber-optical links. [CGK 92] first proposed a light path based approach to design WDM networks to efficiently utilize the fiber bandwidth in wide area networks. WDM networks can be categorized as:

2.3.1 Single-Hop and Multi-Hop Networks

A network is classified as single-hop network when a data at source, once converted into optical form, remains in the optical form during transmission until it reaches the destination i.e. the transmitted signal remains in the optical realm along the path. Since the signal always remains in the optical domain a single-hop network is also called an all-optical network. In an ideal situation for an *n* node single-hop network it is needed to set up lightpaths between all *n(n-1)* nodes. This is not quite possible because available wavelengths are limited and also number of available transmitters and receivers is limited.

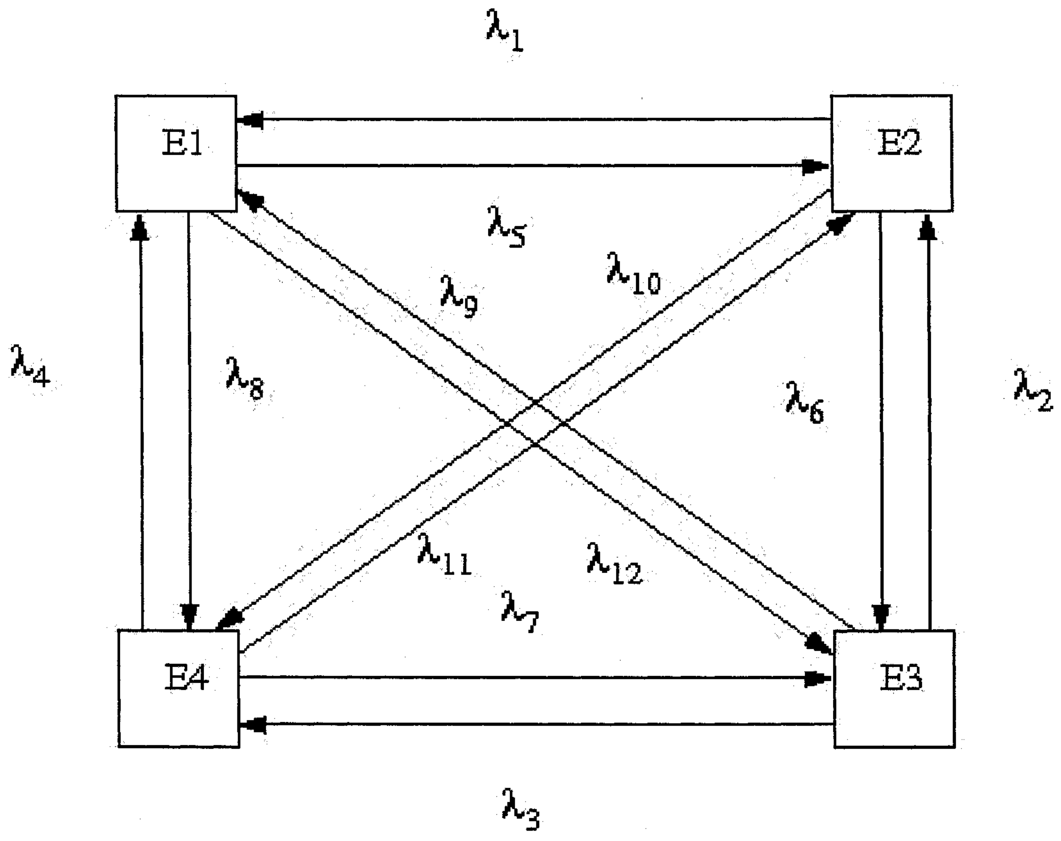


Figure 2.3.1.1 Single Hop Network

Multi-Hop networks were first introduced by Acampora in [Ac 87]. In these networks signal transmitted from source node reaches the destination node via one or more intermediate end-nodes. Signals are converted at each intermediate end-node from optical domain to electrical domain and back to optical domain to be transmitted along the route. In absence of a direct link between source-node and destination node Multi-Hop networks are used. Multi-Hop networks are not suitable for handling high throughput and delay-sensitive traffic [Mu 97]. Figure 2.2.5.1 shows a four node multi-hop network.

2.3.2 Broadcast and Wavelength Routed Networks

In broadcast network for unicast communication, the source end-node selects an appropriate wavelength λ_p and broadcasts the data to be transmitted to all end-nodes in the network using the wavelength λ_p. The receiver at destination end-node is tuned to wavelength λ_p and receiver at all other end-nodes are tuned to wavelengths different from λ_p.[Su 06]

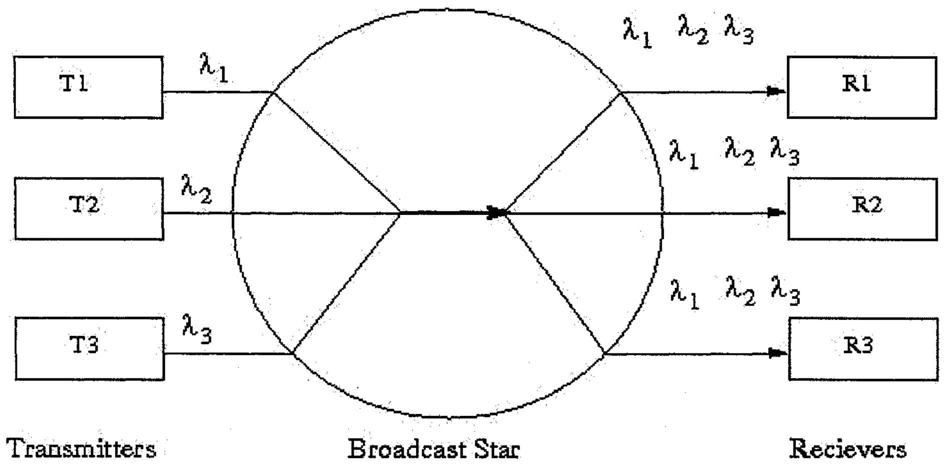


Figure 2.3.2.1 Broadcast network

In Figure 2.3.2.1 transmitters T1, T2 and T3 are transmitting at different frequencies all of them connected to passive star which broadcasts all three frequencies to each receiver and depending on which receiver is tuned to what frequency the appropriate signal is picked up by the receiver.

In wavelength routed networks, routers are connected through lightpaths that may extend over several physical links. At intermediate nodes, incoming channels belonging to in-transit lightpaths are transparently coupled to outgoing channels through an optical router [GLM 01]. For example the network shown in Figure 2.2.4.1 is a wavelength routed network since the end-nodes communicate using lightpaths which are routed from their sources to their respective destinations based on their wavelengths .If router node R1 receives a signal at wavelength λ it routes the signal to router node R2 and for router node R2 when it receives the signal for wavelength λ it routes the signal to end-node E2.

2.3.3 Static and Dynamic Lightpath Allocation

In static lightpath allocation the lightpaths are set up as per the expected traffic requirements, the lightpaths continue to be in existence for some weeks or months. Only when the traffic requirements change significantly, the existing lightpaths are taken down and new lightpaths established.

In dynamic lightpath allocation, the lightpaths are established on demand and are taken down once the communication is complete, In this arrangement a new demand requires setting up of new lightpaths.

2.4 Route and Wavelength Assignment (RWA)

Given the logical and physical topologies of the networks, one important aspect is to embed the logical topology onto the physical topology [MN 01].

The RWA problem is defined as follows:

Given a network topology and a set of end-to-end lightpath requests, determine a route and wavelength for requests, using the minimum possible number of wavelengths [RS 96].

Sometimes the routing of the lightpaths is already specified, so the only thing needed to determine is the wavelength assignment. For wavelength assignment two criteria must be satisfied:

1) No two lightpaths can have same wavelength on a given link
2) In the absence of wavelength converters, a light path must have same wavelength on all the links it traverses.

A lot of research has been done on RWA primarily using the following approaches:

1) RWA as graph Coloring Problem
2) Integer Linear Programming
3) RWA using Heuristics

Solving RWA using mathematical programming is computationally intractable even for moderate sized networks. Heuristics are of help in solving the RWA problem in reasonable time frame.

2.5 Logical Topology Design (LTD) Problem

In a WDM network, an optical fiber link has number of channels available for data communication. The problem of logical topology design is to find which pair of end-nodes are to be connected by a lightpath [Su 06].To setup a data communication between two optical nodes a transmitter (to send optical signal) and receiver (to receive optical signal) tuned to same wavelength is required. The logical topology design is limited by:

- Number of transmitters and receivers at each end-node
- Number of channels n_{ch} permitted on a fiber

The number of possible logical topologies is very large and the number of possible routing schemes for a given logical topology is also very large, hence the computation cost for optimum solution for such problems can be extremely high. Possible approaches to solve Logical Topology Design are:

- MILP (Mixed Integer Linear Program) based Logical Topology
- Heuristic for Logical Topology

2.5.1 MILP

Formulation of a problem that involves binary variables and some continuous variables is called mixed integer linear program (MILP). The problem of logical topology design is to determine which end-nodes are to be connected by lightpaths. Lightpaths may exist between any pair of end-nodes; this is represented using binary variables. Binary variables are needed for every pair of end-nodes. Let b_{ij} be a binary variable for each end-node pair (i,j) such that

For an N-node network

$$0 \leq i,j < N$$

$$bij = \begin{cases} 1 \text{ if a lightpath exists from end - node Ei to end - node Ej}, \\ 0 \text{ otherwise} \end{cases}$$

A common optimization objective is to minimize the maximum level of congestion [ZV 03]. Congestion is defined as traffic on the logical link which has maximum total traffic. Number of binary variables in MILP formulation directly depends on Number of nodes in network. As can be seen from the above equation for a *N*-node network number of binary variable *(bij)* is *N (N-1)*. The computational cost of this type of MILP is dominated by number of binary variables [Su 06]. The time taken to solve even a moderate size *N* node network is significant. It is a computationally intractable problem when the value of *N* is significant.

2.5.2 Heuristic

When size of the problem is sufficiently large sub-optimal techniques must be introduced to search for solutions by means of heuristic rules [CFG 98].Heuristic is defined as any technique used to obtain approximate optimal solution of a given problem. It ignores whether solution can be proven to be correct and it usually (but not surely) produces a good solution. Pseudo Code for Heuristic Logical Topology design Algorithm [RS 96] is given below

***Step** 1*: Given the traffic distribution matrix $P = (p_{ij})$,make a copy $Q = (q_{ij}) = P$.

***Step** 2*: Select the source destination pair (i_{max}, j_{max}) with largest traffic, i.e.,

$$q_{i_{\max}, j_{\max}} = max_{ij}\, q_{ij}$$

If all source-destination pairs with nonzero traffic have been tried already, then go to Step 4.

***Step** 3*: If node i_{max}, has fewer than degree Δ_l, outgoing edges, and node j_{max} fewer than Δ_l

```
        incoming edges, then find lowest available wavelength on the shortest
        propagation-delay path between $i_{max}$ and $j_{max}$ in physical topology $G_p$. (If there is
        more than one shortest path, scan them sequentially)
        If wavelength available then
            Create logical edge $i_{max} j_{max}$
            Find source destination pair $i^`, j^`$ with next highest traffic, i.e.,
            $q_{i^` j^`} = \max_{i \neq i_{\max}\ j \neq j_{\max}} q_{ij}$
            Set $q_{i_{\max}, j_{\max}} = q_{i_{\max}, j_{\max}} - q_{i^` j^`}$
            Go to Step 2.
        Else
            $q_{i_{\max}, j_{\max}} = 0$; go to Step 2.
Else
        $q_{i_{\max}, j_{\max}} = 0$; go to Step 2.
```

Step 4: If number of edges are less than $N\Delta_l$ edges, place as many remaining logical edges as possible at random so that degree constraints are not violated and a wavelength can be found on the shortest path for the logical edge.

2.6 Routing viewed as MCNF problem

Traffic requirement for a network of N end-nodes is represented in form of a matrix T where each individual entry $t_{i,j}$ ($0 \leq i,j < N$) gives us the amount of traffic to be routed from end-node E_i to E_j. For example if capacity of a logical link is 100 units, Traffic Matrix may be given as

$$T = \begin{bmatrix} 0 & 10 & 50 & 20 \\ 15 & 0 & 30 & 10 \\ 5 & 40 & 0 & 20 \\ 10 & 15 & 20 & 0 \end{bmatrix}$$

Figure 2.6.1 Traffic Matrix

where $t_{0,2}$=50 denotes that amount of traffic to be routed from end-node E_0 to E_2 is 50 units.

If the logical topology of a network is already known, then the other part of the problem is to route the traffic $t_{i,j}$ optimally i.e (corresponding routing needs to be determined). This type of problem is known as MCNF problem [AMO 93]. In a MCNF problem, a number of distinct commodities are required to be transported over a transportation network. A directed network graph $G=(V,E)$ may be viewed as the transportation network where V is a set of vertices for graph G where each vertex V_i $(0 \leq i < N)$ represents a possible source or possible destination. Each edge $i \rightarrow j$ represents a link on transportation network which has a specific capacity and cost per unit flow associated with it. Each commodity needs to be shipped from one or more sources to one or more destinations. A situation where there is only one commodity flowing is called as single commodity network flow problem and correspondingly if there are more than one commodities flowing it is known as multi-commodity network flow problem.

A small and medium sized network is identified as a network with less than 30 end-nodes. Since the logical topology is already fixed that implies the value of binary variables $b_{i,j}$ (refer Section 2.5.1) is already known, hence the problem boils down to a Linear Programming (LP) problem.

Let K^k be a distinct commodity associated with a source destination pair for example for Traffic Matrix shown in Figure 2.6.1 there are 12 non-zero entries i.e. there are 12 commodities.

Commodity K^1 is for pair (E_1,E_2), for row number 1 and column number 2 having a traffic of 10 units. Commodity K^2 is for pair (E_1, E_3), for row number 1 and column number 3 having a traffic of 50 units. Commodity K^{12} is for pair (E_4,E_3), for row Number 4 and column number 3 having a traffic of 20 units. The network flow problem as given in [Su 06] can be formulated as:

Let

N – Number of end-nodes in the network.

q – number of commodities

$$\left.\begin{array}{l} src_k \text{ – source End-node } E\,s_k \\ dest_k \text{ – destination End-node } E\,d_k \end{array}\right| \text{ of commodity } k\ 1 \leq k < q$$

T^k- traffic $t(s_k, d_k)$ for commodity k, $1 \leq k < q$

x_{ij}^{k} - is amount of traffic on commodity k flowing on logical edge $E_i \rightarrow E_j$

Λ_{max} - congestion in the network

L – set of all pairs of end-nodes (E_i, E_j) *s.t.* a lightpath exists from end-node E_i to end-node E_j

m - number of logical edges in the network

Objective Function

Minimize Λ_{max}

Subject to

1) Ensure that Λ_{max} is the congestion

$$\sum_{k=1}^{q} x_{ij}^{k} \leq \Lambda_{max}, \ \forall (i,j) \in L$$

2) Apply the conservation rules

$$\sum_{j:(i,j)\in L} x_{i,j}^{k} - \sum_{j:(j,i)\in L} x_{j,i}^{k} = \begin{cases} T^{k} \, if \, src_{k} = i \\ -T^{k} \, dest_{k} = i, \forall i, 0 \leq i \leq N \\ 0, otherwise \forall k, 0 \leq k \leq N \end{cases}$$

It is possible to solve the LP for small and medium sized using a commercial package (e.g CPLEX).

2.6.1 Routing over Logical Topology

Once the logical topology is in place, it is important to determine the routing of traffic over the logical topology to ensure the desired data communication. Yumei's algorithm [Lu 04] makes use of the approximation algorithm by Lisa Fleischer for multi-commodity network flow problems to solve the routing problem. In this approach, routing over a logical topology is viewed as a multi-commodity flow problem. The multi-commodity network flow problem is defined over a network where more than one commodity needs to be shipped from their respective sources to their destinations without violating the capacity constraints associated with each arcs [AL 94].

In this approach for a given graph the nodes represent sources and sinks of the commodities and edges represent a way to transport a commodity from one node to another [Lu 04]. The main contribution of Yumei's work [Lu 04] is to formulate the congestion minimization problem for WDM networks as a MCNF (Multi Commodity Network Flow) problem and then use an approximation algorithm to solve this problem efficiently [Lu 04]. Each source destination pair (*s, d*) in a network having non-zero traffic from the source to the destination corresponds to a commodity from *s* to *d*. The problem is to determine the flows of all the commodities in the network to satisfy the traffic requirement for each commodity. In this approach the result is guaranteed to be within pre-determined bounds of optimal solution.

2.6.2 The Approximation Algorithm

For any linear programming formulation i.e. primal, there exists a corresponding dual formulation [Ta 89]. In case the primal is a minimization problem then the corresponding dual formulation is a maximization problem and vice versa. If both the primal and the dual formulation of a problem are the same, then the optimal value is the same as the objective value of primal and dual solution. If P_i is a primal formulation, D_i is the corresponding dual formulation; and obj_p and obj_d be their objective function values respectively then

1) if primal is maximization and dual minimization problem

$$obj_p \le optimal\_value \le obj_d$$

2) if primal is minimization and dual maximization problem

$$obj_d \le optimal\_value \le obj_p$$

Keeping the above idea in mind it can be said that if the solution is an optimal solution then $obj_d = obj_p = optimal\_value$. In situations where it is computationally expensive to find the optimal solution, it may be a wise idea to find an approximate solution. A way to do that is by using the relationship between the optimal solution, the primal solution and the dual solution, so that the optimal solution always lies between the primal solution and the dual solution. Yumei's algorithm also exploits the above said relationship. The algorithm for Yumei's approach is illustrated below:

Let

d_j units of traffic sent over the network

(s_j, t_j) source destination pair

P_j denotes the set of paths (from s_j to t_j)

$d_{max} = max \{d_j : j = 1, 2, 3, K\}$

$x\ (p)$ to denote the traffic flow on path p

$l(p_j^*)$ The length of an edge represents the marginal cost of using an additional unit of capacity of the edge.

ε accuracy desired for example ε value is 0.01 for accuracy of 1%.

δ a very small value chosen as 0.001 in experiments carried out

$l\ (e)$ dual value associated with each edge

u network congestion

E set of edges

r is the iteration number

```
1. Choose values for $\delta$ and $\varepsilon$ ($\delta>0$ and $\varepsilon>0$)
2. $l(e) := \delta / d_{\max}, x(e) = 0, \forall e \in E$
3. r = 0
4. do
        a. $r = r+1$
        b. for $j=1$ to $K$
                i) $l(p_j) :=$ shortest path distance for commodity $j$, j=$1, 2, 3\ldots\ldots K$
                ii) $x(p_j) := x(p_j) + d_j$
                iii) $l(e) := l(e)[1 + \varepsilon d_j / d_{\max}]$  $e \in p_j$
          end for
        c. primalSolution $= r d_{\max} / \max\{x(e) : e \in E\}$
        d. dualSolution = $\sum_{e \in E} l(e) \; / \; \sum_{j \in k} d_j l(p_j)$
        e. u= $\max\{x(e) : e \in E\}$ / r
while ((dualSolution / primalSolution) < $(1+\varepsilon)$) [Yu 06]
```

STEP 1 set appropriate values for δ and ε.

STEP 2 set the initial values for the dual variables *l(e)*, a very small non-zero value

(δ/d_{max}).

STEP 3 Initialize the iteration counter *r*.

STEP 4

a) Repeated until the final solution is found.

b) Update the iteration counter r.

c) For each commodity j, the following actions are taken:

i) Calculate the shortest path for each commodity,

ii) Send dj units of flow along the shortest path p for commodity j, and

Update the flow on each edge $e \in p$, by dj.

iii) Update the length of each edge $e \in p$ as follows: $l(e) := l(e)[1 + \varepsilon d_j / d_{\max}]$

d) Calculate the primal objective value (L)

e) Calculate the dual objective value (U)

f) If (U/L) < (1+ε) STOP

Otherwise go back to STEP 4(a).

2.7 Tabu Search

A multitude of difficult optimization problems are encountered in practical settings like telecommunication, logistics, financial planning etc. This has spurred new methodologies and techniques to solve these problems, a lot of these techniques derive ideas or approach from already existing research areas. For example, Genetic Algorithms (GAs) approach borrows from biological phenomenon of evolutionary reproduction while Simulated Annealing adapts from physical process in metallurgy. Tabu search selectively borrows from the fields of artificial intelligence and optimization. Tabu search is based on ideas proposed by Fred Glover. "Tabu search is a general heuristic procedure for guiding search to obtain good solution in complex spaces [Gl 90]".

Heuristics have long been used to tackle combinatorial problems. Heuristic is any technique used to obtain an approximate optimal solution of a given problem. Heuristics generally ignore whether a solution produced by the heuristic can be proven to be correct. Meta-heuristic is a higher level heuristic procedure, it is a strategy to guide other heuristics.

Tabu Search is a meta-heuristic that guides a heuristic search procedure to explore the solution space beyond the realm of local optimality. Coarsely speaking tabu search can be interpreted as an extension of hill climbing algorithm with respect to neighborhood structure and move evaluation function. Tabu search is based on procedures geared towards stepping out of the boundaries of local optimality to explore new regions, in search of a globally optimal solution.. Tabu search uses flexible memory structures to intelligently guide the search method. Tabu search does not guarantee an exact solution but opens up the possibility of achieving an approximately optimal solution in an efficient way. Consequently lot of research has been focused in Tabu Search for combinatorial problems where it has shown promising results in various problems.

2.8 Memory in Tabu Search

The search space or the solution space is identified as the set of all possible solutions. A ***move*** in tabu search is defined as any transformation which when applied to a given solution gives a different solution. ***Neighborhood*** in tabu search is identified as a set of solutions which can be achieved by applying a single move.

Tabu search is a meta-heuristic which uses local or neighborhood search procedure. It iteratively moves from a solution x to a solution x' in the neighborhood of x until some stopping criterion has been satisfied. The tabu search does not allow moves which revisit the solution recently visited, ensuring that new regions of solution space will be investigated. To achieve the said functionality tabu search employs memory based strategies to open up regions of solution space. Adaptive Memory and Responsive exploration are the key features that make tabu search intelligent [LG 98]. *Adaptive Memory* refers to selectively remembering key elements of the solutions which have been visited during the search procedure. *Responsive exploration* is to make strategic choices which would allow exploration of unvisited regions in the solution space, while exploiting good solution features.

Memory in tabu search can be both explicit and attributive. In explicit memory complete solutions are stored. Generally these are the *elite solutions* evaluated during the search. It is possible to come back to these solutions to explore them or their attractive neighbors. Attributive memory, on the other hand, is used to store information about attributes that change in moving from one solution to other. This information may be used to make a strategic move for visiting unexplored solution space.

2.9 Short Term Memory

Short term memory is a key aspect of tabu search. "The core of tabu search is embedded in its short-term memory process" [Gl 90]. Short-term memory helps to guide the exploration to find the best possible move within the neighborhood, subject to pre-determined tabu restrictions. The tabu restrictions aid to prevent repetition or reversal of the moves.

A hill climbing algorithm [Gl 96] also makes use of short-term memory, but no restrictions are applied. A simple illustration of a maximization hill climbing algorithm is given below:

Let Neighborhood $N(i)$ be defined for a current solution i, and the next solution j be searched among the solutions in $N(i)$. Consider the following algorithm:

```
Step 1)  Start with an initial solution $i$.
Step 2)  Find a best $j$ in $N(i)$ (i.e. such that $f(j) \geq f(k)$ for any $k$ in $N(i)$).
Step 3)  If $f(j) \leq f(i)$ then stop.
         Otherwise set $i=j$ and go to Step 2.
```

The above algorithm is quite likely to get stuck at a local optimum. The objective is to somehow get out of this local optimum. To achieve this some guiding strategy is required so that it is possible to accept unfavorable solutions, allowing exploration of more regions in the search space. However, the guiding strategy should be such that cycling (i.e. coming back to same set of solutions) can be avoided. The tabu search does restrict the recent moves from neighborhoods by a dynamic list of moves called the "Tabu List".

2.9.1 Tabu List and Tabu Tenure

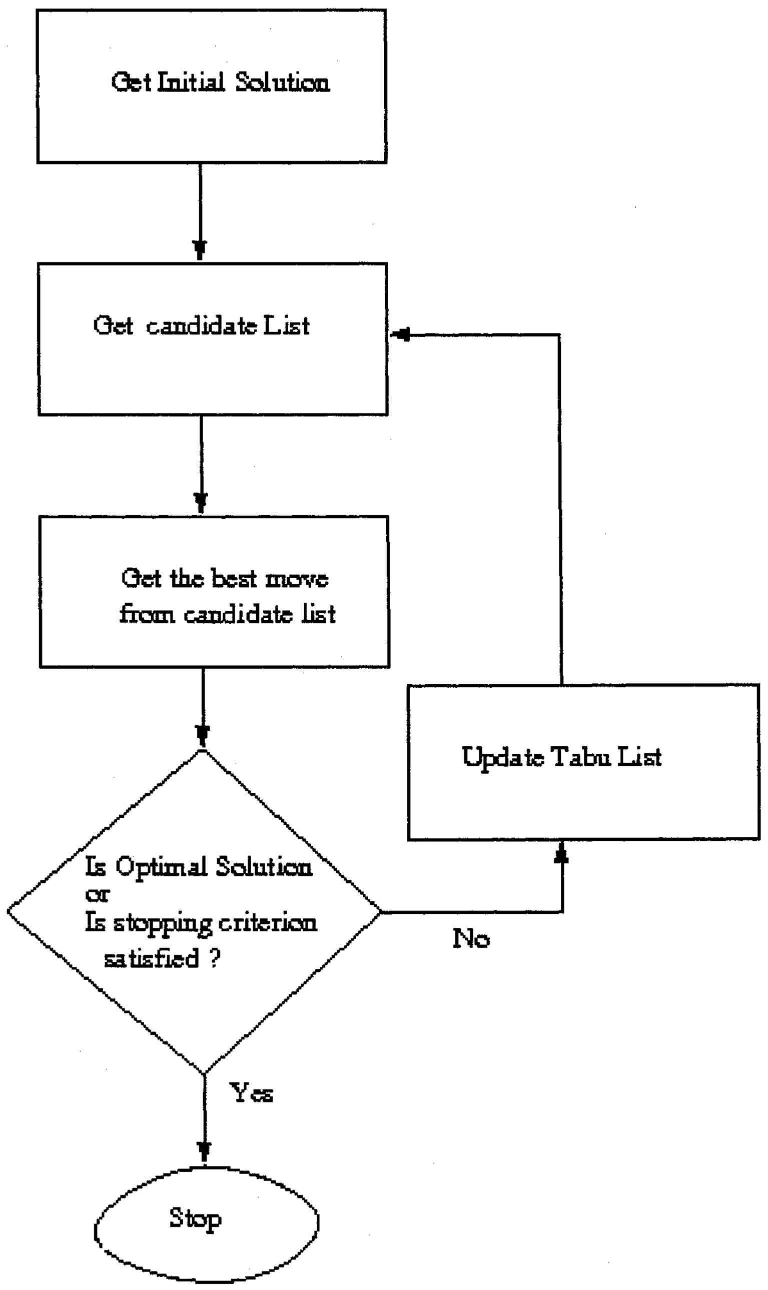
Tabu List plays a significant role in guiding the search procedure. The tabu list is key component of tabu search, giving direction to tabu search. Tabu Search is essentially an iterative procedure, where iteration is said to be complete when a move has been made successfully. The tabu list can be viewed as a FIFO (First In First Out data structure) storing recent moves. The tabu list stores a set of most recent moves which are forbidden.

Any move which is successfully applied is added to the tabu list, replacing the oldest move in the list. How long a move is to remain in tabu list, is decided by experimentation. The size of the tabu list, i.e. the number of iterations for which a move is tabu, is known as *tabu tenure*. The tabu tenure varies according to the current problem; it may remain constant throughout the problem solving or may vary at different stages in the problem[Gl 96]. There is no general rule which specifies the size of a tabu list. A small sized tabu list can result in occurrence of cycles; on the other hand a large tabu list may affect the quality of the solution obtained. Size of the tabu list is a trade off between the two factors, generally decided on basis of empirical observations.

2.9.2 Candidate List

To get the best quality solution from the neighborhood it does seem attractive to define a large neighborhood, but it is not always practical to do so. It may be computationally very expensive to retrieve and evaluate all the moves of the neighborhood [Gl 96]. An obvious way out is not to examine the whole set of moves, but to work with a subset of moves, this subset is referred to as candidate list. The candidate list narrows down the neighborhood to be examined. It may be viewed as a trade off between quality of the solution and effort required to find it.

A simple representation of tabu search using short term memory is illustrated in the flowchart below:



Get Initial Solution
Get candidate List
Get the best move from candidate list
Is Optimal Solution or Is stopping criterion satisfied ?
No
Update Tabu List
Yes
Stop

Figure 2.9.2.1 Short term memory

2.9.3 Aspiration Criteria

Aspiration Criteria determines when the tabu rules can be overridden. During the course of tabu search it may seem attractive to make a move even if it is a tabu move. A simple situation would be when that making a move, the move gives an improvement over previously recorded best solution but the move is a tabu move. The aspiration criterion prevents tabu search from being too restrictive and enables a tabu search method to achieve its best performance levels [Gl 96]

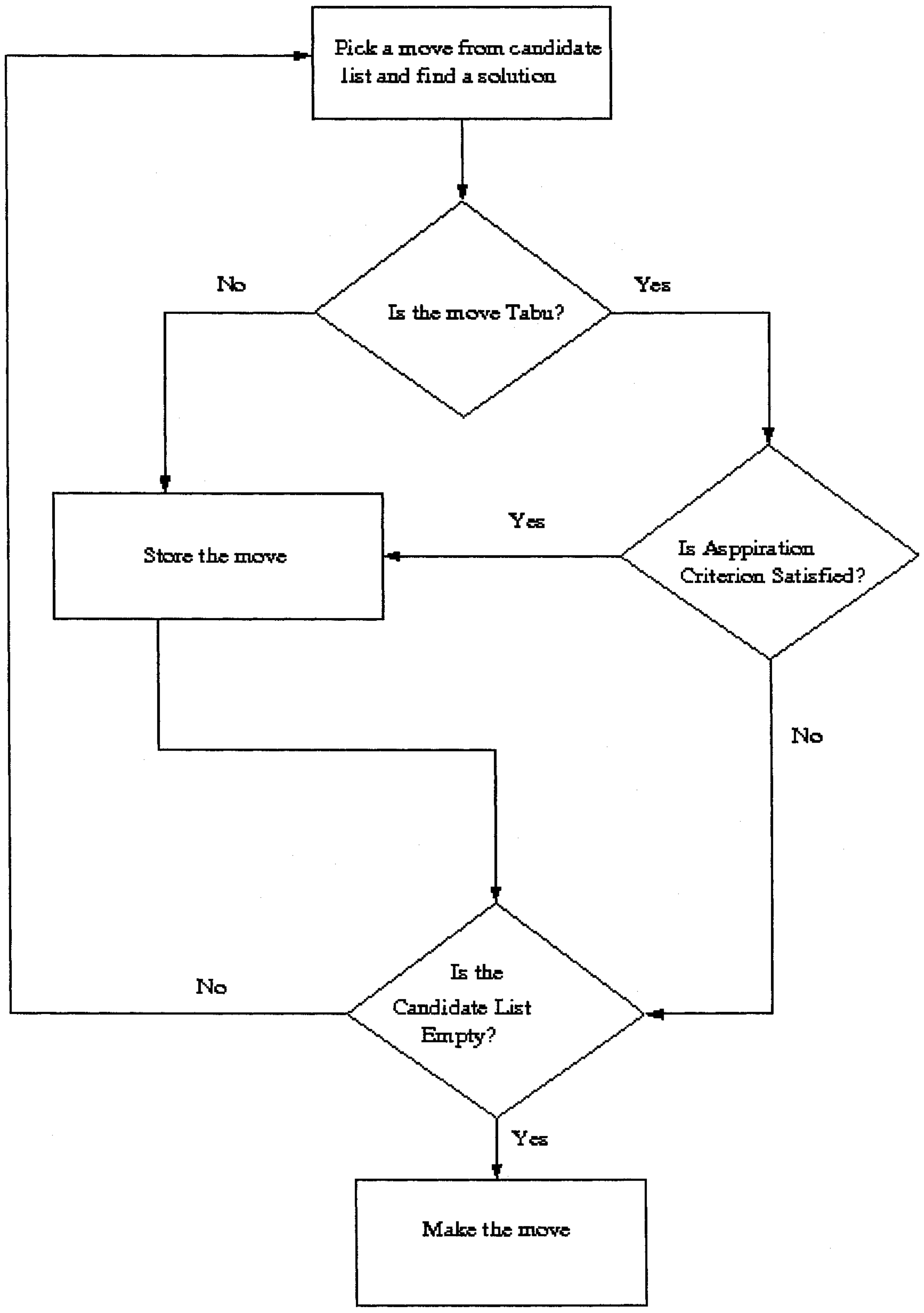

Pick a move from candidate list and find a solution
Is the move Tabu?
No
Yes
Store the move
Yes
Is Asppiration Criterion Satisfied?
No
No
Is the Candidate List Empty?
Yes
Make the move

Figure 2.9.3.1 Selecting the best candidate

2.10 Long Term Memory

In may not always be possible to obtain very high quality solutions using short term memory approach. Tabu search is significantly enhanced by using the long term memory approach [Gl 90]. The short term and the long term tabu search approaches both strive to modify the neighborhood of current solution. In short term approach the neighborhood is always the subset of current solution. Long term approach helps guide the search process to solutions which are not found in immediate neighborhood of the current solution. The idea is to make a move even though unattractive, but which would take us to a different region of solution space. The new region can now be thoroughly searched for attractive solutions. This allows the search process to escape the trap of local optimality. For a search process which is stuck at local optima, as an example it is possible to make a random move hoping that it takes the search process to a different region. The use of long term memory cuts down on search time and helps in achieving the desired results faster than otherwise possible. The idea is to make a move that is not a part of the defined neighborhood. Long term memory approach makes use of the concept of *Intensification* and *Diversification*.

2.10.1 Intensification

Intensification strategies are employed to encourage various move combinations and solution features found historically good [Gl 90]. This is achieved by modifying choice rules. The intensification strategies allow the search process to return to attractive regions to search them more thoroughly. Intensification by itself is not enough to obtain best solutions. Intensification tries to search a solution region more thoroughly..

A simple representation of tabu search intensification approach is given below:

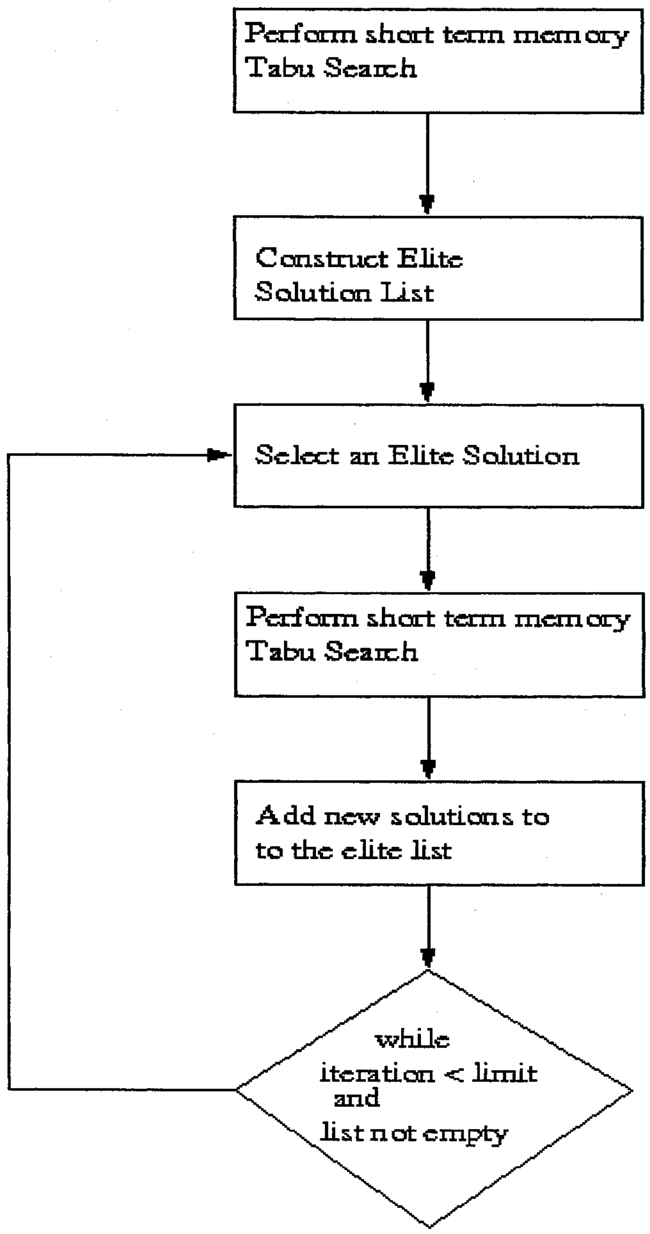

Perform short term memory Tabu Search
Construct Elite Solution List
Select an Elite Solution
Perform short term memory Tabu Search
Add new solutions to to the elite list
while iteration < limit and list not empty

Figure 2.9.1 Simple Intensification Approach

2.10.2 Diversification

The solution space of a given optimization problem has to be searched effectively to achieve the objective of attaining a global optimum[Gl 90]. The diversification strategies provide a mechanism using which the search process can be guided to the unvisited regions of the solution space. The diversification strategies help to generate solutions which may significantly differ from the previous solution.

CHAPTER 3

Problem Specification and Approach

In chapter 2, the problem of optimal logical topology design in WDM networks and the idea of Tabu search were reviewed briefly. It was pointed out, in section 2.5.1, that the standard MILP formulation of the optimal logical topology design problem is expensive to solve for non-trivial networks. In the approach using heuristics, the quality of the solution is unknown. It was also stated, in section 2.6 that, if the logical topology is known, then the problem of routing over the logical topology is a LP problem which can be solved using a LP tool, such as CPLEX, for small and medium sized networks. Even so, the LP is time consuming, since the basis size is O(N^3) where N is the number of end-nodes in the network. One useful way to speed up the process of finding the routing is to allow a sub-optimal routing. This could be done using a heuristic but the quality of such a heuristic is unknown. It was discussed in chapter 2 that it is possible to use approximation algorithms to find a near optimal routing [Lu 04] with a relatively modest computational effort. The advantage of using this approximation algorithm is that the approximation bound is defined in advance, so that the quality of the routing is known. As mentioned, in section 2.6.1, the problem of optimal logical topology design may be simplified by splitting the problem into two sub-problems as follows:

- logical topology design and
- routing over the logical topology.

These two problems are interrelated since the optimal routing cannot be determined until the logical topology is fixed, so that, ideally, the two problems should be solved simultaneously.

The research reported in this chapter views the optimal logical topology design problem as a search problem. Since the search space, consisting of all possible logical topologies of a given WDM network is vast, a Tabu Search technique has been used to limit the time needed to find candidate logical topologies. Once a logical topology is determined, the routing on this logical topology is carried out efficiently using Yumei's approximation algorithm.

3.1 Viewing the problem as a state-space search

To view this problem as a state-space search problem, it is convenient to define the current logical topology as the current state. A move in the state-space means, that the current logical topology is changed to some new logical topology. A move, therefore, is the application of a perturbation to the current logical topology to give a new logical topology. Since the specification of the problem of optimal logical topology design includes the total number of transmitters and receivers in the network, and it is reasonable to use all the resources of the network to reduce the congestion, all transmitters and receivers should be in use in the current topology. This means that the number of logical edges in the network cannot be increased in the new logical topology. In order to get a new logical topology, without using additional transmitters and receivers, n logical edges have to be deleted, for some predetermined number n. The deletion of a logical edge $x \rightarrow y$ frees up a transmitter at end-node x and a receiver at end-node y so that, if n logical edges are deleted, n transmitters and n receivers become available. These newly available transmitters and receivers may be allotted to end-nodes, as needed, and used to set up new lightpaths resulting in a new logical topology and hence a new state. Since n transmitters and n receivers are available, after the deletion of n logical edges and the objective is to reduce the congestion, it is reasonable to use all n available transmitters and receivers to define n new logical edges. In summary, a move is the deletion of n logical edges, followed by the addition of n logical edges. The value of n has to be fixed in advance.

The following greedy hill-climbing algorithm is a straightforward approach to the problem.

Step 1) Using any existing heuristic, define an initial logical topology L for the specified WDM network and the traffic matrix.

Step 2) Find the congestion Λ after determining a routing over the current logical topology L.

Step 3) Find the best move from the current state. The best move is the move so that a routing over the new logical topology gives the smallest congestion. Let the congestion for the best move be Λ_{new} and let the corresponding logical topology be L_{new}.

Step 3) If $\Lambda_{new} \geq \Lambda$, stop. The logical topology L is the optimal topology.

Step 4) Otherwise, the next move is the state representing the new logical topology L_{new}. Set $\Lambda = \Lambda_{new}$ and $L = L_{new}$. Go to step 3.

3.2 An example using 4-node network

A traffic matrix *T* and a logical topology are shown in Figure 3.2.1. A routing to handle the traffic requirements as specified by the traffic matrix *T* is as follows:

(Source, Destination) pair	Amount of traffic	Logical paths	Traffic handled by the logical path
(E1,E2)	60 units	*E1→E3→E2* *E1→E4→E3→E2*	30 units 30 units
(E2,E1)	20 units	*E2→E1*	20 units
(E2,E3)	35 units	*E2→E3*	35 units
(E3,E4)	30 units	*E3→E4*	30 units
(E4,E1)	10 units	*E4→E1*	10 units
(E4,E3)	20 units	*E4→E3*	20 units

Table 3.2.1 Table showing the logical paths

Traffic Matrix

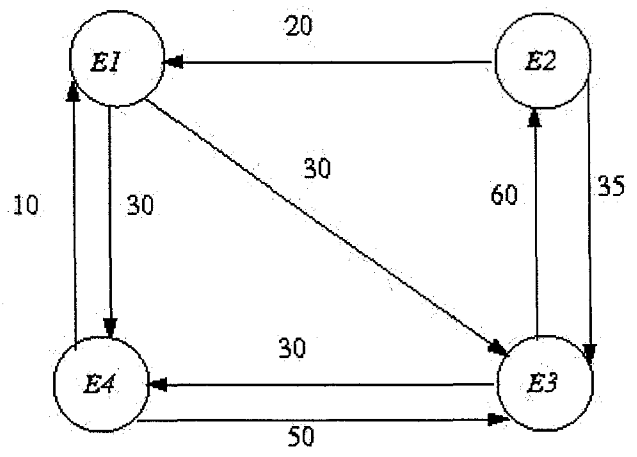
$$T = \begin{bmatrix} 0 & 60 & 0 & 0 \\ 20 & 0 & 35 & 0 \\ 0 & 0 & 0 & 30 \\ 10 & 0 & 20 & 0 \end{bmatrix}$$

Logical Topology

$$L = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

$$F = \begin{bmatrix} 0 & 0 & 30 & 30 \\ 20 & 0 & 40 & 0 \\ 0 & 60 & 0 & 30 \\ 10 & 0 & 50 & 0 \end{bmatrix}$$

Flow Matrix



Logical topology with flows on each edge

Figure 3.2.1 a 4-node network example

From the above routing of flows, as shown in Figure 3.2.1, the congestion for the network is 60 units on the edge *E3→E2.*

This logical topology is not necessarily the optimum topology. To apply the hill climbing heuristic described above, it is necessary to fist define a move. As explained above, a move is to delete *n* logical edges and then add *n* logical edges, for some *n*. The simplest case is to set $n = 1$. In other words, the logical topology *L* may be modified to get a new logical topology *L1* by deleting a logical edge and then adding a logical edge.

In the case of the above topology *L*, any of the existing logical edges of the topology shown in Figure 3.2.1 may be deleted. Considering the case when the logical edge *E4→E1* is deleted, a logical edge that does not exist in *L* needs to be added. The various possible edges that could be added are as follows:

i. Logical edge *E1→E2*

ii. Logical edge *E2→E4*

iii. Logical edge *E3→E1*

iv. Logical edge *E4→E2*

In the above list, logical edges, such as *E1→E3,* were not included since such edges already exist in the logical topology. Since one move for the search is the deletion of one logical edge and the addition of one logical edge, four moves are possible for the case where the logical edge *E4 → E1* is deleted.

To enumerate all the moves, each existing logical edge in the topology has to be considered as a possible candidate for deletion, and the corresponding edges that may be added can be determined.

Some routing algorithm, such as the approximation heuristic described in chapter 2, may be used to find the routing after each move is carried out. The best move is the one that gives the least congestion. For instance, after deleting the logical edge *E4→E1* and adding the logical edge *E4→E2,* gives the logical topology *L1.*The routing algorithm carried out on the new logical topology *L1* and the traffic matrix *T* gives a new value (40 units) of congestion. This is a better value of congestion as compared to the previous logical topology; as such the previous logical topology may be replaced by the new logical topology for next iteration.

In the next iteration, all possible moves on the logical topology *L1* has to be considered. For instance, one move is as follows:

a. delete the logical edge *E4→E3* and

b. add the logical edge *E1→E2*

This modification gives us the new logical topology *L2* (Figure 3.2.3). The congestion for this logical topology is 55 units for the logical topology *L2*. It may be verified that all possible moves on the logical topology *L1* give a congestion value worse than that for *L1*.

Traffic Matrix

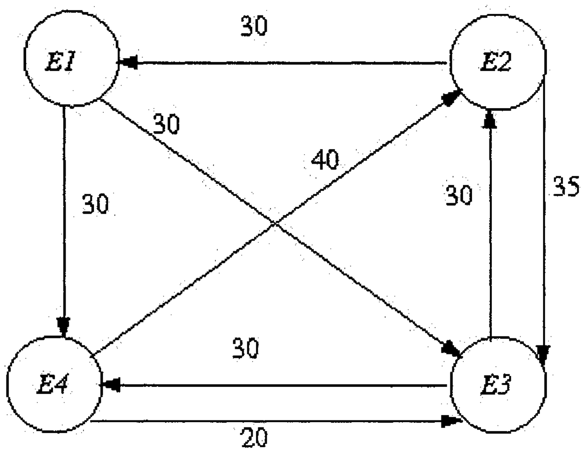
$$T = \begin{bmatrix} 0 & 60 & 0 & 0 \\ 20 & 0 & 35 & 0 \\ 0 & 0 & 0 & 30 \\ 10 & 0 & 20 & 0 \end{bmatrix}$$

Logical Topology

$$L1 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$F1 = \begin{bmatrix} 0 & 0 & 30 & 30 \\ 30 & 0 & 35 & 0 \\ 0 & 30 & 0 & 30 \\ 0 & 40 & 20 & 0 \end{bmatrix}$$

Flow Matrix



Modified logical topology with flows on each edge

Figure 3.2.2 Logical Topology after one move

Traffic Matrix

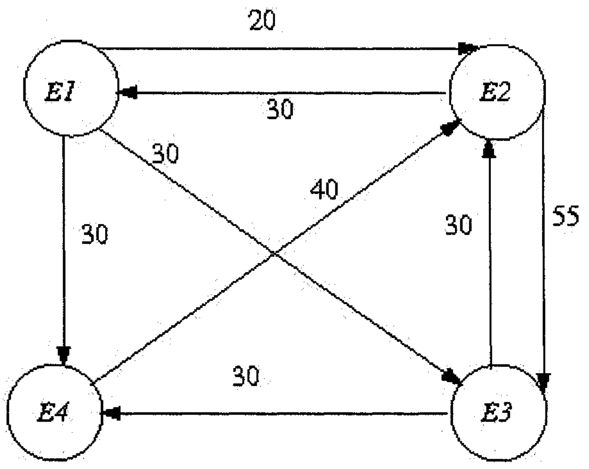
$$T = \begin{bmatrix} 0 & 60 & 0 & 0 \\ 20 & 0 & 35 & 0 \\ 0 & 0 & 0 & 30 \\ 10 & 0 & 20 & 0 \end{bmatrix}$$

Logical Topology

$$L2 = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$F2 = \begin{bmatrix} 0 & 20 & 30 & 30 \\ 30 & 0 & 55 & 0 \\ 0 & 30 & 0 & 30 \\ 0 & 40 & 0 & 0 \end{bmatrix}$$

Flow Matrix



Modified logical topology with flows on each edge

Figure 3.2.3 Logical Topology after another move

There are two following major problems with the greedy hill-climbing heuristic given above:

- In any non-trivial network, the number of possible moves from a given state is very large. In other words, the branching factor in the search tree is very large. It is computationally intractable to search such a large tree and it is necessary to limit the search to a manageable size.

- Hill climbing strategies get stuck in a local optimum.

These issues are discussed below.

3.3 Limiting the number of moves

As explained above, the simplest case is when $n = 1$ and a move is to delete a logical edge and add a logical edge. Therefore, for each move when $n = 1$, there are two sub-problems to be solved:

- identifying the edge to be deleted and
- identifying the edge to be added.

3.3.1 Identifying the edge to be deleted

For a give state, the approach followed here for identifying the edge to be deleted, is to delete the edge that carries the minimum traffic. The rationale is that deleting the edge carrying the minimum traffic would have the least impact on the overall flow of the network traffic.

3.3.2 Identifying the edge to be added

Since the congestion is the traffic on the edge carrying the maximum traffic, and the objective is to reduce the value of the congestion, the search space investigated in this research is the neighborhood of the logical edge carrying the maximum traffic as follows.

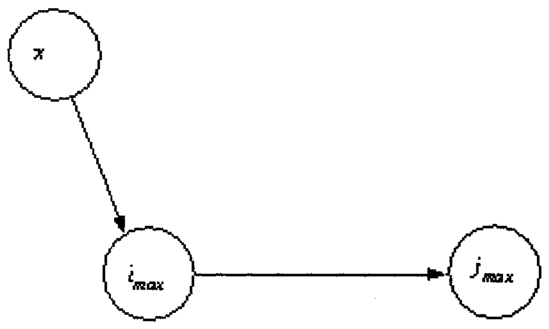


Figure 3.3.2.1a

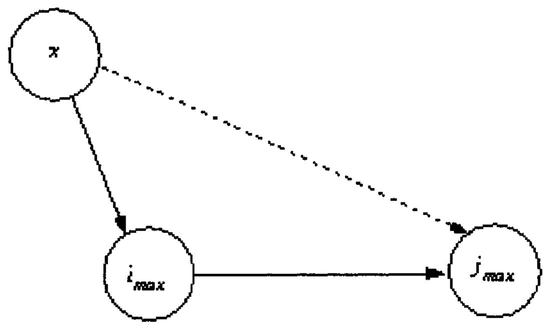


Figure 3.3.2.1b

Figure 3.3.2.1 Adding of Logical Edge Strategy 1

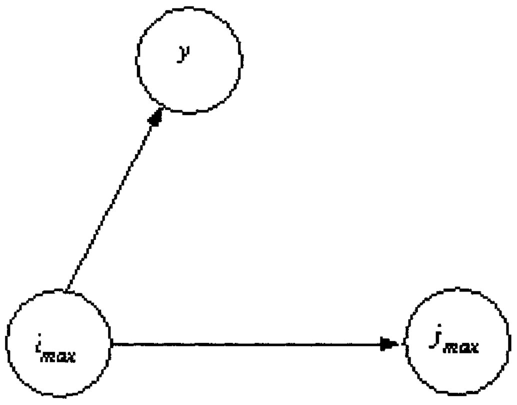
Let the logical edge carrying the maximum traffic be the edge from end-node $i_{\max}$ to end-node $j_{\max}$(Figure 3.3.2.1a). The objective, when adding a logical edge, is to reduce the total traffic on the edge $i_{\max} \rightarrow j_{\max}$.

The traffic on the edge $i_{\max} \rightarrow j_{\max}$. has the following sources:

- traffic whose source is node $i_{\max}$ itself,
- traffic, whose source is some node other than $i_{\max}$, which is being routed through node $i_{\max}$ using the edge $i_{\max} \rightarrow j_{\max}$.

In order to reduce the total traffic on the edge from node $i_{\max}$ to node $j_{\max}$, the following strategies are useful:

- **Strategy 1:** Let there be an edge from node x to node $i_{\max}$ where there does not exist an edge from node x to node $j_{\max}$ as shown in Figure3.3.2.1a. Part of the traffic on the edge $x \rightarrow i_{\max}$ flows on the edge $i_{\max} \rightarrow j_{\max}$. In this case, setting up a new logical edge $x \rightarrow j_{\max}$ is promising since it provides an alternative path for the traffic now flowing on the edge $x \rightarrow i_{\max}$(Figure 3.3.2.1b).



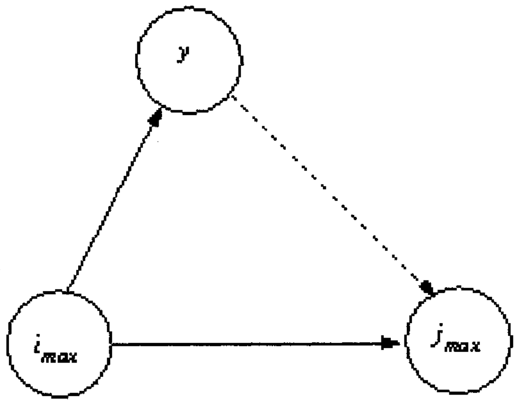
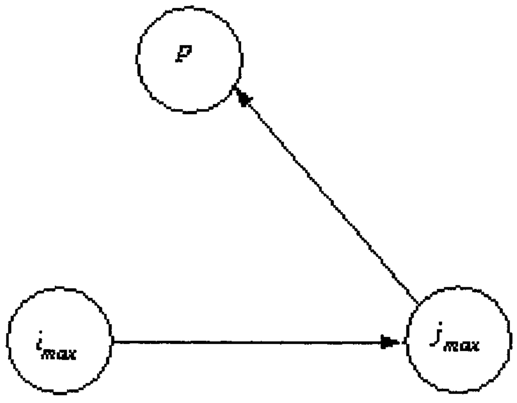


Figure 3.3.2.2a **Figure 3.3.2.2.b**

Figure 3.3.2.2 Adding of Logical Edge Strategy 2

- **Strategy 2:** Let there be an edge from end-node $i_{\max}$ to end-node y where there does not exist an edge from $y \rightarrow j_{\max}$ as shown in Figure 3.3.2.2a. Setting up a new logical edge $y \rightarrow j_{\max}$ is promising since it provides an alternative path for the traffic now flowing on the edge $i_{\max} \rightarrow j_{\max}$(Figure 3.3.2.2b).



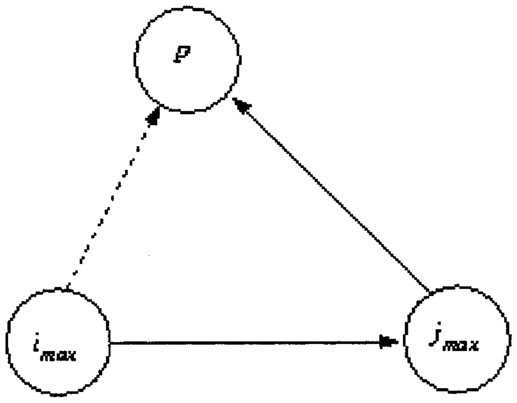


Figure 3.3.2.3a **Figure 3.3.2.3b**

Figure 3.3.2.3 Adding of Logical Edge Strategy 3

- **Strategy 3:** Let there be an edge from end-node $j_{\max}$ to end-node p where there does not exist an edge from $i_{\max} \rightarrow p$ as shown in Figure3.3.2.3a. Part of traffic on the logical edge $j_{\max} \rightarrow p$ may flow on edge $i_{\max} \rightarrow j_{\max}$. Setting up a new logical edge $i_{\max}$→p is promising since it provides an alternative path for the traffic now flowing on the edge $i_{\max}$ →$j_{\max}$ (Figure 3.3.2.3b).

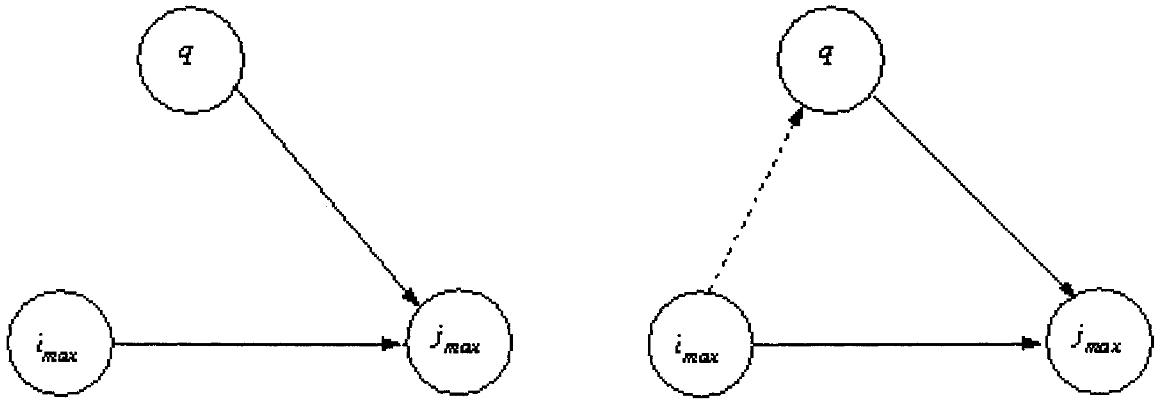


Figure 3.3.2.4a **Figure 3.3.2.4b**

Figure 3.3.2.4 Adding of Logical Edge Strategy 4

➢ **Strategy 4:** Let there be an edge from end-node q to end-node $j_{\max}$ where there does not exist an edge from $i_{\max}$→q as shown in Figure3.3.2.4a. Setting up a new logical edge $i_{\max}$→q is promising since it provides an alternative path for the traffic now flowing on the edge $i_{\max}$ →$j_{\max}$ (Figure3.3.2.4a).

3.4 A block diagram for the search strategy

As a broad overview, the approach can be represented in a simple form as shown below:

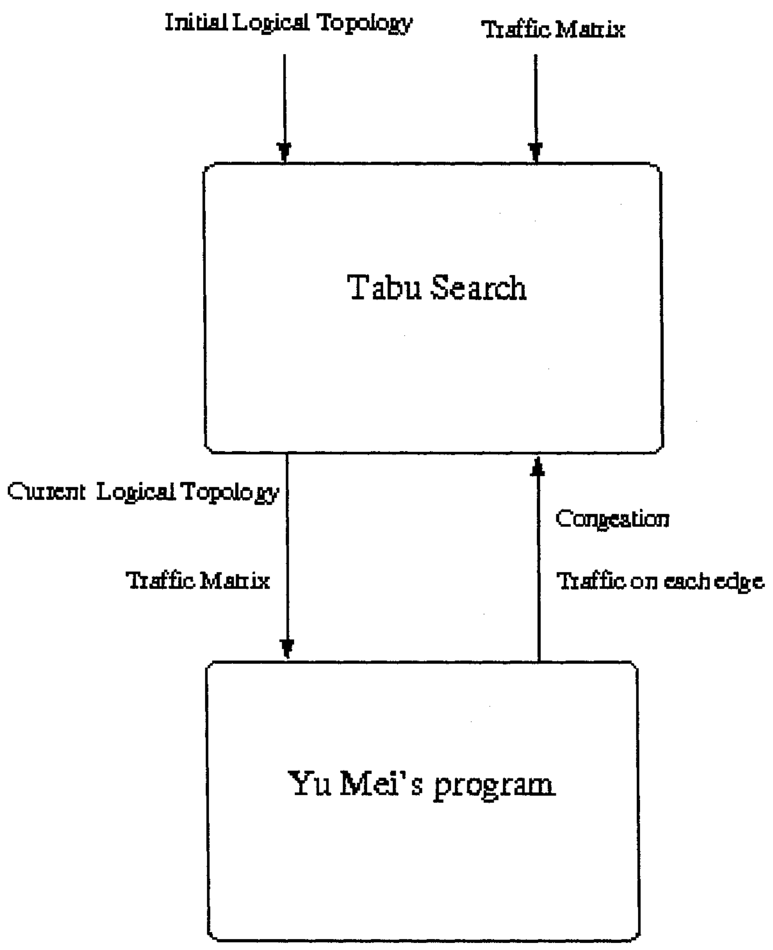


Figure 3.4.1 Block Diagram of the overall scheme

As shown in Figure 3.3.1, the input to the Tabu search program is the traffic matrix and an initial logical topology, computed using some heuristic. The initial logical topology is the current logical topology for the first iteration.

In subsequent iterations, the following tasks have to be carried out:

Step 1) Using Yumei's program, determine the congestion for the current logical topology.

Step 2) Obtain a modified logical topology using the tabu search program to replace the current logical topology.

The above steps are carried out iteratively for a pre-determined number of iterations or until a stopping criterion is met.

For the current logical topology, the edge carrying the minimum traffic is marked for deletion, the possible logical edges to be added are identified as discussed in section 3.3. Each combination of deleting a logical edge and adding a logical edge is identified as a move, and the best move is applied to obtain the modified logical topology. The above approach in this study is identified as move selection procedure. The flow chart representation of the move selection procedure is given below:

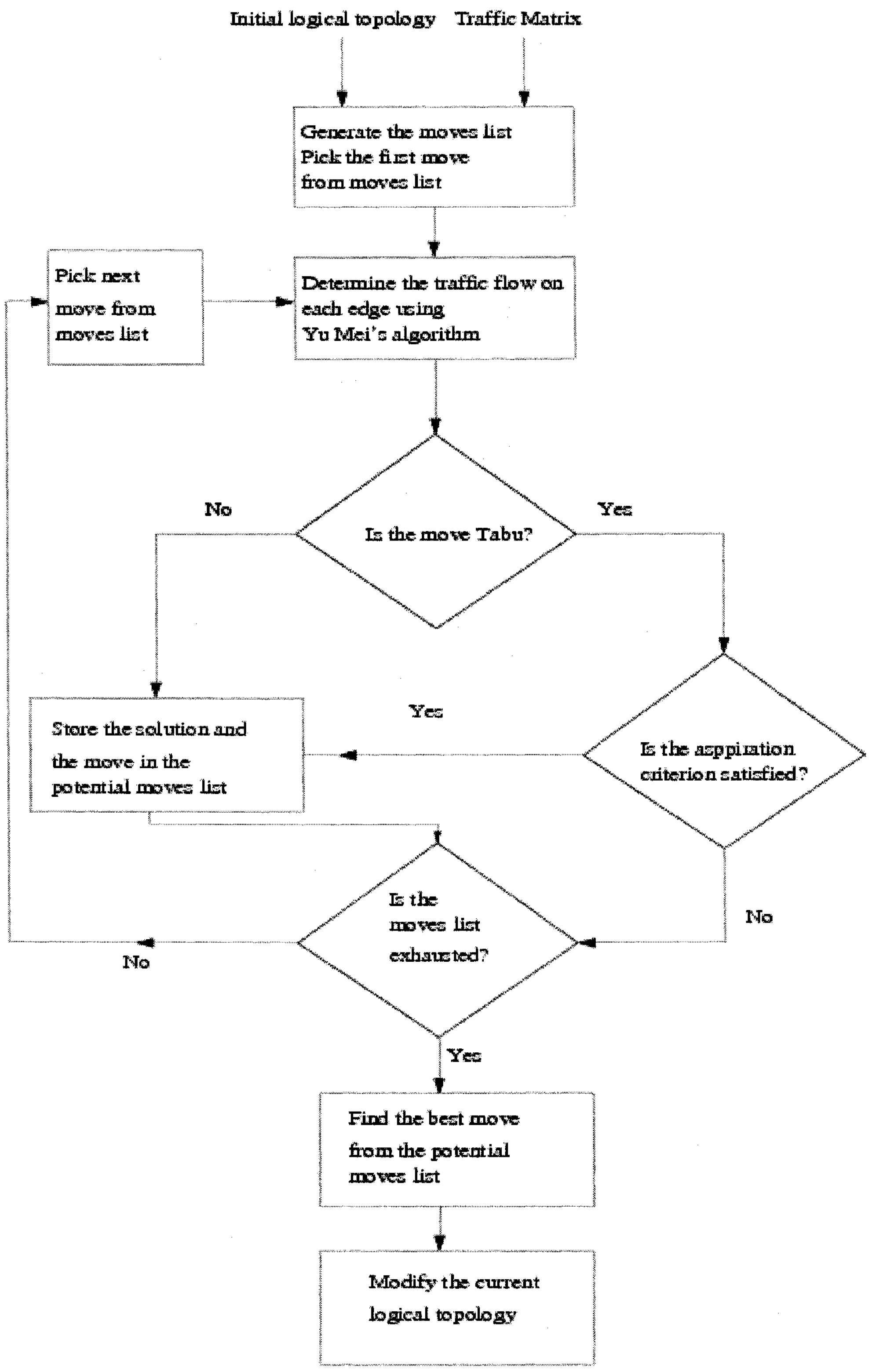


Figure 3.4.2 Move Selection Procedure

The selected move is stored in a tabu list, in other words it is assigned a tabu active status. The modified logical topology thus obtained is now the current logical topology. The current logical topology is again subjected to the move selection procedure as shown in the above Figure 3.4.2. The repeated iterations of above procedure may lead to a situation where numerous additional iterations may not yield an improvement. In such a situation when the congestion value obtained is not less than previously recorded best congestion value, a strategic move is applied which is discussed in detail in Section 4.8. The flowchart for application of strategic move is given below

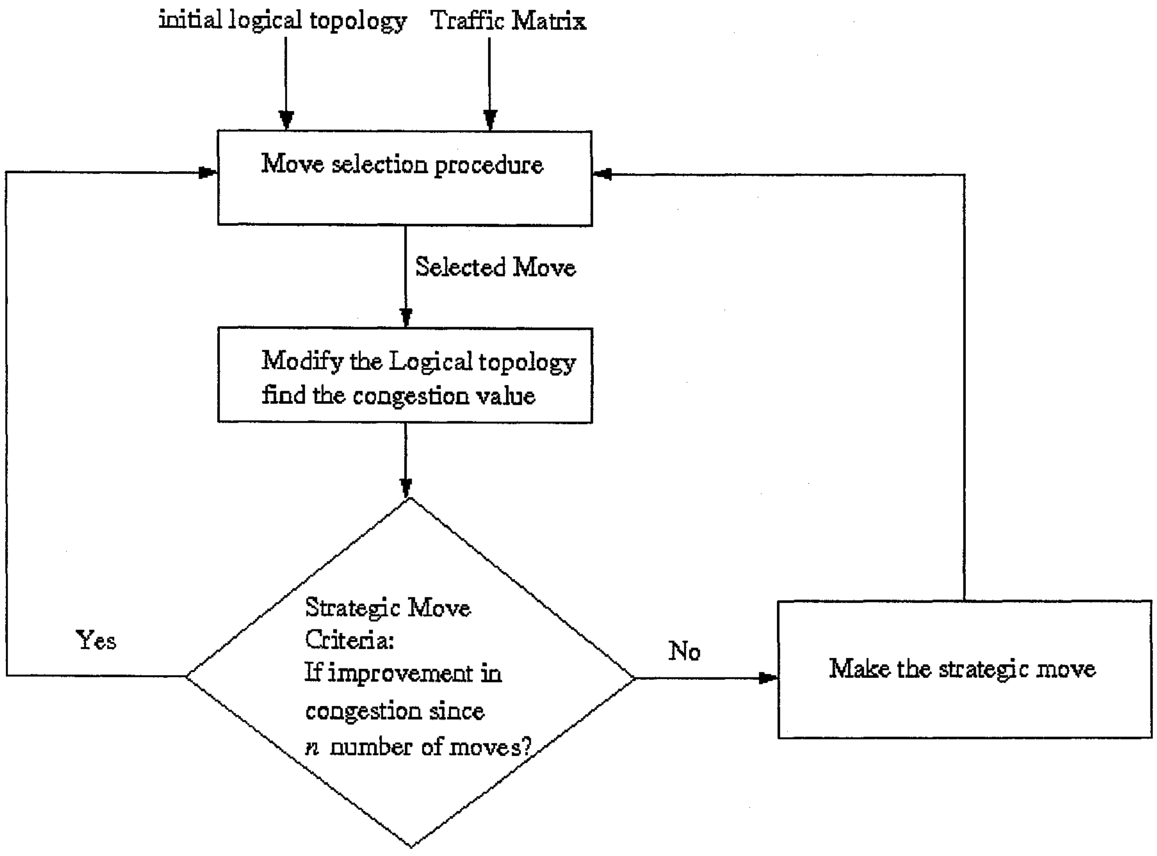

initial logical topology
Traffic Matrix
Move selection procedure
Selected Move
Modify the Logical topology find the congestion value
Strategic Move Criteria: If improvement in congestion since n number of moves?
Yes
No
Make the strategic move

Figure 3.4.3 Strategic Move

The strategic move is made once it is determined that applying the move selection procedure for a large number *n* of moves is making no improvement. The above algorithm is discussed in detail in Chapter 4.

CHAPTER 4

Implementation Details and Experimental Results

In this chapter, the implementation details to solve the problem of logical topology design, as specified in Chapter 3 are discussed. The various aspects of the implementation, discussed in this chapter are:

- generating a traffic matrix,
- determining the initial logical topology,
- determining the traffic flow on each edge in a logical topology,
- generating the moves list for the tabu search,
- forming the tabu list,
- applying the aspiration criterion,
- applying a strategic move.

This chapter also includes the results of our experiments.

4.1 Generating a traffic matrix

The traffic matrix is an *NXN* matrix, where *N* is the number of end-nodes in the network. The traffic matrix entries are generated using the random number generator for the native C compiler. The three different types of traffic matrices generated are given below in a tabular form:

Traffic matrix	Lowest traffic entry	Highest traffic entry
Low	0	20
Medium		40
High		60

Table 4.1.1 Generating Traffic Matrix

The diagonal entries are zero. The remaining entries are generated using the random number generator.

4.2 Determining an initial logical topology

The initial logical topology is generated based on the HLDA [RS 96] algorithm as described in Chapter 2. The input to the HLDA algorithm is the randomly generated traffic matrix and number of transmitters and receivers allowed. The number of transmitters and receivers allocated per node is 5 each for networks with 10 or 14, and for 5 or 6 node networks the allocated transmitters and receivers are 3 each. For instance, in a 10 node network, the total number of available transmitters and receivers is 50.

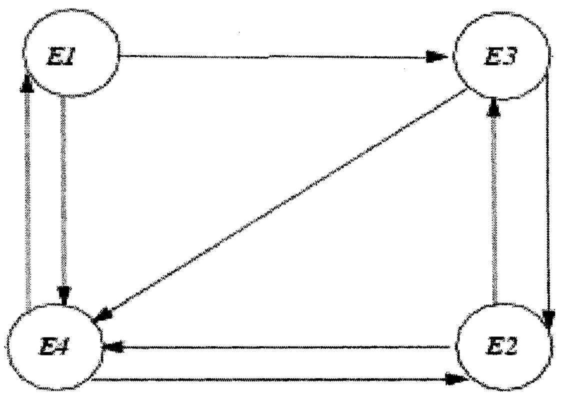
For example consider only 2 transmitters and receivers to be allowed per node and given Traffic Matrix as *T* using the HLDA algorithm, the initial logical topology *L* is generated as shown below.

Traffic Matrix

$$T = \begin{bmatrix} 0 & 2 & 15 & 16 \\ 0 & 0 & 19 & 9 \\ 6 & 8 & 0 & 10 \\ 14 & 14 & 8 & 0 \end{bmatrix}$$

Logical Topology

$$L = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$



Logical Topoplogy gererated using the HLDA algorithm

Figure 4.2.1 Logical Topology using HLDA Algorithm

In determining initial logical topology the highest traffic entry is picked and a logical edge to corresponding (source, destination) pair is added. For example, as shown in Figure 4.2.1, the traffic entry t_{23} is the maximum, hence a logical edge is added from node *E2→E3*. Subsequently the next logical edge is added for the next highest traffic matrix entry, this procedure is carried out till the number of available transceivers are exhausted.

4.3 Determining the traffic flow on each edge

The traffic flow on each edge is determined using Yumei's algorithm discussed in Section 2.6.2. Yumei's algorithm has been modified slightly to return the flow on each edge along with the congestion. The Epsilon value for Yumei's program specifies the approximation bound. For example if the Epsilon value is 0.1 then approximation bound is 10%. The approximation bound used in this investigation was 4%. To speed up the process of determining the congestion, initially a large value of epsilon was specified for the first set of iterations. This allowed quick convergence but the approximation bound was large. In the next set of iterations, the approximation bound was reduced by a factor of 2. This process of successively reducing the approximation bound was continued until the desired approximation bound was reached.

For example, instead of specifying the Epsilon value as 0.04, the initial Epsilon value is specified as 0.64. When the congestion was found with this approximation bound, the Epsilon value is specified as 0.32. This continued until the desired approximation bound of 0.04 was reached. The input to Yumei's program is the traffic matrix and the logical topology. A simple flowchart for the program is given below.

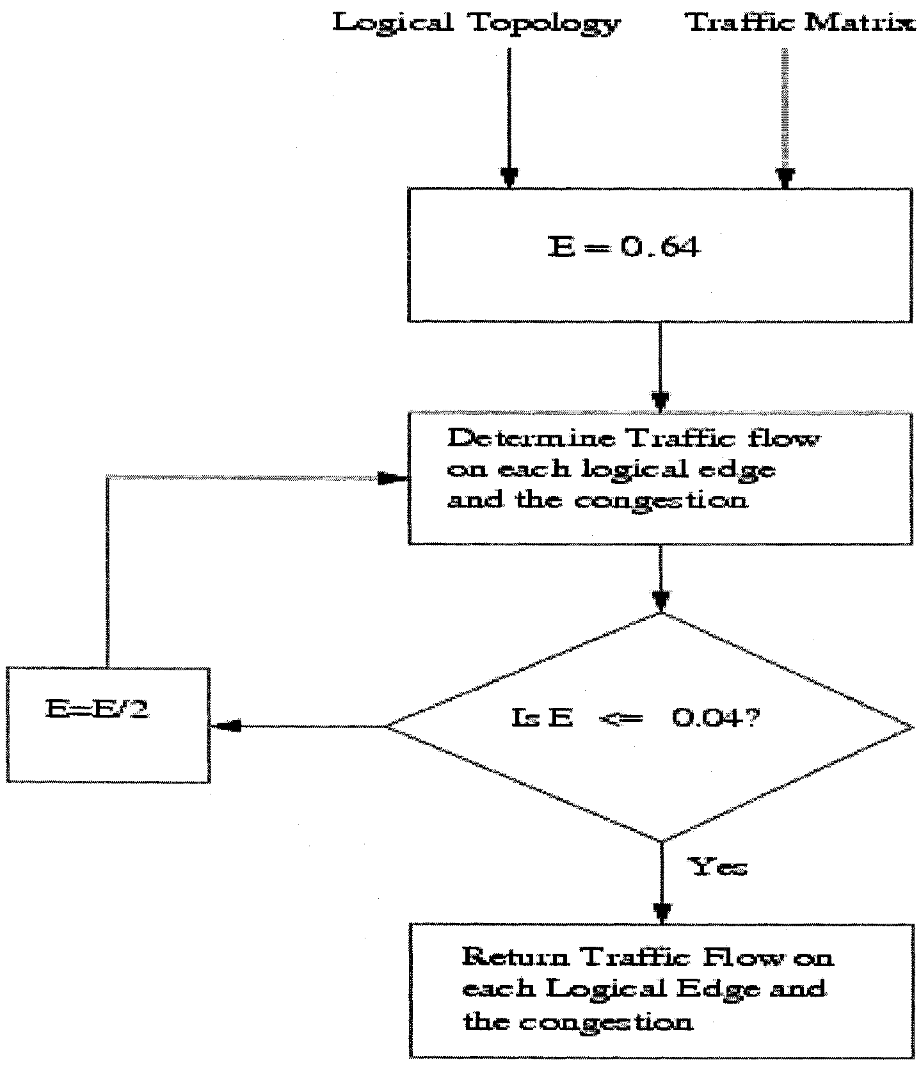

Logical Topology
Traffic Matrix
E = 0.64
Determine Traffic flow on each logical edge and the congestion
Is E <= 0.04?
E=E/2
Yes
Return Traffic Flow on each Logical Edge and the congestion

Figure 4.3.1 Modified Yumei's program

4.4 Generating the moves list

The moves list is generated, based on the concepts discussed in Section 3.3. Let the edge carrying maximum traffic be $i_{max} \rightarrow j_{max}$ and let N be the number of nodes in the network. Let the "incoming source list" be the list of source nodes for which a logical edge exists from an end-node to i_{max}. The details of implementing Strategy1, as discussed in Section 3.3, are represented in the form of flowchart below:

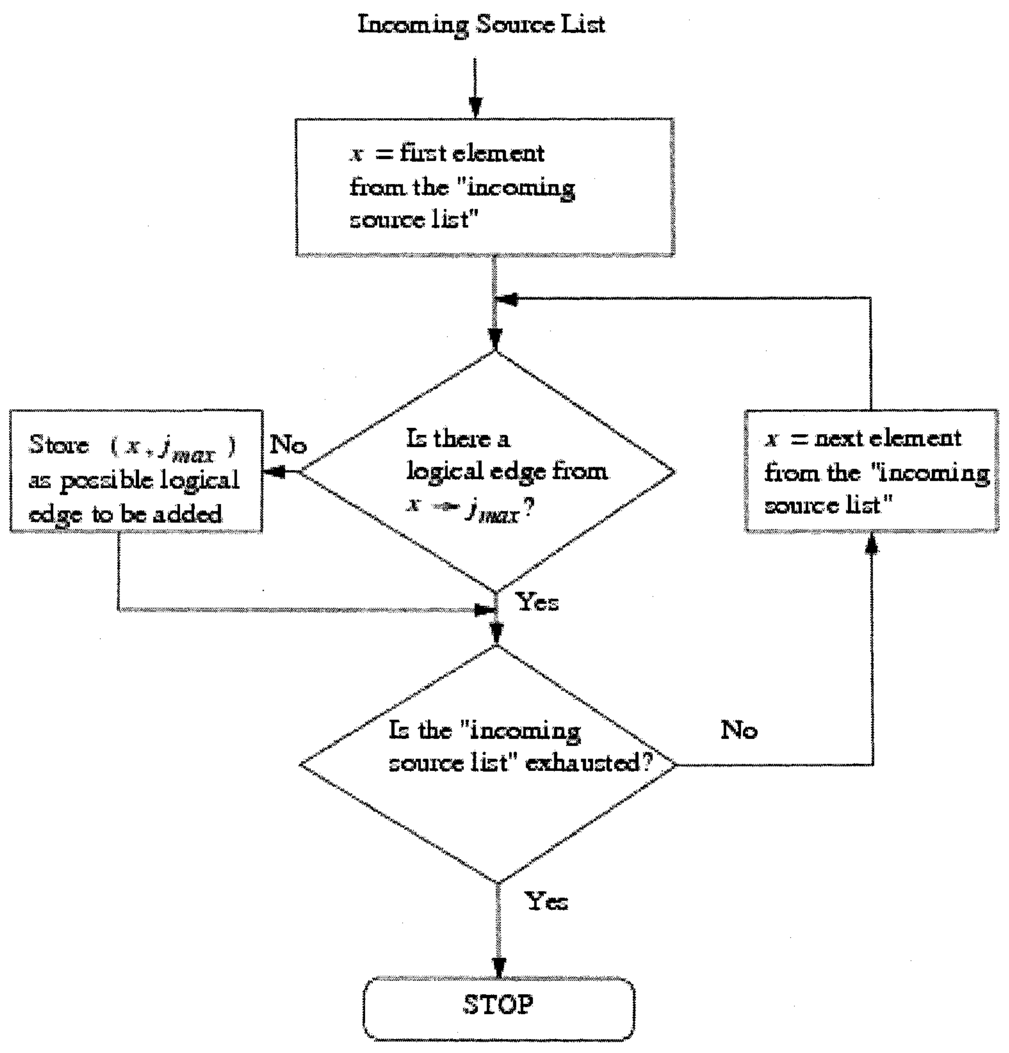

Incoming Source List
x = first element from the "incoming source list"
Is there a logical edge from $x \to j_{max}$?
No
Store (x, j_{max}) as possible logical edge to be added
Yes
x = next element from the "incoming source list"
Is the "incoming source list" exhausted?
No
Yes
STOP

Figure 4.4.1 Implementation Details for strategy 1

Let the "outgoing source list" be list of source nodes for which a logical edge exists from i_{max} to an end-node (excepting j_{max}). Implementation details of Strategy2 as discussed in Section 3.3 are represented in form of flowchart below:

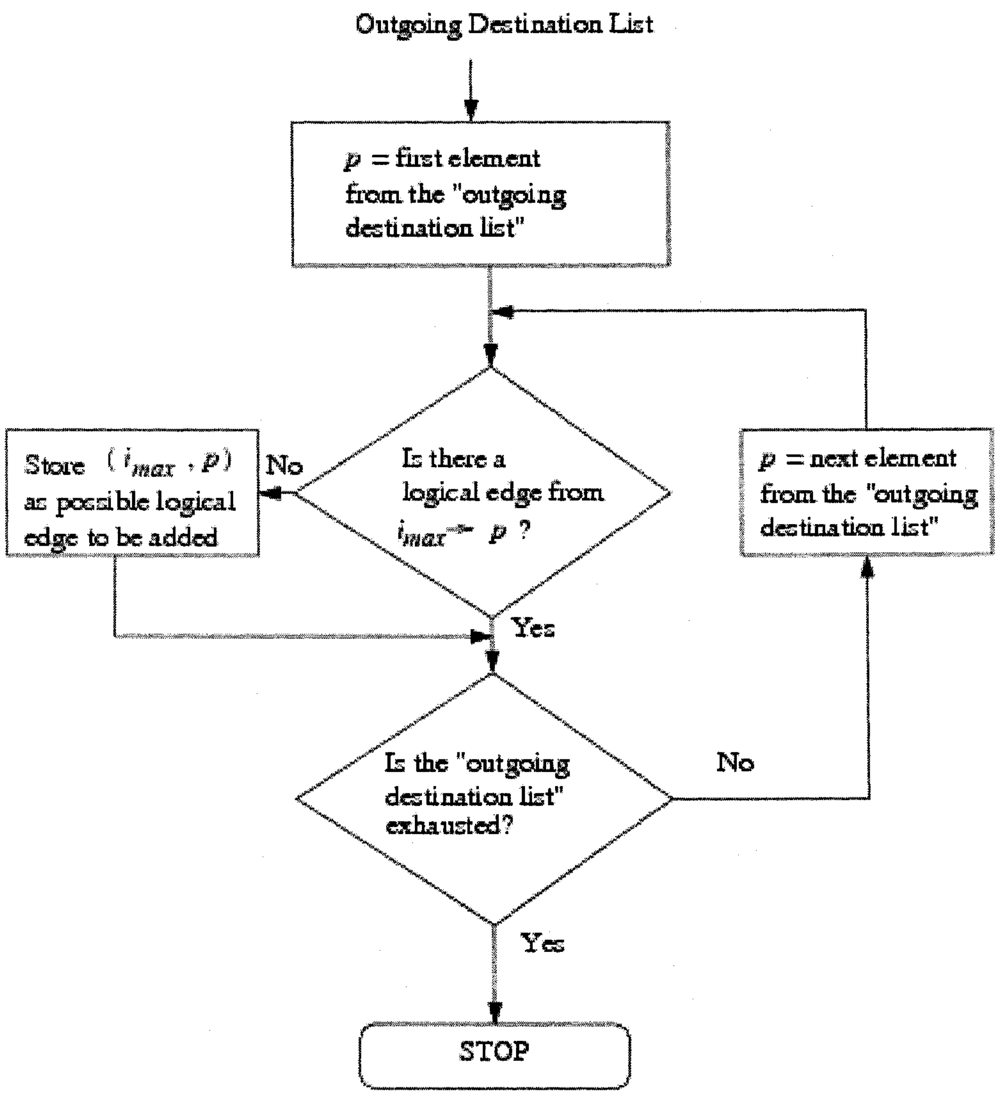

Outgoing Destination List
p = first element from the "outgoing destination list"
Is there a logical edge from $i_{max} \rightarrow p$?
No
Store (i_{max}, p) as possible logical edge to be added
Yes
p = next element from the "outgoing destination list"
Is the "outgoing destination list" exhausted?
No
Yes
STOP

Figure 4.4.2 Implementation Details for strategy 2

The strategy 1 and strategy 2 are shown in the diagram below:

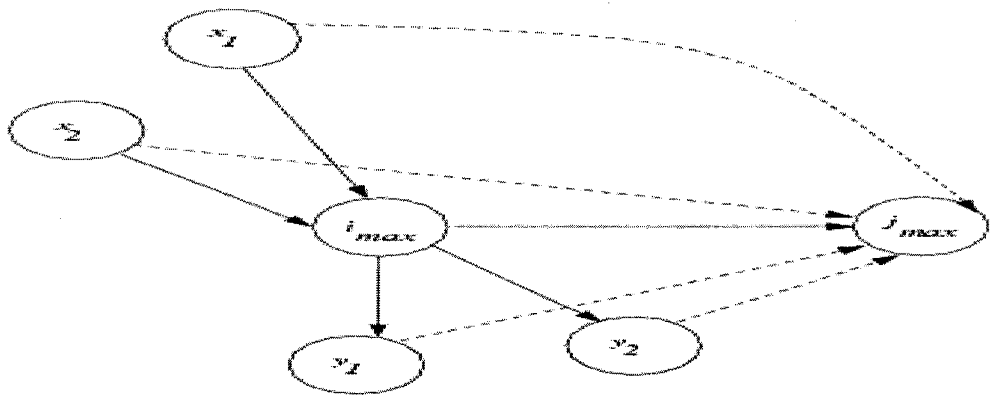


Figure 4.4.3 Finding possible sources

Let "outgoing destination list" be list of source nodes for which a logical edge exists from j_{max} to an end-node. Implementation details of Strategy3 as discussed in Section 3.3 are represented in form of flowchart below:

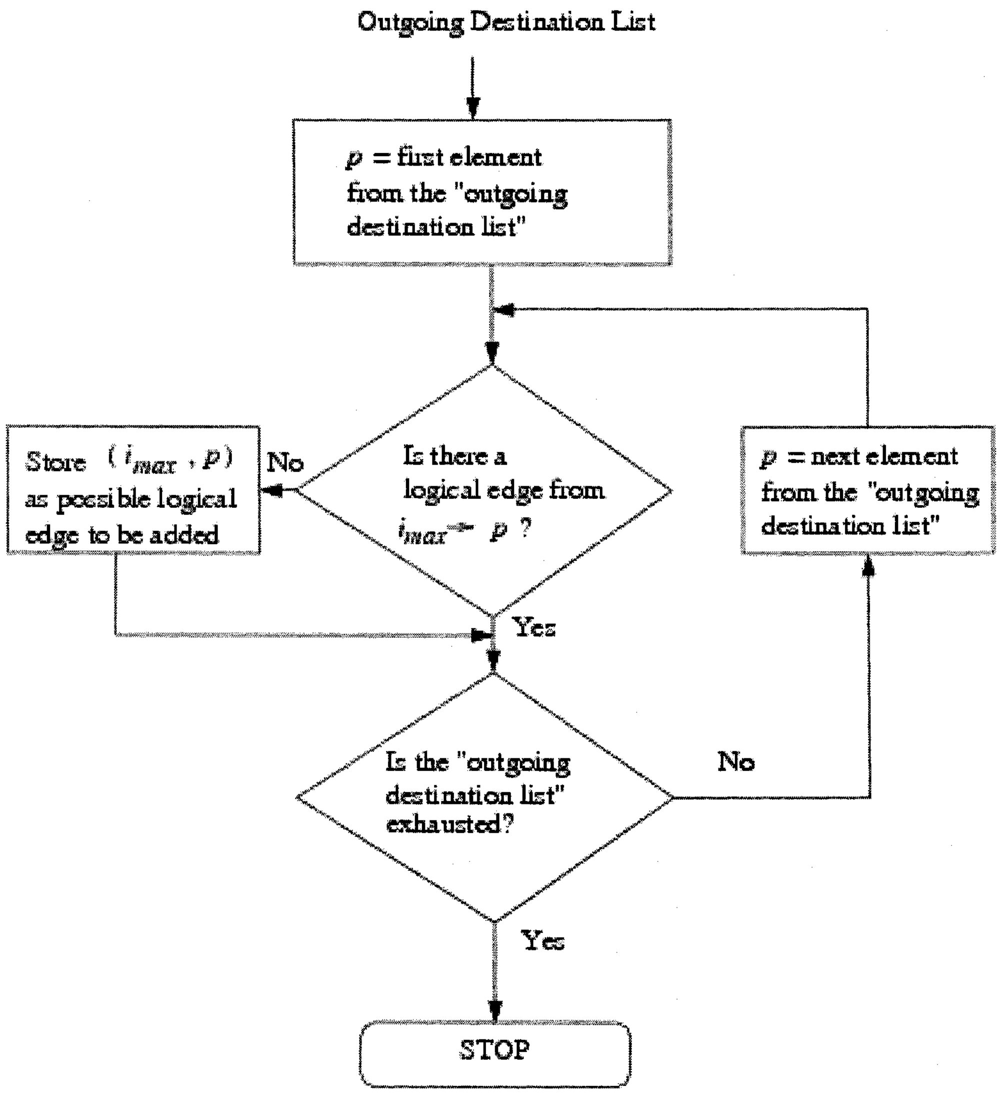

Outgoing Destination List
p = first element from the "outgoing destination list"
Is there a logical edge from $i_{max} \rightarrow p$?
No
Store (i_{max} , p) as possible logical edge to be added
Yes
p = next element from the "outgoing destination list"
Is the "outgoing destination list" exhausted?
No
Yes
STOP

Figure 4.4.4 Implementation Details for strategy 3

Let "incoming destination list" be list of end-nodes for which a logical edge exists from the end-node (excepting i_{max}) to j_{max}. Implementation details of Strategy 4 as discussed in Section 3.3 are represented in form of flowchart below:

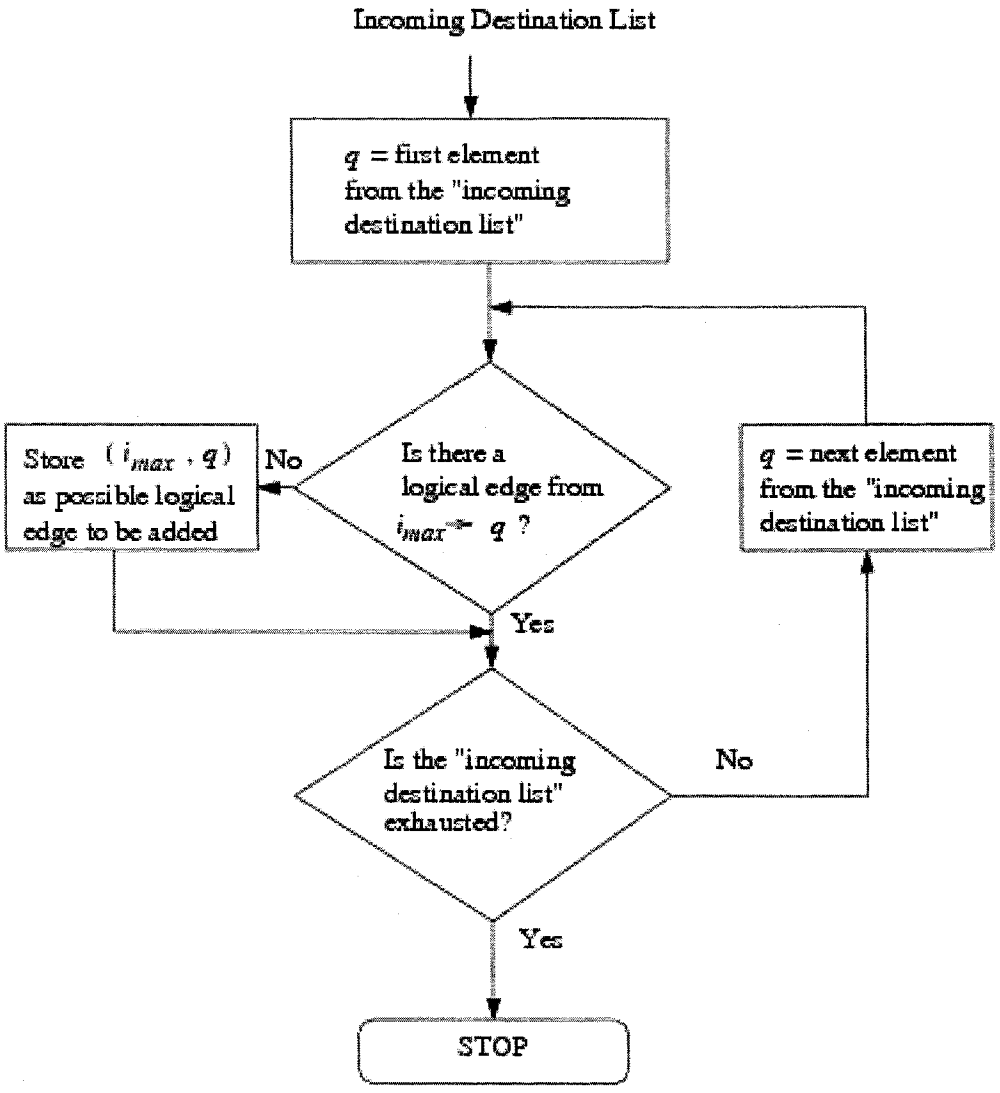

Incoming Destination List
q = first element from the "incoming destination list"
Is there a logical edge from $i_{max} \rightarrow q$?
No
Store (i_{max}, q) as possible logical edge to be added
Yes
q = next element from the "incoming destination list"
Is the "incoming destination list" exhausted?
No
Yes
STOP

Figure 4.4.5 Implementation Details for strategy 4

The strategy 3 and strategy 4 are shown in the diagram below:

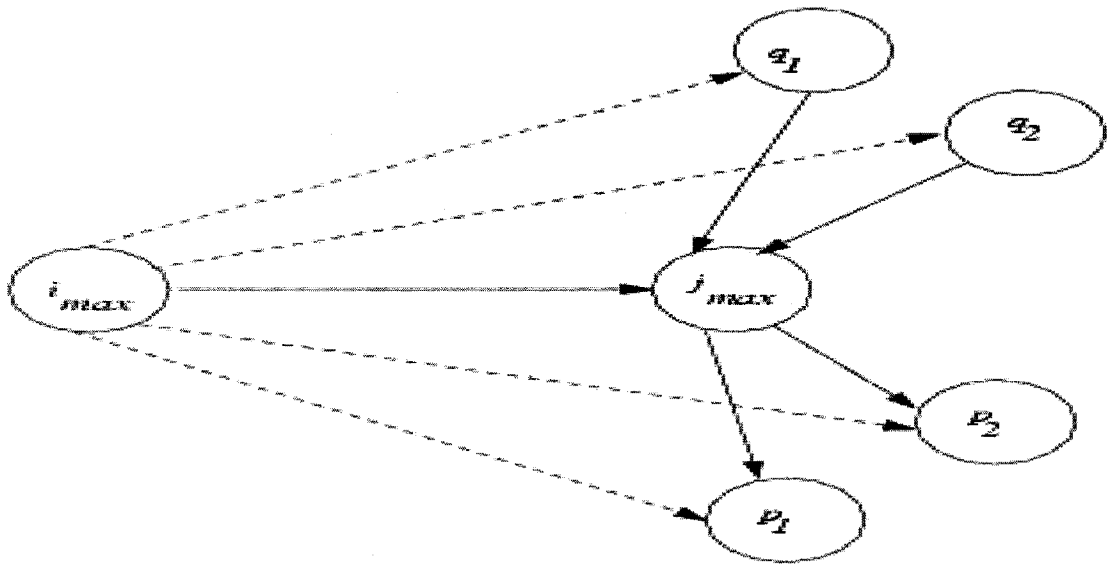


Figure 4.4.6 Finding possible destinations

The concept discussed above in this section defines the moves list. The move as discussed in Section 3.3 has two sub-parts:

i) deleting a logical edge and

ii) adding a logical edge.

The edge that carries the minimum traffic is to be deleted and possible logical edges that can be added are determined in Section 4. For example let $i_{min} \rightarrow j_{min}$ be the logical edge carrying the minimum traffic. With reference to discussion in Section 4.4 one of the possible moves would be

i) Delete the logical edge $i_{min} \rightarrow j_{min}$ and

ii) Add the logical edge $x_1 \rightarrow j_{max}$

4.5 Forming the tabu list

Once a move has been successfully applied, it is required to assign a tabu active status to the move. It is required to assign tabu status to both parts of the move i.e. the deletion of the logical edge and the addition of the logical edge. The tabu status that need to assigned are:

- *add-tabu* (the logical edge deleted in the move cannot be added to the logical topology) and
- *delete-tabu* (the logical edge added in the move cannot be deleted from the logical topology)

For example, if the logical edge $i_{min} \rightarrow j_{min}$ is deleted, it implies that the logical edge $i_{min} \rightarrow j_{min}$ is assigned an add-tabu, (i.e. the logical edge $i_{min} \rightarrow j_{min}$ cannot be added to the logical topology for specified tabu tenure).

The tabu tenure in this case is simply the number of iterations a move is in the tabu list. The tabu tenures that have been experimented in this thesis are 5, 7 and 10. The two strategies of applying the tabu have been described below.

4.5.1 "Add or Delete" tabu strategy

In this strategy, the deletion of a logical edge and the addition of a logical edge are considered to be tabu independently. Considering the Figure 4.4.1, if a move consists of deleting logical edge $i_{min} \rightarrow j_{min}$ and adding logical edge $x_l \rightarrow j_{max}$, then logical edge $i_{min} \rightarrow j_{min}$ is tabu and logical edge $x_l \rightarrow j_{max}$ is also tabu. The logical edge $i_{min} \rightarrow j_{min}$ is added to the add tabu list. Similarly, if the logical edge $x_l \rightarrow j_{max}$ is added, it implies that the logical edge $x_1 \rightarrow j_{max}$ is assigned a delete-tabu (i.e. the logical edge $x_1 \rightarrow j_{max}$ cannot be deleted from the logical topology for the specified tabu tenure). The logical edge $x_1 \rightarrow j_{max}$ is added to the delete tabu list.

This means that

- a move that involves deleting the logical edge $x_1 \rightarrow j_{max}$ or
- a move which involves adding the logical edge $i_{min} \rightarrow j_{min}$

is not allowed for the specified tabu tenure.

In this approach, before the tabu tenure expires, it is not possible to delete the logical edge $x_1 \rightarrow j_{max}$ and to add a different logical edge. This strategy requires two separate tabu lists as follows:

- storing the deleted edge in the first tabu list and
- storing the added edge in the second tabu list.

4.5.2 "Add and Delete" tabu strategy

In this strategy, both the deletion of a logical edge and the addition of a logical edge are considered to be tabu simultaneously. Considering the Figure 4.4.1, if logical edge $i_{min} \rightarrow j_{min}$ is deleted and logical edge $x_1 \rightarrow j_{max}$ is added, then the pair $(i_{min} \rightarrow j_{min}, x_1 \rightarrow j_{max})$ is tabu. This means that a move deleting the logical edge $x_1 \rightarrow j_{max}$ and adding the logical edge $i_{min} \rightarrow j_{min}$ is not possible simultaneously for the specified tabu tenure. It is possible, however, to delete the logical edge $i_{min} \rightarrow j_{min}$ and add a different logical edge.. This strategy requires a single tabu list storing the attributes (add-tabu edge, delete-tabu edge) where add-tabu edge is the logical edge that has been deleted and delete-tabu edge is the edge that has been added.

4.6 Aspiration Criterion

As discussed in Section 2.9.3, the tabu search is restrictive in nature and can sometimes block certain very promising moves. The aspiration criterion is used to allow such promising moves which otherwise would be tabu. In this study, a move that improves over the previous congestion value is said to satisfy the aspiration criterion.

For example, let edge $i_{min} \rightarrow j_{min}$ be deleted in iteration number 10 and let edge $x_1 \rightarrow j_{max}$ be added giving us a congestion of 45 units. In iteration 11 it is found that edge deleting edge $i_m \rightarrow j_n$ and adding edge $i_{min} \rightarrow j_{min}$ gives a congestion value of 43 units. Since edge $i_{min} \rightarrow j_{min}$ has tabu active status it may not be possible to add the logical edge, but since the congestion value is an improvement over previous congestion it is possible to override the tabu status. This situation is when aspiration criterion is met and it supersedes the tabu status.

4.7 Strategic moves

In this study a move is admissible if the move is not tabu or if the move satisfies the aspiration criterion. It is possible that, even after a large number of iterations, the search may not yield a lower congestion value than the best congestion value recorded so far. In such a situation, it is required to implement a strategic move to try and explore new regions of search space. Two such strategic moves that have been investigated in this study are given below:

- *Deleting two logical edges and adding two logical edges*. The logical edge carrying the lowest traffic is deleted and the logical edge carrying the second lowest traffic is deleted. A logical edge is added in the neighborhood of the logical edge carrying the highest traffic, following the same logic as discussed in Section 3.2 to obtain a intermediate logical topology $L^{'}$. Routing for the logical topology $L^{`}$ is carried out using Yumei's algorithm and another logical edge is added around the logical edge carrying maximum traffic for logical topology $L^{`}$.
- *Deleting the logical edge with maximum traffic and adding another logical edge.* The logical edge carrying the maximum traffic is deleted and another logical edge is added around the deleted logical edge following the same concept as discussed in Section 3.2

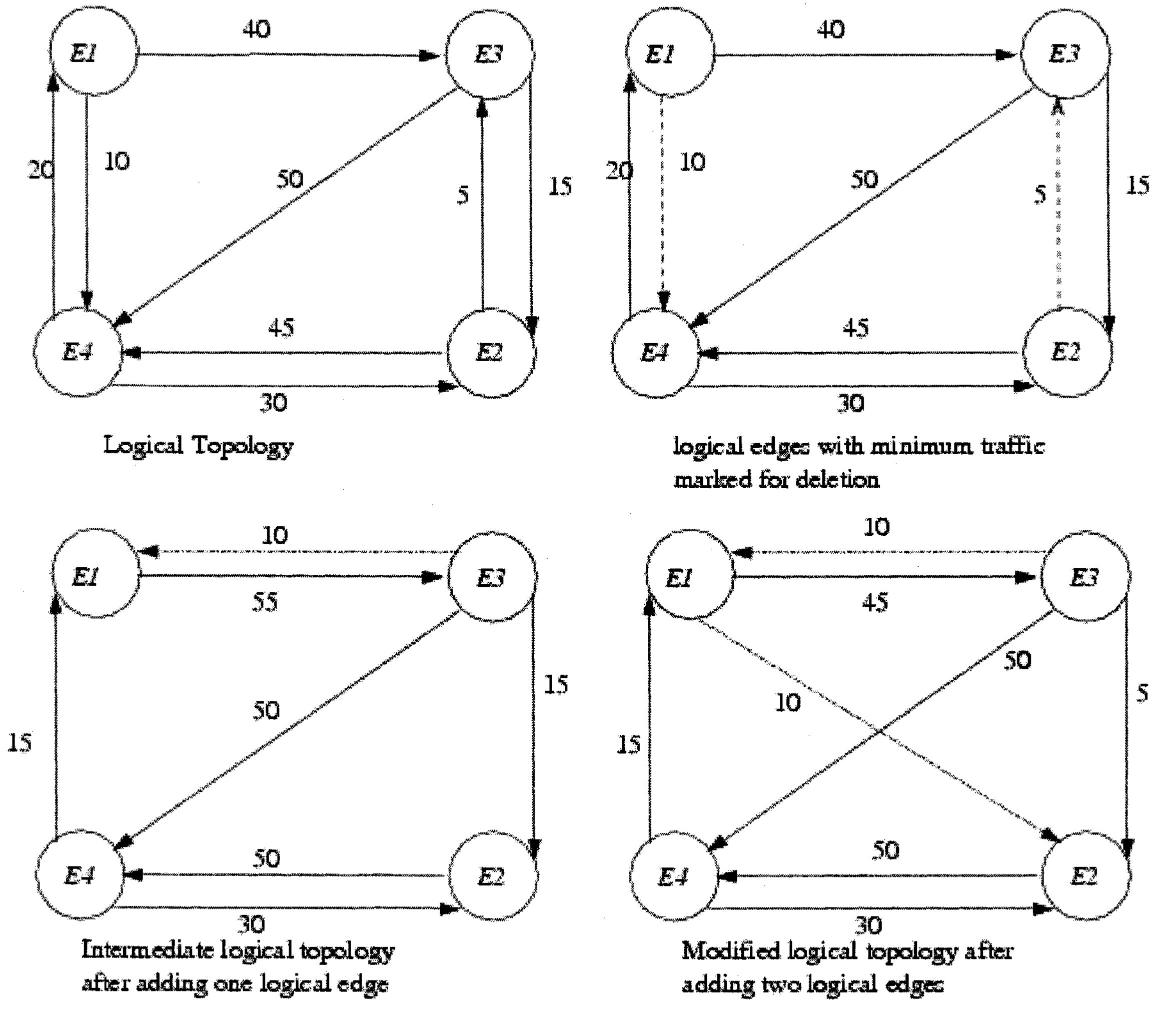

E1
E2
E3
E4
40
20
10
50
5
15
45
30
Logical Topology
logical edges with minimum traffic marked for deletion
10
55
45
15
Intermediate logical topology after adding one logical edge
Modified logical topology after adding two logical edges

Figure 4.7.1 Example illustrating strategic move of deleting two logical edges and adding two logical edges

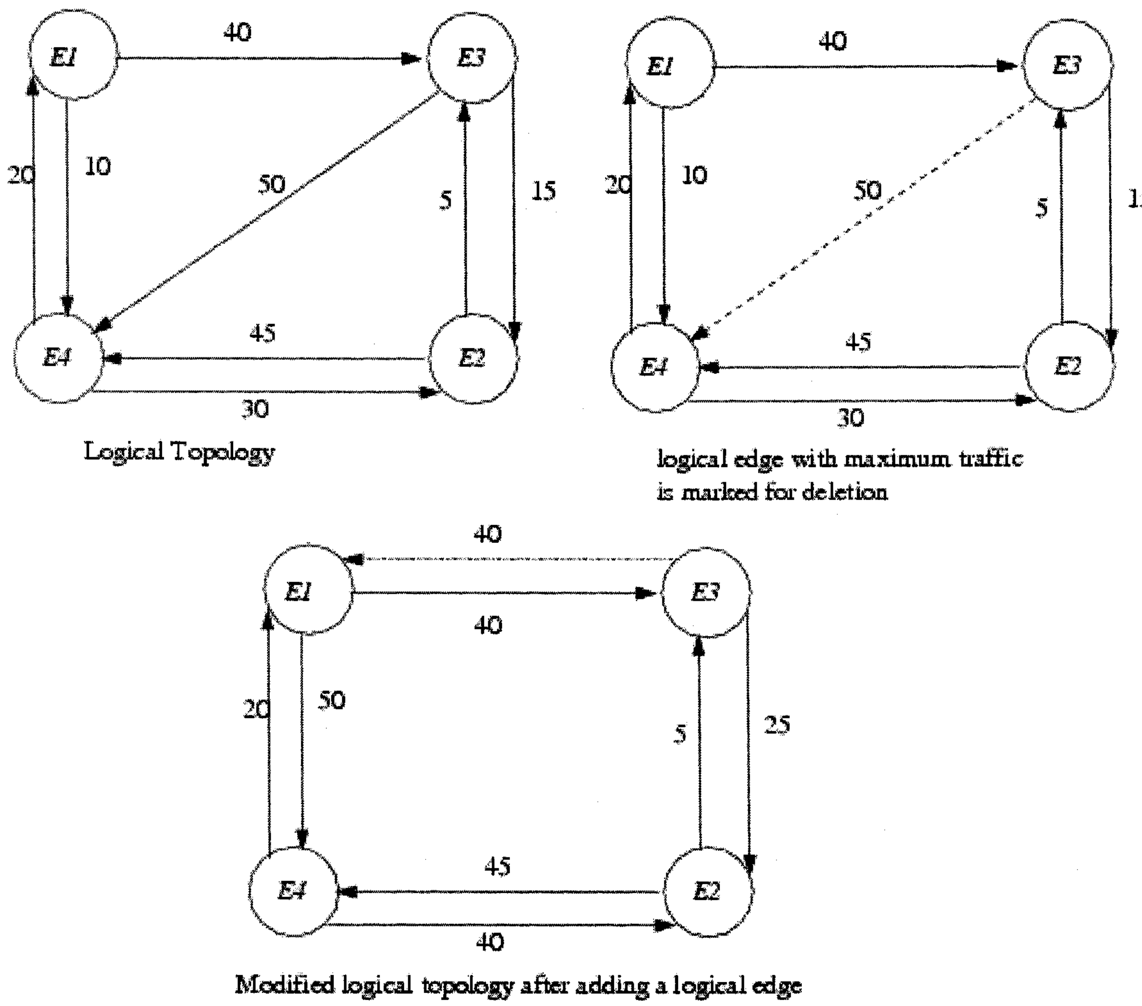

E1
E3
E4
E2
40
20
10
50
5
15
45
30
Logical Topology
E1
E3
E4
E2
40
20
10
50
5
15
45
30
logical edge with maximum traffic is marked for deletion
E1
E3
E4
E2
40
40
20
50
5
25
45
40
Modified logical topology after adding a logical edge

Figure 4.7.2 An example illustrating strategic move of deleting the logical edge with maximum traffic

4.8 Details of experiments

The experiments have been carried out on a large number of networks, ranging from 5 node networks to 14 node networks.

The experiments have been carried out with two distinct ways of attaching a tabu to the move. As explained in Section 4.5 the there are two cases given below:

- ➢ Deleting of a logical edge and adding of a logical edge both are considered tabu simultaneously
- ➢ Deleting of a logical edge and adding of a logical edge are considered tabu separately

The changing of tabu tenure affects the search process. Different tabu tenures that have been experimented with are:

- ➢ Tabu Tenure of 5
- ➢ Tabu Tenure of 7
- ➢ Tabu Tenure of 10

To investigate new regions in the search space two different strategic moves have been experimented with as mentioned below:

- ➢ Deleting two logical edges and adding two logical edges
- ➢ Deleting edge with maximum traffic and adding another edge

Each case has been evaluated using a number of different traffic requirement matrices. A different seed value has been used to randomly generate the different traffic requirement matrices. The two parameters which have been compared for are

- ➢ Quality of the solution i.e. how much has the congestion value improved as compared to initial congestion and
- ➢ Performance of the algorithms i.e. how quickly can the solution be found

The 95% confidence interval has been computed for all the test cases with a sample size of 5. The following formula has been used to calculate the 95% confidence interval

Let *s* be the sample size which in this thesis is 5

$x_1, x_2 \dots x_s$ be the percentage improvement in the congestion value

$\bar{x}$ be the mean of $x_1, x_2 \dots x_s$.

t be the constant determined from tables which is 2.571

σ be the standard deviation where $\sigma = \sqrt{\sum (x - \bar{x})^2} \, / \, (s\text{-}1)$

The confidence interval is determined by the formula $\bar{x} \pm t * \sigma / \sqrt{s}$

The experimental results have been presented below

4.9 Experimental Results

Case 1:

- tabu move: deletion of a logical edge or addition of a logical edge
- strategic move: deletion of two logical edges and addition of two logical edges

Number of nodes	Number of edges	Traffic Matrix	Tabu Tenure	Initial Congestion	Final Congestion	%impro -vement	95% confidence interval	Time Taken (seconds)
5	15	Low	5	17.040817	16.205788	4.9002	4.9±0.845	157
		Medium		37.047405	34.912819	5.7618	5.76±0.549	245
		High		57.017803	53.142857	6.796	6.8±0.556	262
		Low	7	17.040817	16.198348	4.9438	4.94±0.704	160
		Medium		37.047405	35.069767	5.3381	5.34±0531	204
		High		57.017803	53.040180	6.9761	6.8±1.007	226
		Low	10	17.040817	16.238356	4.7091	4.71±.609	180
		Medium		37.047405	35.253719	4.8416	4.84±0.561	228
		High		57.017803	53.261437	6.5881	6.59±0.584	230

Average percentage gain 5.6505

Table 4.9.1 A 5-node network experimental results

Number of nodes	Number of edges	Traffic Matrix	Tabu Tenure	Initial Congestion	Final Congestion	%impro -vement	95% confidence interval	Time Taken (seconds)
6	18	Low	5	26.126184	23.01309	11.9156	11.916±0.932	411
		Medium		54.852764	47.87744	12.7164	12.716±0.817	409
		High		76.049385	68.40659	10.0497	10.05±0.772	488
		Low	7	26.126184	23.16365	11.3393	11.339±0.861	452
		Medium		54.852764	46.96104	14.3871	14.387±0.665	392
		High		76.049385	67.95429	10.6445	10.645±0.734	526
		Low	10	26.126184	22.92434	12.2553	12.255±1.103	417
		Medium		54.852764	48.73342	11.1559	11.159±0.684	394
		High		76.049385	69.12435	9.1059	9.106±0.594	537

Average percentage gain 11.50774

Table 4.9.2 A 6-node network experimental results

Number of nodes	Number of edges	Traffic Matrix	Tabu Tenure	Initial Congestion	Final Congestion	%impro -vement	95% confidence interval	Time Taken (seconds)
10	50	Low	5	22.1866	20.043265	9.6606	9.66±0.876	2899
		Medium		46.5619	41.238214	11.434	11.434±1.005	3757
		High		74.0494	64.193468	13.31	13.31±0.946	3827
		Low	7	22.1866	20.051624	9.6229	9.623±1.106	2986
		Medium		46.5619	41.621136	10.611	10.611±0.784	3797
		High		74.0494	64.41253	13.014	13.014±0.962	3889
		Low	10	22.1866	20.076231	9.512	9.512±0.698	3350
		Medium		46.5619	41.318294	11.262	11.262±1.002	3824
		High		74.0494	63.465768	14.293	14.293±0.918	3936

Average percentage gain 11.41328

Table 4.9.3 A 10-node network experimental results

Number of nodes	Number of edges	Traffic Matrix	Tabu Tenure	Initial Congestion	Final Congestion	%impro -vement	95% confidence interval	Time Taken (seconds)
14	70	Low	5	47.217392	41.165476	12.817	12.817±0.982	9630
		Medium		85.133331	75.238695	11.623	11.623±0.872	9764
		High		150.77777	130.65843	13.344	13.344±1.103	10034
		Low	7	47.217392	40.358267	14.527	14.527±.906	9941
		Medium		85.133331	75.176823	11.695	11.695±0.946	10094
		High		150.77777	129.67464	13.996	13.996±0.896	10284
		Low	10	47.217392	40.974582	13.221	13.221±0.846	10110
		Medium		85.133331	75.216438	11.649	11.649±.962	10280
		High		150.77777	131.02638	13.1	13.1±0.758	10568

Average percentage gain 12.88578

Table 4.9.4 A 14-node network experimental results

Case 2:

- tabu move: deletion of a logical edge and addition of a logical edge
- strategic move: deletion of two logical edges and addition of two logical edges

Number of nodes	Number of edges	Traffic Matrix	Tabu Tenure	Initial Congestion	Final Congestion	%impro -vement	95% confidence interval	Time Taken (seconds)
5	15	Low	5	17.040817	16.203873	4.9114	4.911±0.634	170
		Medium		37.047405	35.321964	4.6574	4.657±0.812	252
		High		57.017803	53.745647	5.7388	5.739±0.663	280
		Low	7	17.040817	16.112346	5.4485	5.448±0.786	158
		Medium		37.047405	35.967367	2.9153	2.915±0.462	240
		High		57.017803	54.243608	4.8655	4.865±0.824	246
		Low	10	17.040817	16.837216	1.1948	1.195±0.446	172
		Medium		37.047405	35.833953	3.2754	3.275±.0.628	236
		High		57.017803	54.425778	4.546	5.546±0.858	248

Average percentage gain 4.172567

Table 4.9.5 A 5-node network experimental results

Number of nodes	Number of edges	Traffic Matrix	Tabu Tenure	Initial Congestion	Final Congestion	%impro -vement	95% confidence interval	Time Taken (seconds)
6	18	Low	5	26.126184	23.942156	8.3595	8.359±0.843	432
		Medium		54.852764	48.764235	11.1	11.1±0.724	440
		High		76.049385	70.875342	6.8035	6.803±.937	486
		Low	7	26.126184	23.963683	8.2771	8.277±1.001	462
		Medium		54.852764	49.984312	8.8755	8.875±0.871	448
		High		76.049385	69.764982	8.2636	8.264±0.943	538
		Low	10	26.126184	24.384877	6.665	6.665±0.756	437
		Medium		54.852764	50.341824	8.2237	8.224±.0.902	427
		High		76.049385	70.434688	7.383	7.383±0.936	560

Average percentage gain 8.216767

Table 4.9.6 A 6-node network experimental results

Number of nodes	Number of edges	Traffic Matrix	Tabu Tenure	Initial Congestion	Final Congestion	%impro -vement	95% confidence interval	Time Taken (seconds)
10	50	Low	5	22.1866	20.946215	5.5907	5.591±0.623	2941
		Medium		46.5619	43.214863	7.1884	7.188±0.844	3094
		High		74.0494	68.946486	6.8912	6.891±0.847	3284
		Low	7	22.1866	21.016678	5.2731	5.273±0.692	2963
		Medium		46.5619	43.364754	6.8664	6.866±0.794	3102
		High		74.0494	68.425163	7.5952	7.595±0.907	3321
		Low	10	22.1866	21.761458	1.9162	1.9162±0.432	3017
		Medium		46.5619	43.975684	5.5544	5.554±.0.872	3468
		High		74.0494	68.573687	7.3947	7.395±0.974	3762

Average percentage gain 6.030033

Table 4.9.7 A 10-node network experimental results

Number of nodes	Number of edges	Traffic Matrix	Tabu Tenure	Initial Congestion	Final Congestion	%impro -vement	95% confidence interval	Time Taken (seconds)
14	70	Low	5	47.217392	43.432716	8.0154	8.015±0.823	9618
		Medium		85.133331	77.348965	9.1437	9.144±0.946	9774
		High		150.77777	134.83472	10.574	10.574±1.007	9982
		Low	7	47.217392	43.846378	7.1393	1.139±0.986	9762
		Medium		85.133331	77.726348	8.7005	8.700±0.942	10032
		High		150.77777	134.46239	10.821	10.821±0.817	10198
		Low	10	47.217392	43.942271	6.9363	6.936±0.841	10126
		Medium		85.133331	77.821547	8.5886	8.588±.0.972	10256
		High		150.77777	133.87453	11.211	11.211±1.001	10506

Average percentage gain 9.014422

Table 4.9.8 A 14-node network experimental results

Case 3:

- tabu move: deletion of a logical edge or addition of a logical edge
- strategic move: deleting the logical edge with maximum traffic and adding a logical edge in the neighborhood

Number of nodes	Number of edges	Traffic Matrix	Tabu Tenure	Initial Congestion	Final Congestion	%impro -vement	95% confidence interval	Time Taken (seconds)
5	15	Low	5	17.040817	15.988632	6.1745	6.174±0.756	160
		Medium		37.047405	34.586819	6.6417	6.642±0.823	220
		High		57.017803	52.867832	7.2784	7.278±0.816	258
		Low	7	17.040817	16.064587	5.7288	5.729±0.798	171
		Medium		37.047405	34.943567	5.6788	5.679±0.912	227
		High		57.017803	53.036428	6.9827	6.983±0.742	234
		Low	10	17.040817	16.193368	4.9731	4.973±0.641	177
		Medium		37.047405	35.158319	5.0991	5.099±.0.721	230
		High		57.017803	53.113872	6.8469	6.847±0.801	235

Average percentage gain 6.156

Table 4.9.9 A 5-node network experimental results

Number of nodes	Number of edges	Traffic Matrix	Tabu Tenure	Initial Congestion	Final Congestion	%impro -vement	95% confidence interval	Time Taken (seconds)
6	18	Low	5	26.126184	22.013093	15.743	15.743±0.964	402
		Medium		54.852764	46.87744	14.54	14.54±0.936	416
		High		76.049385	65.406593	13.995	13.995±1.002	468
		Low	7	26.126184	22.163653	15.167	15.167±1.108	432
		Medium		54.852764	46.961041	14.387	14.387±0.982	419
		High		76.049385	65.954285	13.274	13.374±0.958	479
		Low	10	26.126184	22.924337	12.255	12.255±0.891	422
		Medium		54.852764	46.733418	14.802	14.802±.1.006	428
		High		76.049385	66.124346	13.051	13.051±1.105	503

Average percentage gain 14.135

Table 4.9.10 A 6-node network experimental results

Number of nodes	Number of edges	Traffic Matrix	Tabu Tenure	Initial Congestion	Final Congestion	%impro -vement	95% confidence interval	Time Taken (seconds)
10	50	Low	5	22.1866	20.035782	9.6942	9.694±0.972	2834
		Medium		46.5619	40.821354	12.329	12.329±0.984	3736
		High		74.0494	63.346458	14.454	14.454±1.003	3840
		Low	7	22.1866	20.016429	9.7814	9.7814±0.978	2972
		Medium		46.5619	40.213862	13.634	13.634±0.954	3810
		High		74.0494	63.453478	14.309	14.309±1.112	3908
		Low	10	22.1866	20.062407	9.5742	9.574±0.876	3298
		Medium		46.5619	40.161284	13.746	13.746±0.998	3792
		High		74.0494	63.279354	14.544	14.544±1.103	3958

Average percentage gain 12.45176

Table 4.9.11 A 10-node network experimental results

Number of nodes	Number of edges	Traffic Matrix	Tabu Tenure	Initial Congestion	Final Congestion	%impro -vement	95% confidence interval	Time Taken (seconds)
14	70	Low	5	47.217392	40.652337	13.904	13.904±1.012	9806
		Medium		85.133331	73.385675	13.799	13.799±1.084	9982
		High		150.77777	129.31486	14.235	14.235±0.993	10046
		Low	7	47.217392	40.036724	15.208	15.208±1.078	9843
		Medium		85.133331	72.236714	15.149	15.149±0.984	9996
		High		150.77777	128.42614	14.824	14.824±0.992	10376
		Low	10	47.217392	40.024536	15.233	15.233±1.046	10112
		Medium		85.133331	72.976231	14.28	14.28±1.015	10295
		High		150.77777	128.72547	14.626	14.626±1.033	10643

Average percentage gain 14.58422

Table 4.9.12 A 14-node network experimental results

Case 4:

- tabu move: deletion of a logical edge and addition of a logical edge
- strategic move: deletion of two logical edges and addition of two logical edges

Number of nodes	Number of edges	Traffic Matrix	Tabu Tenure	Initial Congestion	Final Congestion	%impro -vement	95% confidence interval	Time Taken (seconds)
5	15	Low	5	17.040817	16.453758	3.445	3.445±.652	167
		Medium		37.047405	35.121429	5.1987	5.199±0.734	234
		High		57.017803	54.042652	5.2179	5.218±0.813	242
		Low	7	17.040817	16.395372	3.7876	3.788±0.712	161
		Medium		37.047405	35.482467	4.2242	4.224±0.637	212
		High		57.017803	53.30184	6.5172	6.517±0.645	218
		Low	10	17.040817	16.286372	4.4273	4.427±0.706	184
		Medium		37.047405	35.754195	3.4907	3.491±0.695	225
		High		57.017803	53.834565	5.5829	5.583±0.715	239

Average percentage gain 4.6546

Table 4.9.13 A 5-node network experimental results

Number of nodes	Number of edges	Traffic Matrix	Tabu Tenure	Initial Congestion	Final Congestion	%impro -vement	95% confidence interval	Time Taken (seconds)
6	18	Low	5	26.126184	22.935668	12.212	12.212±.0.876	409
		Medium		54.852764	47.634745	13.159	13.159±0.994	421
		High		76.049385	67.063535	11.816	11.816±0.947	477
		Low	7	26.126184	23.065382	11.715	11.715±1.015	397
		Medium		54.852764	48.147293	12.224	12.224±0.997	410
		High		76.049385	67.525346	11.209	11.209±1.009	513
		Low	10	26.126184	22.947589	12.166	12.166±0.985	412
		Medium		54.852764	47.921437	12.636	12.636±0.872	452
		High		76.049385	66.763592	12.21	12.21±1.025	470

Average percentage gain 12.15

Table 4.9.14 A 6-node network experimental results

Number of nodes	Number of edges	Traffic Matrix	Tabu Tenure	Initial Congestion	Final Congestion	%impro -vement	95% confidence interval	Time Taken (seconds)
10	50	Low	5	22.1866	20.385326	8.1187	8.119±.0.764	2828
		Medium		46.5619	41.215416	11.483	11.483±1.003	3747
		High		74.0494	65.217459	11.927	11.927±0.963	3932
		Low	7	22.1866	20.614239	7.087	7.087±0.907	2956
		Medium		46.5619	42.139245	9.4984	9.498±0.893	3805
		High		74.0494	65.537489	11.495	11.495±0.991	3915
		Low	10	22.1866	20.624718	7.0398	7.04±0.851	3123
		Medium		46.5619	42.112432	9.556	9.556±0.817	3719
		High		74.0494	66.035214	10.823	10.823±0.977	3976

Average percentage gain 9.6697

Table 4.9.15 A 10-node network experimental results

Number of nodes	Number of edges	Traffic Matrix	Tabu Tenure	Initial Congestion	Final Congestion	%impro -vement	95% confidence interval	Time Taken (seconds)
14	70	Low	5	47.217392	43.375642	8.1363	8.136±.0.646	10003
		Medium		85.133331	76.852591	9.7268	9.727±0.807	10102
		High		150.77777	134.36442	10.886	10.886±0.935	10364
		Low	7	47.217392	44.033624	6.7428	6.748±0.714	9975
		Medium		85.133331	77.167484	9.3569	9.357±0.911	10081
		High		150.77777	134.92673	10.513	10.513±1.023	10443
		Low	10	47.217392	44.043625	6.7216	6.722±0.885	10115
		Medium		85.133331	76.987145	9.5687	9.569±0.782	10365
		High		150.77777	133.98543	11.137	11.137±0.995	10705

Average percentage gain 9.1988

Table 4.9.16 A 14-node network experimental results

4.10 Analysis

The results of the experiments using tabu search for the logical topology design have been list in the Table 4.9.1 to Table 4.9.16. The above tables show the results of applying a tabu search strategy for the logical topology design. The various cases considered are:

- different traffic scenarios
- the affect of changing tabu tenure
- the two different methods of applying tabu and
- the two different approaches of applying strategic moves.

The above cases are evaluated for percentage improvement over the initial logical topology. From the above results it can be seen that on average we can expect an improvement of *12% (approx.)* for a 10 node network in using the tabu search approach described in the case 3 in this thesis to solve the logical topology design problem. Considering the above set of results where the number of moves has been fixed to 50, the most appropriate tabu list size for a 10 node network is observed to be 7. Similar conclusions can be drawn based on the data for different number of nodes in the network.

CHAPTER 5

Conclusions and Future Work

5.1 Conclusion

In this thesis, a practical and efficient algorithm for logical topology design in WDM networks using the meta-heuristic technique of tabu search has been presented. The algorithm has been implemented for different scenarios and shows encouraging results. Three variations of tabu tenure, two different ways to apply tabu and two different strategic moves have been experimented with. The experimental results indicate that tabu search is in fact a promising approach to solve the problem of logical topology design in WDM networks.

The main contribution of this thesis is to develop a tabu search approach to solve the problem of logical topology design efficiently. The subpart of the problem of routing over the logical topology has been solved using the Yumei's approximation algorithm. It is observed from the results that Case 3 shows the maximum improvement in congestion amongst all the four cases. It is seen that tabu tenure of 5 is most appropriate for 5 node network and for 6, 10 and 14 node network more promising results are obtained for tabu tenure of 7.

The results show that the approach proposed in this thesis can be used to solve the problem of logical topology design for 14 node networks restricted to 50 moves, in approx. 10,000 seconds.

5.2 Future Work

In this thesis, the well known problem of logical topology design has been investigated. The problem of logical topology design is considered to be intractable using the mathematical approach. The meta-heuristic approach opens new avenues to solve the problem. It would be very useful to further investigate the application of tabu search meta-heuristic to the problem of logical topology design. It would be interesting to develop an alternative tabu search approach by experimenting in more detail with different strategic moves. Also an entirely different architecture for tabu search procedure is possible. The solutions can be compared to present approach for the percentage improvement over the initial congestion.

SELECTED BIBLIOGRAPHY

[Ac 87] A.S. Acampora, " A multi-channel multihop local lightwave network", Proceeedings IEEE Globecom '87, November 1987.

[AL 94] Baruch Awerbuch, Tom Leighton, "Improved approximation algorithms for the multi-commodity flow problem and local competitive routing in dynamic networks", Proceedings of the twenty-sixth annual ACM symposium on Theory of computing, May 1994

[AMO 93] Revindra K. Ahuja, Thomas L. Magnanti, and James B. OrLin, "Network flows," Prentice-Hall, Inc. 1993

[BMS 94] S.Banerjee, B.Mukherjee, D.Sakar, Heuristic Algorithms for Constructing Optimized Structures of Linear Multihop Lightwave Networks", -IEEE Transactions on Communication, Vol.42, n.2, pag.222-231, Feb. Mar. Apr. 1994.

[CGK 92] I. Chlamtac, A. Ganz, and G. Karmi, "Lightpath communications: An approach to high bandwidth optical WAN's,"IEEE Transactions on Communications, Vol. 40, No. 7, pp. 1171--1182, July 1992

[CFG 98] Costamagna, E., A. Fanni and G. Giacinto (1998). "A Tabu Search Algorithm for the Optimisation of Telecommunication Networks." European Journal of Operational Research, 106: pp357-372.

[CFZ 96] I. Chlamtac, A. Farago, and T. Zhang, "Lightpath (wavelength) routing in large WDM networks," IEEE Journal on Select. Areas of Communications, vol. 14, pp. 909–913, June 1996.

[DR 00] R. Dutta and G. N. Rouskas, "A survey of virtual topology design algorithms for wavelength routed optical networks," Optical Networking Magazine, Vol. 1, no. 1, pp. 73–89, Jan. 2000.

[Gl 90] Fred Glover, "Tabu Search: A Tutorial", pages 74-94 in Interfaces 20: 4 july-August 1990

[Gl 96] Fred Glover, "Tabu Search and Adaptive Memory Programming - Advances, Applications and Challenges", in: Interfaces in Computer Science and Operations Research, Barr, Helgason and Kennington eds., Kluwer Academic Publishers, 1996.

[Gr 91] Paul E. Green, "The Future of Fiber-Optic Computer Networks," Computer, Volume: 24 Issue: 9 , Sept. 1991 Page(s): 78 –87

[GL 97] Fred Glover, Manuel Laguna, "Tabu Search" Kluwer Academic Publishers

[GLM 01] A.Grosso, E.Leonardi, M.Mellia, A.Nucci, "Logical Topologies Design over WDM Wavelength Routed Networks Robust to Traffic Uncertainties", IEEE Communications Letters, vol 5(4), Apr 2001, pp 172–174.

[Lu 04] Yumei Lu, Approximation algorithms for optimal routing in Wavelength Routed WDM Networks. Masters thesis, Department of Computer Science, University of Windsor, Ontario, Canada.

[LG 98] A. Lokketangen and F. Glover, "Solving zero-one mixed integer programming problems using tabu search", European Journal of Operations Research, pp. 624-658,1998

[Mu 97] B. Mukherjee, "Optical Communication Networks" McGraw Hill 1997

[MN 01] Eytan Modiano, Aradhana Narula-Tam Survivable Routing of Logical Topologies in WDM Networks, pages 348-357 INFOCOM 2001.

[RS 96] Rajiv Ramaswami, Kumar N. Sivarajan. Design of Logical Topologies for Wavelength-Routed Optical Networks. IEEE Journal on Selected Areas in Communications, Vol.14, No.5, June 1996

[RS 02] Rajiv Ramaswami, Kumar N. Sivarajan, "Optical Networks: A practical perspective Second Edition" Morgan Kaufmann publishers. 2002

[Su 06] Subir Bandyopadhyay , manuscript in print

[Ta 89] Taha H A, Operations Research ± An Introduction, 4th ed. (New York: Macmilan Publishing Company) (1989)

[ZV 03]Andrew Zalesky, Hai Le Vu, A Framework for Solving Logical Topology Design Problems Within Constrained Computation Time IEEE communications letters, pp 499-501 vol. 7, no. 10, October 2003.

Vita Auctoris

Ashutosh Sood was born in 1974 in Rajpura, Punjab, India. He graduated from D.A.V. College, Chandigarh, India in 1991. From there he went on to the Thapar Instititute of Engineeering and Technology, Patiala, India where he obtained a Bachelor of Engineering in Civil in 1996. He is currently a candidate for the Master's degree in Chemistry at the University of Windsor and hopes to graduate in Summer 2006.