

Topology Design of Enterprise Networks

by

Salman Ahmad Khan

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER ENGINEERING

December, 1999

INFORMATION TO USERS

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

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

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

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

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

**Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600**

UMI[®]



Topology Design of Enterprise Networks

BY

Salman Ahmad Khan

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE
In
COMPUTER ENGINEERING

December 1999

UMI Number: 1398760

UMI[®]

UMI Microform 1398760

Copyright 2000 by Bell & Howell Information and Learning Company.

All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

Bell & Howell Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS
DHAHRAN 31261, SAUDI ARABIA
COLLEGE OF GRADUATE STUDIES

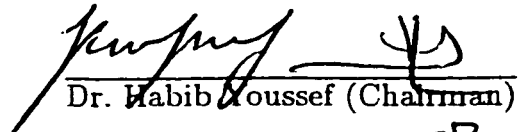
This thesis, written by

SALMAN AHMAD KHAN

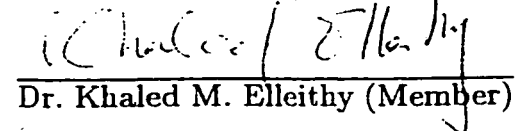
under the direction of his Thesis Advisor and approved by his Thesis Committee,
has been presented to and accepted by the Dean of the College of Graduate Studies,
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER ENGINEERING

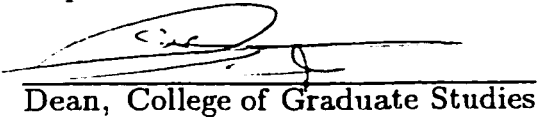
Thesis Committee


Dr. Habib Moussef (Chairman)


Dr. Sadiq M. Salt (Co-Chairman)


Dr. Khaled M. Elleithy (Member)


Department Chairman


Dean, College of Graduate Studies

12-2-2007
Date



Dedicated

to

my beloved parents

and

to the memory of my maternal grandfather.

Acknowledgements

All praise be to Allah, Subhanahu-wa-ta-A'la, for his limitless blessing and guidance. May Allah bestow peace on his prophet, Muhammad (peace and blessings of Allah be upon him), and his family. I acknowledge the support and facilities provided by King Fahd University of Petroleum and Minerals, Dhahran, Kingdom of Saudi Arabia.

All my family members, especially my parents, were constant source of motivation and support. Their love and care carried me through some difficult moments in my life. Their prayers, guidance and inspiration lead to this accomplishment. I am also very thankful to both of my sisters for their kind support.

I would like to express my profound gratitude and appreciation to my thesis committee chairman Dr. Habib Youssef and co-chairman Dr. Sadiq M. Sait, for their guidance, patience, and sincere advice throughout this thesis. I acknowledge both of them for their valuable time, constructive criticism, and stimulating discussions. Thanks are also due to my thesis committee member, Dr. Khaled M. Elleithy for his comments and critical review of the thesis.

I am very thankful to the department chairman Dr. Abdullah I. Al-Mojel and the dean of College of Computer Science and Engineering Dr. Khalid M. Al-Tawil, for their cooperation and support and giving me a chance to pursue my MS degree in the department.

I am also thankful to my fellow graduate students and all my friends on the campus especially Hussain, Ahsan, Naved, Zafar, and Rashid who provided a wonderful company. Thanks are also due to Khalid Al-Utaibi for helping me in the Arabic abstract of this thesis.

Contents

Acknowledgements	ii
List of Tables	ix
List of Figures	x
Abstract (English)	xiv
Abstract (Arabic)	xv
1 Introduction	1
2 Literature Review	7
2.1 Constructive Heuristics	8
2.2 Iterative Heuristics	14
2.3 Conclusion	20
3 Preliminaries	21

3.1	Introduction	21
3.2	Assumptions and Terminology	22
3.3	Formal Description and Notation	23
3.4	Objective Function	27
3.5	Computation of Objective Values	27
3.5.1	Monetary cost	27
3.5.2	Average network delay	28
3.5.3	Maximum number of hops	29
3.5.4	Constraints	30
3.6	Simulated Evolution	31
3.7	Fuzzy Logic	36
3.7.1	Fuzzy Set Theory (FST)	36
3.7.2	Fuzzy Reasoning	39
3.7.3	Linguistic Variable	39
3.7.4	Fuzzy Rules	40
3.7.5	Fuzzy Logic System (FLS)	42
3.8	Conclusion	43
4	Fuzzy Logic Based Simulated Evolution Algorithm for Network	
	Topology Design	44
4.1	Introduction	44

4.2	Proposed Scheme and Implementation Details	47
4.2.1	Initialization	47
4.2.2	Proposed Fuzzy Evaluation Scheme	48
4.2.3	Selection	51
4.2.4	Proposed Fuzzy Allocation Scheme	53
4.3	Stopping Criterion	60
4.4	Conclusion	61
5	Experiments and Results	62
5.1	Introduction	62
5.2	Backbone Design Using Fuzzy Simulated Evolution Algorithm	63
5.2.1	Comparison of SE_FF and SE_VB	67
5.2.2	Effect of Tabu Search Based Allocation and Tabu List Size	80
5.2.3	Comparison of SE_FF and SE_TS	94
5.2.4	Comparison of SE_TS and Esau-Williams algorithm	102
5.2.5	Comparison of Simulated Annealing and SE_TS	104
5.2.6	Quality of Final Solution with Different Initial Solutions	107
5.3	Assignment of LAN Segments to Network Devices using Augmenting Path Algorithm	110
5.4	Design of the Internal Structure of a Local Site using Fuzzy Simulated Evolution Algorithm	111

5.5	Conclusion	117
6	Conclusion	118
6.1	Summary	118
6.2	Future Research	122
	BIBLIOGRAPHY	123

List of Tables

5.1	Classification of our SE implementations.	65
5.2	Parameter values for different stages of the SE algorithm.	65
5.3	Characteristics of test cases used in our experiments. LCostMin, LCostMax, and TCostMin are in US\$. TDelayMin is in milliseconds. Traffic is in Mbps.	66
5.4	Equipment parameters and their characteristics assumed in our experiments.	66
5.5	Best solution for different bias values. Monetary cost is in US \$, delay is in milli seconds per packet, and execution time is in minutes. . . .	69
5.6	Comparison of SE_FF and SE_VB. B = best bias, C = Cost in US \$, D = Delay in milli seconds per packet, H = hops, and T = execution time in minutes. The percentage gain shows the improvement achieved by SE_VB compared to SE_FF	73
5.7	Best solution for different tabu list sizes. Monetary cost is in US \$, delay is in milli seconds per packet, and execution time is in minutes.	84

5.8	Results for best tabu list size. Execution time is in minutes.	94
5.9	Comparison of SE_FF and SE_TS. B = best bias, C = Cost in US \$, D = Delay in milli seconds per packet, H = hops, M = Membership in "Good Topology", T = execution time in minutes, TL= Tabu list size.	96
5.10	Percentage gain in improvement achieved by SE_TS compared to SE_FF. C = Cost in US \$, D = Delay in milli seconds per packet, H = hops.	96
5.11	Comparison of EW and SE_TS. C = Cost in US \$, D = Delay in milli seconds per packet, H = hops, T = execution time in minutes, TL= Tabu list size. The percnetage gain shows the improvement achieved by SE_TS compared to EW.	103
5.12	Time analysis for different phases of SE_TS algorithm.	104
5.13	Comparison of SA and SE_TS. C = Cost in US \$, D = Delay in milli seconds per packet, H = hops, T = execution time in minutes, TL= Tabu list size. The percentage gain shows the improvement achieved by SE_TS compared to SA.	105
5.14	Quality of final solution with different <u>initial</u> solutions for SE_TS. TL= Tabu list size, C = Cost in US \$, D = Delay in milli seconds per packet, H = hops.	110

5.15 Segment-concentrator and concentrator-concentrator distances in meters. The concentrator to which the segment has been assigned is also shown. 114

List of Figures

1.1	A typical Enterprise Network	3
3.1	Structure of the simulated evolution algorithm.	33
3.2	Membership function for a fuzzy set A.	37
3.3	Fuzzy logic system.	42
4.1	Depths of links with respect to the root.	51
4.2	Membership function for monetary cost of a link.	52
4.3	Membership function for depth of a link.	52
4.4	Basic components for a good topology.	54
4.5	Membership function for monetary cost.	55
4.6	Membership function for average network delay.	57
4.7	Membership function for maximum number of hops between a source- destination pair.	58
4.8	Two disjoint graphs containing nodes P and Q.	59
4.9	Controlled moves. The dotted lines show five controlled moves.	59

4.10	Random moves. The dotted lines show five random moves.	60
5.1	Cardinality of selection set for different bias values for test case n50.	70
5.2	Comparison of SE_FF and SE_VB for n33 (a) current monetary cost (US \$) (b) best monetary cost.	74
5.3	Comparison of SE_FF and SE_VB for n33 (a) current average delay (seconds) (b) best average delay.	75
5.4	Comparison of SE_FF and SE_VB for n33 (a) current maximum hops (b) best maximum hops.	76
5.5	Cardinality of selection set against frequency of solutions for SE_FF and SE_VB for n33.	77
5.6	Comparison of SE_FF and SE_VB for n33 (a) Membership in "Good Topology";(b) Cardinality of selection set versus iterations.	78
5.7	Links and their utilizations for best solution generated for n33 by (a) SE_FF (b) SE_VB. Traffic is in Mbps.	81
5.8	Variation of monetary cost with iterations for n50, Tabu list size=7 (a) current monetary cost (US \$); (b) best monetary cost.	86
5.9	Variation of average delay with iterations for n50, Tabu list size=7 (a) current average delay (seconds); (b) best average delay.	87
5.10	Variation of maximum hops with iterations for n50, Tabu list size=7 (a) current maximum hops; (b) best maximum hops.	88

5.11 Membership in function “good topology” for n50, Tabu list size=7	
(a) current membership; (b) best membership.	89
5.12 Frequency of solutions falling in different ranges of function “good topology” for n50, Tabu list size=7.	90
5.13 Variation of average goodness of links with iterations for n50, Tabu list size=7.	91
5.14 Variation of minimum and maximum goodness of a link with iterations for n50, Tabu list size=7; (a) minimum goodness of a link (b) maximum goodness of a link.	92
5.15 Links and their utilizations for best solution generated by SE_TS for test case n50. Traffic is in Mbps.	93
5.16 Links and their utilizations for best solution generated for n40 by (a) SE_FF (b) SE_TS. Traffic is in Mbps.	97
5.17 Comparison of SE_FF and SE_TS for n40 (a) current monetary cost (US \$)(b) best monetary cost.	99
5.18 Comparison of SE_FF and SE_TS for n40 (a) current average delay (seconds) (b) best average delay.	100
5.19 Comparison of SE_FF and SE_TS for n40 (a) current maximum hops (b) best maximum hops.	101
5.20 Frequency of solutions for different membership ranges for SA and SE_TS for n25.	108

5.21 Cumulative frequency of solutions for different membership ranges for n25 (a) SA (b) SE_TS.	109
5.22 Segments and concentrators before assignment.	112
5.23 Segments and concentrators after assignment.	113
5.24 Topology generated for concentrators.	116

THESIS ABSTRACT

Name: SALMAN AHMAD KHAN
Title: TOPOLOGY DESIGN OF ENTERPRISE NETWORKS
Major Field: COMPUTER ENGINEERING
Date of Degree: DECEMBER 1999

Topological Design of Enterprise networks (ENs) is a hard problem. Therefore, we resort to heuristics to come up with desirable solutions. Enterprise Network topology design problem comprises several conflicting objectives such as minimization of cost, minimization of network delay, minimization of maximum number of hops etc. Furthermore, some of the objectives are imprecise. Fuzzy Logic provides a suitable mathematical framework in such a situation. In this thesis, an approach to EN topology design problem has been presented. It comprises of three main steps, namely, assignment of segments to network devices within a local site, design of the internal topology of a local site, and backbone topology design. Augmenting path algorithm is used for assignment of segments to network devices within a local site. For the other two steps, an approach based on Simulated Evolution algorithm is suggested. Fuzzy cost functions at the two main phases of the algorithm, namely, evaluation and allocation, have been incorporated. Different variations of the algorithm using fixed and variable bias schemes in the selection phase have been compared. Tabu Search-based characteristics have also been incorporated in the allocation phase of a variation that uses variable bias. This approach is then compared with the Esau-Williams algorithm, which is one of the most widely used greedy algorithms for centralized network design and also with Simulated Annealing algorithm which is another well-known iterative heuristic. Results suggest that Tabu Search allocation-based Simulated Evolution algorithm performs better than the Esau-Williams algorithm as well as Simulated Annealing algorithm.

MASTER OF SCIENCE DEGREE

King Fahd University of Petroleum and Minerals, Dhahran.
December 1999

خلاصة الرسالة

اسم الطالب: سلمان أحمد خان
عنوان الرسالة: تصميم طوبولوجي لشبكات المؤسسات
الدرجة: ماجستير علوم

التصميم الطوبولوجي لشبكات المؤسسات (Enterprise Networks) يعتبر من المشكلات التي لا يمكن حلها في زمن ممثل بكثيرة حدود (Hard Problem)، لذلك نلجأ للخوارزميات التقريبية لإيجاد حلول مقبولة لهذه المشاكل.

إن مشكلة التصميم الطوبولوجي لشبكات المؤسسات تتضمن عددا من الأهداف المتعارضة كتقليل التكلفة وزمن النقل في الشبكة، وتقليل الحد الأعلى للقفزات (Hops). بالإضافة إلى ذلك فإن هذه الأهداف غير محدد بشكل دقيق، وفي هذه الحالات يوفر المنطق المبهم (Fuzzy Logic) إطارا رياضيا مناسباً لحل المشكلة.

في هذه الرسالة نقدم طريقة لحل مشكلة التصميم الطوبولوجي لشبكات المؤسسات تتضمن ثلاث خطوات هي: توزيع القطاعات على أجهزة الشبكة ضمن نطاق محلي، وتصميم الطوبولوجية الداخلية للنطاق المحلي، والتصميم الطوبولوجي للسلسلة الفقرية (Backbone).

في الخطوة الأولى يتم توزيع القطاعات باستخدام خوارزم المسار المتريد (Augmenting Path Algorithm)، أما في الخطوات الأخرى فنستخدم نهجا يقوم على خوارزم النشوء المحاكي (Simulated Evolution Algorithm). وتتضمن المرحلتان الرئيستان من الخوارزم المقترح – وهما التقييم والتخصيص – دوال تكلفة مبهمه (Fuzzy Cost Functions).

في هذه الرسالة تم مقارنة صور متعددة من الخوارزم باستخدام التحفيز الثابت والتحفيز المتغير في مرحلة الاختيار. كما قمنا بتضمين مرحلة للتخصيص – في إحدى صور الخوارزم التي تستخدم التحفيز المتغير – دوال مبنية على البحث بطريقة Tabu (Tabu Search)، ومن ثم قمنا بمقارنة هذا الخوارزم بخوارزم آخر يعتبر من أكثر الخوارزميات استخداما في تصميم الشبكات المركزية وهو Esau-Williams Algorithm، وقد دلت نتائج هذه المقارنة على أن أداء الخوارزم المقترح (Tabu Search allocation-based Simulated Evolution Algorithm) أفضل من أداء الخوارزم Esau-Williams Algorithm.

درجة الماجستير في العلوم
جامعة الملك فهد للبترول والمعادن
الظهران-المملكة العربية السعودية
ديسمبر ١٩٩٩ م

Chapter 1

Introduction

Enterprise Network (EN) topology design is an important engineering problem. A typical enterprise network consists of a large number of components, such as main frame computers, mini systems, workstations, PCs, user terminals, printers, etc., [1]. Various devices such as hubs, workgroup switches, backbone switches, routers, etc., are used to interconnect and network these computers and peripherals. The EN topology is governed by several constraints. Geographical constraints dictate the break-down of such internetworks into smaller parts or groups of nodes, where each group makes up what is called a LAN. Thus, we can define an EN as a collection of interconnected LANs. Further, the nodes of a LAN may be subdivided into smaller parts, called *LAN segments*, to satisfy other constraints and objectives, for example, capacity constraints, cabling limitation, minimizing delay, cabling and equipment cost etc., [1]. The topology design of LAN itself consists of two main

issues: *segmentation*, where LAN segments are found and *design of actual topology*, which consists of interconnecting the individual segments. Topology design at LAN level requires interconnection of LAN segments via bridges and layer 2 switches [2, 3]. Users (stations) are allocated to these segments based on some criteria. Only spanning tree topologies can be used as active LANs configurations [2, 3, 4, 5, 6]. However, when the bridges support the spanning tree protocol, the actual physical topology does not have to be loop-free.

In designing the topology of these enterprise networks, an important step is to find the best possible locations of components to optimize the performance criteria, such as cost and delay. A good and structured enterprise network has three layers (see Figure 1.1):

1. Local Access Layer, which provides workgroup access to the network.
2. Distribution Layer, which provides policy based connectivity among the workgroups. This layer is implemented with layer 3 switches, routers, and gateways. This is where packets manipulation takes place.
3. Backbone Layer, which provides high speed optimal transport of data among local sites.

Thus, the design of such a structured enterprise network can be approached in four steps:

1. Assignment of users/stations to LAN segments.

2. Assignment of LAN segments to local sites that will make up a single LAN.
3. Design of the internal structure of each local site (i.e., in what topology the LAN segments of a local site are connected).
4. Backbone design, where the local sites are connected to the backbone.

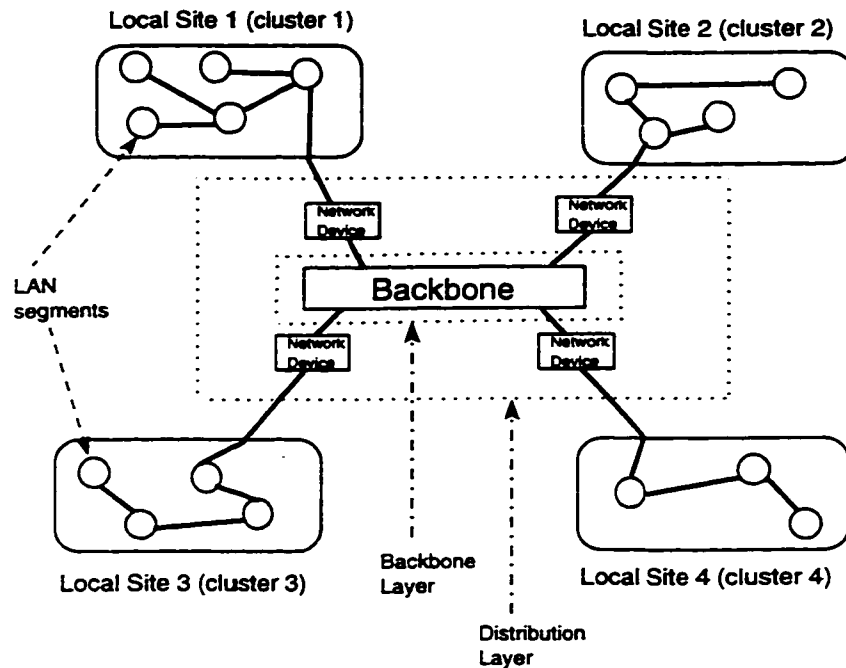


Figure 1.1: A typical Enterprise Network .

Topological design of enterprise networks is a hard problem [1]. Even the design of a LAN is itself an NP-hard problem [2, 3, 4]. Therefore, we have to use intelligent methods known as 'heuristics' to get near optimal solutions in reasonable amount of time. There are two categories of heuristics: *constructive* and *iterative*.

Constructive heuristics produce a complete solution by making deterministic

moves. Constructive techniques are fast but require careful implementation and tuning. Because of their local view of the search space and their greedy nature, the solution they return is usually far from optimum. Further, for highly constrained problems, they may fail to find a feasible solution [7].

Iterative heuristics attempt to improve a complete solution by making controlled stochastic moves [7]. Generally we can classify iterative schemes into two subclasses: schemes which always accept good solutions like local search, and schemes which occasionally accept bad solutions like simulated annealing [8], genetic algorithms [9], tabu search [7], and simulated evolution [7]. This class of schemes generally outperform constructive and greedy schemes because their “hill climbing” property saves them from getting trapped in local optima [7, 10].

Iterative heuristics have been used for topological network design. The use of Genetic Algorithm (GA) for topological network design has been proposed in [2, 3, 11, 12, 13]. Similarly, use of Simulated Annealing (SA) has been reported in [4]. However, there were some concerns over the execution times of these schemes [14][15]. Kling and Banerjee proposed *Simulated Evolution* (SE) heuristic [16] for VLSI Cell Placement. The heuristic combines iterative improvement and constructive perturbation. It saves itself from getting trapped in local optima by using stochastic selection of design components for perturbations.

The EN topology design (TDEN) problem is a constrained multiobjective optimization problem, where some of the objectives can only be estimated and where

several decisions are usually based on expert knowledge. For such class of problems, fuzzy logic provides an easy way of expressing expert human knowledge in decision making process of any iterative heuristic. The advantage of fuzzy logic over classical crisp logic is that it establishes an approximate truth value of propositions in accordance with the rules designed by the expert, while in crisp logic the proposition will either be true or false. Furthermore, it provides a rigorous algebra for dealing with imprecise information.

Research Objectives

In this thesis we present an approach to TDEN problem based on a Fuzzy Simulated Evolution Algorithm for multiobjective optimization. In our scheme we minimize three cost parameters of the topology design: monetary cost, average network delay, and maximum number of hops between a source-destination node pair. We also use some other algorithms in this process. We have investigated the fuzzification of different stages of SE algorithm. The SE algorithm consists of three distinct steps: **evaluation**, **selection**, and **allocation**. We propose fuzzification of **evaluation** and **allocation** stages of the simulated evolution algorithm. Our proposed fuzzy evaluation scheme combines monetary cost and delay for evaluating the placement of a link between two nodes. The proposed fuzzy allocation scheme combines the monetary cost, average network delay, and maximum number of hops between a source-destination pair.

Organization of Thesis

The rest of this thesis is organized as follows. In Chapter 2 we review related literature. This chapter covers different approaches that have been taken in network topology design problems and the objectives they optimize.

In Chapter 3 we cover different aspects of TDEN problem. This includes the formal description of the problem, notation, assumptions, terminology, cost function, and objective values computation. The chapter also gives a background on simulated evolution algorithm and fuzzy logic.

In Chapter 4 we discuss our proposed fuzzy logic based Simulated Evolution algorithm for TDEN. In this chapter we describe different steps of our fuzzy SE algorithm and the approach we have taken to implement each one of them.

In Chapter 5 we present and discuss our experiments and results. The thesis ends with conclusion and future work in Chapter 6.

Chapter 2

Literature Review

Network topology design is an NP-complete problem [17]. Heuristics have been used to solve problems that are related to network topology design. Heuristic algorithms can be classified as *constructive* and *iterative*. Constructive heuristics produce a complete solution by making deterministic moves while iterative heuristics attempt to improve a complete solution by making controlled stochastic moves [10]. Simulated Annealing [8], Genetic Algorithm [9], Tabu Search [15, 7], and Simulated Evolution [18, 19, 20, 21] are examples of iterative heuristics. In this chapter, literature is reviewed related to constructive and iterative heuristics as applied to network topology design.

2.1 Constructive Heuristics

Constructive schemes build a topology in a piecewise manner. These schemes join two nodes together at a time, until the topology is complete. Such schemes are fast but fall short of good topology. This is due to the fact that these schemes make decision about joining two nodes on the basis of partial topology only. Secondly, once a decision is made, no matter how bad it is, there is no mechanism to reverse it. Therefore, these schemes may remain trapped in local minima. Following are some related constructive heuristics for topology design.

Kruskal Algorithm

Kruskal suggested a greedy algorithm for finding minimal spanning trees (MSTs). Since the topology of a structured enterprise network is usually a spanning tree, Kruskal's Algorithm can be used. The algorithm starts with each node being a single node fragment. A *fragment* is a subtree (i.e., a subgraph that is a tree) of an MST. The algorithm then successively combines two of the fragments by using the arc that has minimum weight over all arcs that when added to the current set of fragments do not form a cycle and do not violate other stated constraints. The algorithm proceeds by building up simultaneously several fragments that eventually join into an MST; however, only one arc at a time is added to the current set of fragments. The algorithm terminates in $(N - 1)$ iterations [22]. The complexity of

the algorithm is $O(m \log m)$, where m is the number of arcs.

Prim Algorithm

Prim [23] suggested an algorithm to find minimum spanning trees, which could be applied to design spanning tree network topologies. The algorithm starts with an arbitrarily selected single node as a fragment and enlarges the fragment by successively adding a minimum weight outgoing arc. More specifically, the algorithm starts with one node in the tree and all other nodes not in the tree (out-of-tree). Then, while there are still out-of-tree nodes, it finds the out-of-tree node which is nearest to the tree and does not violate constraints, brings that node into the tree, and records the edge which connects it to the tree. The algorithm also terminates in $(N - 1)$ iterations and has a worst case efficiency of $O(n^2)$ [22, 24].

Esau-Williams Algorithm

Esau and Williams [25] suggested an algorithm for approximating the optimal network topology based on spanning trees. They based their algorithm on the fact that links have a significant effect on the cost of most telecommunication systems, and it is important to find a configuration that is reasonably optimal. In their work they assumed that cost is proportional to distance. The goal is to connect all the nodes to the central node, using multipoint links, and giving a minimum possible, or rather an optimal cost. The constraint they assumed is that there is no traffic overflow

on any link. The problem therefore falls in the category of designing *Constrained MSTs*.

The algorithm is suitable for centralized topology design, where the traffic from all the N nodes must go through the central node (denoted by 0). Therefore, it can be assumed that there is input traffic only from the non-central nodes to the central node and the reverse. At each successive iteration, a link $(i,0)$ connecting some node i with the central node 0 is deleted from the current spanning tree, and a link (i,j) is added. These links are chosen so that:

1. No cycle is formed,
2. the capacity constraints of all the links of the new spanning tree are satisfied,
and
3. the saving $w_{i0} - w_{ij}$ in link weight obtained by exchanging $(i,0)$ with (i,j) is positive and is maximized over all nodes i and j for which (1) and (2) are satisfied.

The algorithm terminates when there are no nodes i and j for which requirements (1) to (3) are satisfied when $(i,0)$ is exchanged with (i,j) [22][25].

The Augmenting Path Algorithm

The algorithm has been used to solve the “terminal assignment problem” where the terminals (users) in a network are assigned to concentrators (e.g., hubs, or bridges)

based on a cost function. The algorithm can be used to find an optimal solution [24].

The algorithm is based on the following observations:

1. Ideally, every terminal would be assigned to the nearest concentrator. The only reason for not doing so is that the capacity constraint (of the concentrator) prevents this [24] .
2. If a terminal is already assigned to a nearby concentrator, the only reason for moving it to one that is farther away is to make room for another terminal that would have to detour even farther. Thus, it is only necessary to move terminals on concentrators that are full, and then only to make room for another terminal [24].
3. Given an optimal partial solution with k terminals assigned (i.e., these k terminals are assigned to concentrators at a minimum possible total cost), an optimal partial solution with $k + 1$ terminals can be found by finding the least expensive way of adding the $k + 1$ st terminal to the k terminal solution. Note that this may involve reassigning some of the first k terminals [24].

Stated this way, it is clear that with sufficient effort it is possible to find an optimal solution, since an entirely new solution can be found by reassigning as many terminals as desired. Clearly, any solution, including the optimal one, is obtainable this way.

Clustering Algorithms

The identification of the nodes of each local site can be seen as a clustering problem.

A methodology based on this approach has been suggested by Youssef et al., [1] to design enterprise network backbones. The methodology consists of two phases:

1. A clustering phase (Phase-1). The objective here is to decide the number of devices required to network all the nodes of the computing resources of the enterprise. The nodes are partitioned into clusters, and a network device of the appropriate type and capacity is assigned to each cluster. Constraints considered in this phase include, maximum forwarding capacity of each networking device representing a cluster, the type of each node, and the maximum geographic proximity of a certain node to be included in a cluster [1].
2. A topology design phase (Phase-2). The outcome of this phase is a minimum spanning tree that interconnects the networking devices of all the clusters. Also, a routing matrix is automatically generated to route packets to *minimum hop count* metric. In this phase the designer has also the option of specifying an upper bound on the utilization of each backbone link [1].

NetMod

A modeling tool called NetMod for the design of campus networks is presented in [26]. The tool provides analysis of different parameters (such as delay, or traffic

per link), of Token ring and Ethernet networks. Performance measures such as mean delay, utilization, and throughput are generated for considered networks. Also, the tool has two user interfaces: a HyperCAD graphical user interface and a spreadsheet mathematical user interface based on Microsoft Excel. NetMod decomposes the entire network into a hierarchy of interrelated submodels. A submodel can be any of the following:

1. Network Devices such as bridges, routers, hubs, etc.
2. Polling Submodel: This submodel is primarily used for token ring networks. It can also be used as an approximation for FDDI networks under normal traffic loads. It uses cyclic server queuing system with non-exhaustive service. Expressions developed by other research works are used for the calculation of mean waiting time.
3. Ethernet Submodel: Well known CSMA/CD Ethernet protocol formulae based on the work of Hammond et al. [27] are utilized in this submodel for the utilization and throughput calculation. Delay calculation is based on the work done by Lam [26].

Add-Drop Algorithms

These algorithms are used in mesh topological design of networks. Mesh topologies can be produced by starting with a topology, usually a feasible topology, and then

locally modifying it by adding or dropping links. The add or drop procedure is usually in some sense greedy, maximizing or minimizing some figure of merit [24]. It is possible, for example, to start with a minimal spanning tree with adequate capacity to carry the entire load at a reasonable level of delay. Then, links are added to the network in a greedy way (i.e., the links which give the minimum increase in cost) while still satisfying the requirements on throughput and delay. Another possibility is to start with a complete graph and then identify a link to drop. Again, we can choose a link which, for example, gives the highest reduction in the cost [24].

2.2 Iterative Heuristics

In contrast to constructive techniques, iterative schemes require larger computational time to generate good network topologies. Iterative techniques start with an initial solution and repeatedly modify the solution in each iteration until no more improvement occurs. The modification in solution is intended to reduce the cost of the solution. Iterative schemes can be further classified on the basis of whether they can accept bad solutions probabilistically or not. Those iterative algorithms which allow acceptance of bad moves probabilistically fall in the sub-category of “stochastic iterative algorithms”. This property is called “hill climbing property”. It saves algorithms from getting trapped in local minima. However, it is required

that acceptance of bad moves be controlled to avoid random traversal of search space. Examples of such probabilistic iterative schemes are simulated annealing (SA), genetic algorithm (GA), simulated evolution (SE), and tabu search (TS). Of these schemes, SA and GA have been successfully applied to topology design of networks. Following is a brief review of these implementations.

Simulated Annealing

Simulated Annealing is a popular combinatorial optimization algorithm proposed by Kirkpatrick et. al. [8]. It is derived from the analogy of the physical annealing process of metals. SA works on a single solution. The neighborhood state of the solution is generated by randomly selecting modules and interchanging their positions. All good moves are accepted. However, bad moves are stochastically accepted. The acceptance probability of bad moves is controlled by a cooling schedule. In early stage of the search, number of bad moves are accepted with high probability. However, as search progresses, temperature decreases and so does the probability of accepting bad moves. In the last part of the search, SA behaves as a greedy algorithm, accepting only good moves. For details of simulated annealing algorithm, interested readers are referred to [7].

Ersoy et al. [4] have used an approach based on Simulated Annealing (SA) for topological design of interconnected LAN/MAN networks. The main objective is to minimize the average network delay. They have considered transparent bridges,

which are required to form a spanning tree topology. To measure the quality of solution, they found a lower bound for the average network delay. First, minimal delay spanning tree networks are identified using SA algorithm. Then, overall backbone interconnecting these networks with a minimum delay is constructed. The topology is modeled by a network of queues. The objective is to optimize the delay of the spanning tree connecting N different LANs with $N-1$ bridges such that traffic flow constraints are satisfied. It is noticed here that certain constraints are not considered, such as cost, and bridge capacities. SA is used again to identify the minimal delay of the MAN interconnecting the LANs constructed in the first stage. The objective function is similar to stage one with one additional constraint, that only one bridge per cluster is allowed. Finally, clusters with a maximum access time exceeding a certain threshold are revisited. Each cluster with such situation is allowed to have one more bridge to lower its delay below the access time threshold. If the cluster maximum access time still exceeds the threshold, another bridge is added and so on. Simulation showed that the algorithm found optimal solution for smaller problems. For larger size problems, comparison with lower bound indicated that the solution found was not very far away from the global optimum.

Fetterolf [6] used simulated annealing to design LAN-WAN computer networks with transparent bridges. He developed mathematical models of LAN-WAN networks and formulated an optimization problem. A simulated annealing algorithm was proposed which generates sequences of neighboring spanning trees and evalu-

ates design constraints based on maximum flow, bridge capacity, and end-to-end delay. As the annealing temperature is lowered the algorithm moves towards the global optimal solution. Experimental results showed that LAN-WAN designs using simulated annealing were better than 99.99% of all feasible designs.

Genetic Algorithm

Another powerful and popular optimization algorithm is Genetic Algorithm (GA) which works by emulating the natural process of evolution as a means of progressing toward optimum. It was first suggested by Holland [28]. Unlike simulated annealing, which works repeatedly on single solution, GA works on a set of solutions in parallel. The set of solutions are known as a *population*. Each solution is represented by a string of symbols known as *chromosome*. Four genetic operators, namely, *selection*, *crossover*, *mutation*, and *inversion* are repeatedly applied on a collection of solutions to generate new offsprings. Extensive literature is available on genetic algorithms and their use in combinatorial optimization problems. Interested reader is referred to [7, 9].

Pierre et al. [29] used genetic algorithm to solve the topological design problem of distributed packet switched networks. The goal was to find a topology that minimizes the communication costs by taking into account constraints such as delay and reliability. They conducted experiments to measure the quality and sensitivity of solutions provided by their algorithm. The results showed that the set of solutions

obtained at each generation followed a normal distribution, based on Wilk's test for normality [29]. Furthermore, other experiments undertaken to study the influence of probability variations associated with genetic operators showed, that in terms of network cost, a crossover of 60% generated better solutions in general. They also observed that the higher the crossover probability, the greater is the dispersion of results. Also, a high inversion probability seemed less beneficial than a lower one. This lower probability was in the order of 10%. A comparison of this Genetic Algorithm was done with Simulated Annealing to evaluate the effectiveness of GA. In almost all cases, GA led to better solutions than SA, both in terms of network cost and average delay. GA was also compared to cut-saturation algorithm [30] on 2-connected networks with sizes ranging from 10 to 25 nodes. The costs found by GA turned out to be better than those obtained with cut-saturation on medium-size networks. However, GA takes much more time than cut-saturation algorithm to converge to a good solution.

Dengiz et al. [12][13] used genetic algorithm to optimize a network topology using the cost and reliability as optimization measures. They called their algorithm GA with knowledge-based steps (GAKBS). A comparison between GAKBS and a simple GA (GA which does not include problem-specific structure). It appeared that GAKBS performed better than simple GA in terms of finding optimal costs.

In [3], Gen et al. used genetic algorithm for topological network design based on spanning trees. Here, two criteria are optimized: average network message delay,

and connecting cost. They have used two main entities which they call service centers (e.g., a bridge) and nodes (i.e., user) which are connected to service centers. The constraints they have considered are that the number of nodes connected to a service center must not exceed the capacity of the center and that the traffic flowing through a service center must not exceed its traffic capacity. They have used Prufer number [5] based encoding to represent their solution. Simulation analysis was done on different test cases and registered Pareto optimal solutions technique was used to give out the compromise solution and the TOPSIS method [3] was used to determine the best compromise solution among Pareto solutions.

Elbaum et al. [2] have used GA for designing LANs with the objective of minimizing the average network delay. The constraint they considered is that the flow on any link does not exceed the capacity of that link. The topology design approach they took includes issues such as determination of the number of segments in the network, allocating the users to different segments, and determining the interconnections and routing among the segments. They have used Huffman encoding to represent their solution. Lower bounds on the average network delay have been developed. Simulation results were obtained for different test cases. The average network delay of the obtained designs was compared with average network delay lower bounds. Results suggested that the GA performs well for the test cases considered.

2.3 Conclusion

In this chapter, literature related to network topology design has been reviewed. The chapter covers different approaches that have been taken in the network topology design problems and the objectives they optimize.

Chapter 3

Preliminaries

3.1 Introduction

In this chapter, necessary problem specific information is included. This information is required in order to understand concepts, terminology, and related work described in subsequent chapters. Section 3.2 gives the assumptions and terminology we have adopted. The formal definition of the TDEN problem, along with the notation, is given in Section 3.3. Section 3.4 describes the objective function for TDEN problem. In Section 3.5, we describe the objective values computation as well as the constraints we considered in our research.

This research is based on the use of the simulated evolution algorithm. Thus, a quick review of this heuristic is given in Section 3.6. A primer on fuzzy logic is given in Section 3.7. The chapter ends with a conclusion in Section 3.8.

3.2 Assumptions and Terminology

- A single user (station) is called a “node”.
- A “LAN segment” (or simply a segment) is a part of a cluster and is comprised of nodes.
- A “cluster” (also called Local Site) is an interconnected collection of LAN segments.
- The “capacity” of a network device is defined as the number of interfaces it has.
- The location of a node within a cluster can be represented by its (x, y) coordinates with respect to some reference point.
- A node is either 10/100baseT Ethernet [31, 32, 33] or Token Ring type [31, 32, 33].
- A cluster is either made of 10/100baseT Ethernet segments only or all Token Ring segments.
- The backbone is assumed to be running on Fast Ethernet using Fiber Optic. The “root” node is a collapsed backbone with given Ethernet and Token Ring interfaces.

- Within a local site, only Category 5 cable [34, 35] is used, while between two local sites, only fiber optic cable [34, 35, 31] is used.
- If the distance between two local sites is less than 2 km, multi-mode fiber optic cable is used. If distance is less than 3 km, a single-mode fiber optic cable is used [34, 35].
- Class C networks are assumed. Therefore, we limit the number of nodes per local site to at most 254.
- The number of segments is known a priori and nodes have already been assigned to segments.
- Hubs, bridges, routers and other networking devices cannot be placed in any location. There are designated locations to do so.

3.3 Formal Description and Notation

Topology design of enterprise networks is formally described as follows:

1. Let O be the set of n local sites: $O = \{o_1, o_2, \dots, o_n\}$.
2. Let L be the set of m backbone links: $L = \{l_1, l_2, \dots, l_m\}$ and $|L| = \frac{n(n-1)}{2}$.
3. Let R be the root node.

4. S^i is the set of p segments associated with local site $o_i \in O$: $S^i = \{s_1^i, s_2^i, \dots, s_p^i\}$, and let $S = \{S^1, S^2, \dots, S^i, \dots, S^d\}$. The cardinality of set S^i , i.e., $|S^i|$, is generally different for each local site $o_i \in O$.
5. F^i is the set of w networking devices associated with local site $o_i \in O$: $F^i = \{f_1^i, f_2^i, \dots, f_w^i\}$, and let $F = \{F^1, F^2, \dots, F^i, \dots, F^n\}$. The cardinality of set F^i , $|F^i|$, is generally different for each local site $o_i \in O$.
6. Q^i is the set of k local site links associated with local site $o_i \in O$: $Q^i = \{q_1^i, q_2^i, \dots, q_k^i\}$, and let $Q = \{Q^1, Q^2, \dots, Q^i, \dots, Q^n\}$. The cardinality of set Q^i , i.e., $|Q^i| = \frac{w(w-1)}{2}$.
7. $U = [u_{ij}]$ is a $n \times n$ traffic matrix where u_{ij} represents the traffic from local site o_i to local site o_j .
8. $V^l = [v_{ij}]$ is a $w \times w$ traffic matrix where v_{ij} represents the traffic from network device f_i to network device f_j within local site O^l .
9. $A = [a_{ij}]$ is a $n \times n$ distance matrix where a_{ij} represents the distance from local site o_i to local site o_j .
10. Let $E = [e_{ij}]$ is a $w \times w$ distance matrix where e_{ij} represents the distance from a network device f_i to another network device f_j within the same local site.

Given the above, a topology has to be designed such that:

1. n local sites and root R are connected together using links from set L , thus forming the spanning tree backbone.

2. m segments of a local site α_i are connected to w network devices.
3. w network devices of a local site α_i are interconnected.

The objectives to be optimized are monetary cost, network average delay, and maximum number of hops between any source destination pair, while satisfying some constraints, which follow in Section 3.5.4.

The notations that has been used is presented below.

n number of clusters.

m number of LAN segments in a cluster.

T^b $n \times n$ local site topology matrix where $t^b_{ij} = 1$, if local sites i and j are connected and $t^b_{ij} = 0$ otherwise.

T^l $w \times w$ within-cluster topology matrix where $t^l_{ij} = 1$, if network devices i and j are connected and $t^l_{ij} = 0$ otherwise.

X^b $n \times 1$ coordinate vector which gives the x coordinate of each local site (i.e., x coordinate of the network device which is connecting the local site to the backbone).

Y^b $n \times 1$ coordinate vector which gives the y corrdinate of each local site (i.e., y coordinate of the network device which is connecting the local site to the backbone).

c_{fo}	cost of fiber optic cable per unit length.
c_{tp}	cost of twisted pair cable per unit length.
c_{switch}	cost of a switch.
c_{router}	cost of a router.
c_{hub}	cost of a single port of a hub.
ω_i	traffic within cluster i .
λ_i	traffic on link i .
$\lambda_{max,i}$	capacity of link i .
L	number of links of the proposed topology.
D_{nd}	delay due to network devices.
g_i	maximum number of cluster which can be connected to cluster i .
γ_{ij}	external traffic between clusters i and j .
γ	overall external traffic.
S_k	the forwarding speed of network device k .
h_{ij}	number of hops between cluster i and cluster j .
P_{router}	the maximum number of ports of a router.
P_{switch}	the maximum number of ports of a switch.
P_{hub}	the maximum number of ports of a hub.

3.4 Objective Function

As mentioned earlier, topology design of enterprise networks is a hard problem. There could be several criteria, such as cost, delay, throughput, utilization, number of hops etc., which could be optimized. In this research, the criteria in the objective function are: monetary cost, maximum number of hops between any source-destination pair, and average network delay.

3.5 Computation of Objective Values

3.5.1 Monetary cost

The goal of monetary cost optimization is to find the topology with minimum possible cost, while meeting all the requirements and constraints. There are several factors that govern the cost of the network. This includes costs of cabling, routers, switches, hubs, etc. To further categorize, there are several categories of the same type of cables, where each category is suitable for a certain kind of application. Similarly, network devices are also available with many different combinations of technologies and ports, which change their cost drastically. Thus, the function for monetary cost has been devised considering these facts. Before proceeding further, it is important to mention that, as indicated earlier, we have to design topology at two levels: backbone level, and local site level. Since the cost of the cable and the

cost of the network devices are the two main entities affecting the monetary cost, our function for monetary cost can mathematically be defined as:

$$cost = (s \times c_{cable}) + (c_{nd}) \quad (3.1)$$

where s represents the total length of cable, c_{cable} represents the cost per unit of the cable used (whether fiber optic or twisted pair), and c_{nd} represents the combined costs of all the routers, switches, and hubs used. This function is used for calculating the monetary cost of the backbone topology design as well as local site topology design.

3.5.2 Average network delay

Another important criterion to consider is the average network delay per packet. The goal, of course, is to minimize the delay as much as possible, while considering the constraints and requirements. A high network average delay would show that there is a very high link utilization and load has built up on network devices. A low average network delay would mean a faster travel of information, thus a good quality of service to users.

To devise a suitable function for average network delay, we proceed as follows: We use an M/M/1 model to describe the behavior of a link and network device. For M/M/1 model, the average delay of the next packet sent over link i is expected to

be [2]:

$$D_i = \frac{1}{\mu C_i - \lambda_i} \quad (3.2)$$

where C_i is the capacity of link i , $\frac{1}{\mu}$ is the average packet size in bits, and λ_i is the average number of packets that traverse link i for given topology. The delay per bit due to network device between local site i and j is $B_{i,j} = \mu b_{i,j}$, where $b_{i,j}$ is the delay per packet. If γ_{ij} is the total traffic through the network device between local sites i and j , then the average delay due to all network devices is:

$$D_{nd} = \frac{1}{\gamma} \sum_{i=1}^d \sum_{j=1}^d \gamma_{ij} B_{ij} \quad (3.3)$$

Given the above definitions, we conclude that the total average network delay is composed of delays of links and network devices. The average network delay D is therefore [2]

$$D = \frac{1}{\gamma} \sum_{i=1}^m \frac{\lambda_i}{\lambda_{max,i} - \lambda_i} + \frac{1}{\gamma} \sum_{i=1}^n \sum_{j=1}^n \gamma_{ij} B_{ij} \quad (3.4)$$

3.5.3 Maximum number of hops

The maximum number of hops between any source destination pair is found through a subroutine in the program. The subroutine starts from one node and keeps counting the number of nodes until it reaches the destination node.

3.5.4 Constraints

Four important constraints are used in this research. These are explained below.

1. The first set of constraints is due to the limitation of the capacity of the links. A good network would be one in which links are “reasonably” utilized, otherwise this would cause delays, congestion, and packet loss. Thus the traffic flow on any link i must never exceed a threshold value:

$$\lambda_i < \lambda_{max,i} \quad i = 1, 2, \dots, e \quad (3.5)$$

2. The second constraint is that the number of clusters attached to a network device G must not be more than the port capacity of that device. Mathematically, we can represent this constraint as:

$$\sum_{j=1}^n t_{ij}^b < g_i \quad i = 1, 2, \dots, n \quad \forall i \neq j \quad (3.6)$$

3. The third possible constraint is that the designer might like to enforce certain hierarchies on the network devices. For example, he might not allow a hub to be the parent of a router or a switch to be the parent of a router. This constraint is dependent on the choice of the designer.

4. The last constraint is that the total number of links must be equal to one less than the number of local sites (if we are dealing at the backbone level) to maintain a spanning tree topology. Thus this constraint can be represented mathematically as:

$$L = \delta - 1 \quad (3.7)$$

where δ represents n at backbone level and w at the local site level.

3.6 Simulated Evolution

Simulated Evolution (SE) is a general iterative heuristic proposed in [16]. It falls in the category of algorithms which emphasize the behavioral link between parents and offspring, or between reproductive populations, rather than the genetic link [15]. This scheme combines iterative improvement and constructive perturbation and saves itself from getting trapped in local minima by using stochastic approach. It iteratively operates a sequence of evaluation, selection and allocation (perturbation) on one solution. Using a time homogeneous irreducible Markov chain formulation, Kling and Banerjee [19] showed that SE algorithm converges in the limit to a global minimum with probability one.

In this section we will review simulated evolution (SE) algorithm in detail, as given in [20].

Description of SE Algorithm

Simulated Evolution is a general heuristic for solving a variety of combinatorial optimization problems. The SE proceeds as follows. It starts with a randomly or constructively generated valid initial solution. The main loop of the algorithm consists of three steps: **evaluation**, **selection** and **allocation**. These steps are carried out repetitively in a main loop until some stopping condition is satisfied. Other than these three steps, some input parameters for the algorithm are initialized in an earlier step known as **initialization**. Following is the description of these procedures. The pseudo code of the algorithm is given in Figure 3.1.

Initialization

Initialization step is carried out only once. It consists of selecting a starting valid solution for the problem under consideration. This solution can be generated randomly or the output of any constructive heuristic. The other important parameters which are initialized in this step are a stopping condition and selection bias (B). Different stopping conditions can be used. For example, the stopping condition can be a fixed number of iterations of the main loop or a function of improvement in the solution cost. Selection bias is used to compensate errors made in the estimation of the optimum cost used in the computation of goodness (see the evaluation stage below). The selection bias controls the magnitude of the perturbation of current solution. It effects the overall execution speed of the algorithm and the quality of

Simulated_Evolution($B, \Phi_{\text{initial}}, \text{StoppingCondition}$)

NOTATION

B = Bias Value. Φ = Complete Solution.
 e_i = Individual link in Φ . O_i = Lower bound on cost of i^{th} link.
 C_i = Current cost of i^{th} link in Φ . g_i = Goodness of i^{th} link in Φ .
 S = Queue to store the selected links.
 $\text{ALLOCATE}(e_i, \Phi_i)$ =Function to allocate e_i in partial solution Φ_i

Repeat

EVALUATION: **ForEach** $e_i \in \Phi$ **DO**
 begin
 $g_i = \frac{O_i}{C_i}$
 end

SELECTION: **ForEach** $e_i \in \Phi$ **DO**
 begin
 IF $\text{Random} > \text{Min}(g_i + B, 1)$
 THEN begin
 $S = S \cup e_i$; Remove e_i from Φ .
 end
 end

Sort the elements of S

ALLOCATION: **ForEach** $e_i \in S$ **DO**
 begin
 $\text{ALLOCATE}(e_i, \Phi_i)$
 end

Until *Stopping Condition is satisfied*
 Return Best solution.
(Simulated_Evolution)

Figure 3.1: Structure of the simulated evolution algorithm.

the final solution. A carefully selected value of the bias results in a good quality solution.

Evaluation

In this step, each individual member of the solution is evaluated on the basis of problem constraints and objectives. This evaluation is represented by the **goodness** for each element of the current solution. The goodness of an element of the design is defined as follows.

$$g_i = \frac{o_i}{a_i} \quad (3.8)$$

where o_i is the optimum value of the cost of element i and a_i is the actual cost estimate for this element in the current design. The goodness represents a measure of how near each element is to its optimum position. As is obvious from Equation 3.8, the goodness of an element is between 0 and 1. A value of goodness near 1 means that element i is near its optimum location.

Selection

The goodness is used to probabilistically select elements in the **selection** step. Elements with low goodness have a higher probability of getting selected for reposition. Selection bias (B) is used to compensate errors made in estimation of the optimum cost. Its objective is to inflate or deflate the goodness of elements. A high positive value of bias decreases the probability of selection or vice versa. A carefully tuned

bias value results in good solution quality and reduced execution time [19]. The selection step results in a partial solution of only unselected elements, while selected elements are saved in a queue for allocation.

Allocation

The purpose of the allocation is to perturb the current solution in such a way that better links are selected to form the topology. The allocation function affects the quality of solution as well as convergence of the search. Different constructive allocation schemes are proposed in [20]. One such scheme is **sorted individual best fit**, where all the selected elements are sorted in descending order in a queue with respect to desired criteria. The sorted elements are removed one at a time and *trial* moves are carried out for all the available empty positions. The element is *finally* placed in a position where maximum reduction in cost for the partial solution is achieved. This process is continued until the selected queue is empty. The overall complexity of this algorithm is $O(s^2)$ where s is the number of selected elements. Other more elaborate allocation schemes are **weighted bipartite matching allocation** and **branch-and-bound search allocation** [20]. However, these schemes are more complex allocation strategies than “sorted individual best fit”, but result in comparable solution quality [20].

3.7 Fuzzy Logic

3.7.1 Fuzzy Set Theory (FST)

A crisp set is normally defined as a collection of elements or objects $x \in X$ that can be finite, countable or uncountable. Each single element can either belong to a set or not. However, in real life situations objects do not have crisp [1 or 0] membership criteria. Fuzzy Set Theory (FST) aims to represent vague information, like ‘low load’, ‘high load’, or ‘low latency’, etc., which are difficult to be represented in classical(crisp) set theory. A fuzzy set is characterized by a membership function which provides a measure of the degree of presence for every element of the set [36, 37]. A fuzzy set A of a universe of discourse X is defined as $A = \{(x, \mu_A(x)) \mid \text{all } x \in X\}$, where $\mu_A(x)$ is a membership function of $x \in X$ being an element in A . Figure 3.2 shows one example of a membership function.

Like crisp sets, set operations such as union, intersection, and complementation etc., are also defined on fuzzy sets. There are many operators for fuzzy union and fuzzy intersection. For fuzzy union, the operators are known as **s-norm** operators (denoted as \oplus). While fuzzy intersection operators are known as **t-norm** (denoted as $*$). Some examples of **s-norm** operators are given below, (where A and B are fuzzy sets of universe of discourse X) [36].

1. Maximum. $[\mu_{A \cup B}(x) = \max[\mu_A(x), \mu_B(x)]]$.

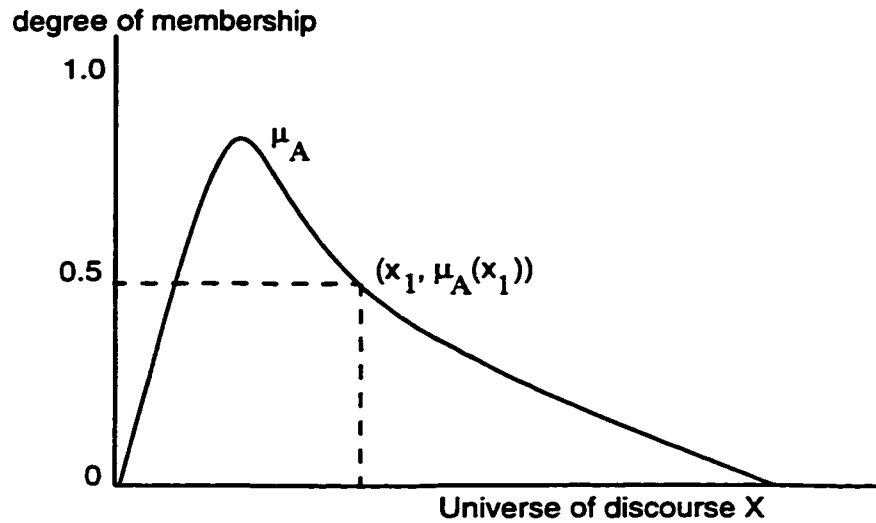


Figure 3.2: Membership function for a fuzzy set A.

2. Algebraic sum. $[\mu_{A \cup B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x)]$.
3. Bounded sum. $[\mu_{A \cup B}(x) = \min(1, \mu_A(x) + \mu_B(x))]$.
4. Drastic sum. $[\mu_{A \cup B}(x) = \mu_A(x)$ if $\mu_B(x) = 0$, $\mu_B(x)$ if $\mu_A(x) = 0$, 1 if $\mu_A(x), \mu_B(x) > 0]$.

An **s-norm** operator satisfies commutativity, monotonicity, associativity and $\mu_{A \cup 0} = \mu_A$ properties. Following are some examples of fuzzy intersection operators known as **t-norm**.

1. Minimum. $[\mu_{A \cap B}(x) = \min[\mu_A(x), \mu_B(x)]]$.
2. Algebraic product. $[\mu_{A \cap B}(x) = \mu_A(x)\mu_B(x)]$.
3. Bounded product. $[\mu_{A \cap B}(x) = \max(0, \mu_A(x) + \mu_B(x) - 1)]$.

4. Drastic product. [$\mu_A \cap_B(x) = \mu_A(x)$ if $\mu_B(x) = 1$, $\mu_B(x)$ if $\mu_A(x) = 1$, 0 if $\mu_A(x), \mu_B(x) < 1$].

Like s-norms, t-norms also satisfy commutativity, monotonicity, associativity and $\mu_A \cap_1 = \mu_A$ properties. Additionally, the membership function for fuzzy complementation operator is defined as.

$$\mu_{\bar{B}}(x) = 1 - \mu_B(x)$$

Ordered Weighted Averaging Operator

Generally, formulation of multi criteria decision functions do not desire pure “anding” of **t-norm** nor the pure “oring” of **s-norm**. The reason for this is the complete lack of compensation of **t-norm** for any partial fulfillment and complete submission of **s-norm** to fulfillment of any criteria. Also the indifference to the individual criteria of each of these two forms of operators led to the development of Ordered Weighted Averaging (OWA) operators [38]. This operator allows easy adjustment of the degree of “anding” and “oring” embedded in the aggregation. According to [38], “orlike” and “andlike” OWA for two fuzzy sets A and B are implemented as given in Equations 3.9 and 3.10 respectively.

$$\mu_A \cup_B(x) = \beta \times \max(\mu_A, \mu_B) + (1 - \beta) \times \frac{1}{2}(\mu_A + \mu_B) \quad (3.9)$$

$$\mu_{A \cap B}(x) = \beta \times \min(\mu_A, \mu_B) + (1 - \beta) \times \frac{1}{2}(\mu_A + \mu_B) \quad (3.10)$$

β is a constant parameter in the range $[0,1]$. It represents the degree to which OWA operator resembles a pure “or” or pure “and” respectively.

3.7.2 Fuzzy Reasoning

Fuzzy Logic is a mathematical discipline invented to express human reasoning in rigorous mathematical notation. Unlike classical reasoning in which a proposition is either true or false, fuzzy logic establishes approximate truth value of proposition based on linguistic variables and inference rules. A *linguistic variable* is a variable whose values are words or sentences in natural or artificial language [39]. By using hedges like ‘more’, ‘many’, ‘few’ etc., and connectors like AND, OR, and NOT with linguistic variables, an expert can form *rules*, which will govern the approximate reasoning.

3.7.3 Linguistic Variable

As defined in [40], a linguistic variable is a variable whose values are words or sentences in a natural or artificial language. A linguistic variable is characterized by a quintuple $(\Omega, T(\Omega), X, G, N)$, detailed as follows:

1. Ω is the name of the linguistic variable;
2. $T(\Omega)$ is the term-set of Ω , i.e., the collection of its linguistic values;

3. X is a universe of discourse;
4. G is a syntactic rule which generates the terms in $T(\Omega)$; and
5. N is a semantic rule which associates with each linguistic value its meaning.

$N(\omega)$ denotes a fuzzy subset of X for each $\omega \in T(\Omega)$. Let us take an example to clarify the meaning of a linguistic variable. Let X be the universe of *network average delay*, A be the fuzzy subset *network average delay near 0.05 seconds*, and $\mu_A(\cdot)$ be the membership function for A . In this example, *network average delay* is a linguistic variable ($\Omega = \text{network average delay}$). The linguistic values of *network average delay* can be defined as $T(\Omega) = (\text{very small delay}, \text{small delay}, \text{delay near 0.05 seconds}, \text{large delay}, \text{little large delay})$. Each linguistic value is characterized by a membership function which associates a meaning to that value. The universe of discourse X is a possible range of *network average delay* for some designs. $N(\omega)$ defines a fuzzy set for each linguistic value $\omega \in T(\Omega)$. The term-set of a linguistic variable is the collection of all its linguistic values.

3.7.4 Fuzzy Rules

One of the major components of a fuzzy logic system are *Rules*. The rules are expressed as logical implications. Fuzzy logic rules are “if-then ” rules and define relations between linguistic values of outcome (then-part, i.e., consequent) and linguistic values of condition (if-part, i.e., antecedent) [41]. They are expressed as:

IF u_1 is F_1^l and u_2 is F_2^l and ... u_p is F_p^l THEN v is G^l

where $l = 1, 2, \dots, M$.

F_i^l and G^l are fuzzy sets. For example:

If the monetary cost is low and maximum number of hops are low and network average delay is low Then the solution is good.

Here *monetary cost*, *maximum number of hops*, *network average delay*, and *solution* are linguistic variables and *low* and *good* are linguistic values.

Rules are a form of propositions. A *proposition* is an ordinary statement involving terms which have been defined. In traditional propositional logic, an implication is said to be *true* if one of the following holds:

- antecedent is true, consequent is true,
- antecedent is false, consequent is false, and
- antecedent is false, consequent is true.

Rules may be provided by experts or can be extracted from numerical data. In either case, *engineering rules* are expressed as a collection of IF-THEN statements.

Basically, building a rule needs the understanding of[41]:

1. linguistic variables versus numerical values of a variable;
2. quantifying linguistic variables by using fuzzy membership functions.
3. logical connections for linguistic variables;

4. implications, i.e., “IF A THEN B”;
5. additionally, combination of more than one rule.

3.7.5 Fuzzy Logic System (FLS)

A Fuzzy logic system (FLS) [36], shown in Figure 3.3, is a model of fuzzy based decision making in engineering applications. It accepts crisp data as input and converts it into fuzzy input sets through *fuzzifier* block. This is needed in order to activate rules which are expressed in terms of linguistic variables. The decision making is carried out in the *inference engine*, which is governed by the *rules*. The output of the decision making is fuzzy sets. If an application requires crisp output data then FLS contains a *defuzzifier* module to convert fuzzy output to crisp values. Generally, optimization applications do not require crisp output, therefore defuzzifier module is not used.

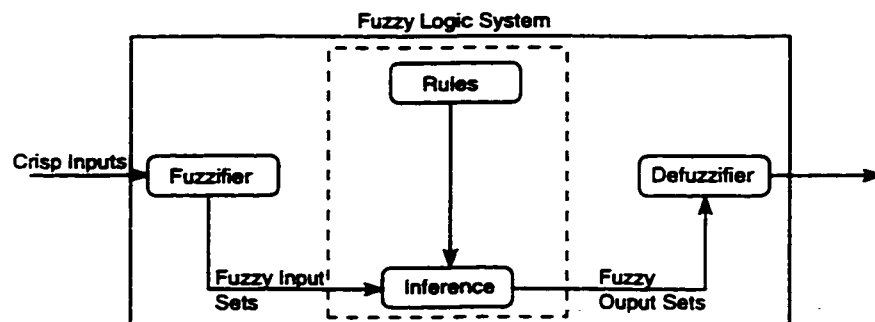


Figure 3.3: Fuzzy logic system.

3.8 Conclusion

In this chapter we have seen background material for subsequent chapters. We have defined the topology design of enterprise network (TDEN) problem and reviewed the cost functions with respect to monetary cost, average network delay, and maximum number of hops between a source-destination pair. The given inputs, assumptions, terminology, and constraints have also been mentioned. We have also reviewed the simulated evolution heuristic and fuzzy logic. This heuristic is a powerful stochastic iterative heuristic for general combinatorial optimization problems. It is based on the analogy between evolutionary processes and optimization. Underlying principle of this heuristic is that evolutionary system be subjected to the evaluation process such that it stochastically discards inferior parts and retains superior parts of the system. The constructive perturbation in allocation stage ensures that the algorithm will converge to a suboptimal solution. On the other hand, fuzzy logic provides a convenient algebra of combining conflicting objectives and expert human knowledge.

Chapter 4

Fuzzy Logic Based Simulated Evolution Algorithm for Network Topology Design

4.1 Introduction

In Chapter 2, we reviewed different literature with respect to iterative heuristics, which included genetic algorithm, and simulated annealing, as applied to topology design problem. Most of these included delay, or the cost, or both, for optimization. There are some other measures, such as maximum number of hops, which is also an important objective to optimize. All these criteria together make the topology design problem a multiobjective optimization problem. Thus, in our scheme, we

optimize three cost parameters of the topology: monetary cost, average network delay, and maximum number of hops between any source destination pair.

When dealing with conflicting objectives, the concept of optimum is not clear. Further, it is not clear how one can precisely compare two competing topologies when one topology is not better across all three criteria. Let C_i , D_i , and H_i be the monetary cost, delay, and maximum hops of a topology solution S_i . A solution S_i is said to dominate a solution S_j , denoted as $S_i \prec S_j$ iff

$$C_i < C_j$$

$$D_i < D_j$$

$$H_i < H_j$$

Obviously if $S_i \prec S_j$, then S_i is a better solution than S_j , and the solution that dominates all other solutions is the best solution. Unfortunately the relation \prec does not define a partial order on all possible solutions, i.e., there are numerous cases where two solutions only partially dominate each other with respect to one or two of the objective criteria. To deal with this problem, researchers traditionally adopted one of the following two approaches:

1. Ranking, where individual objectives are ranked against each other. For example cost would be number 1 objective, then delay, then maximum hops.

Then a solution S_i is better than S_j if

$$C_i < C_j, \text{ or}$$

$$(C_i = C_j \text{ and } D_i < D_j) \text{ or}$$

$$((C_i = C_j \text{ and } D_i = D_j) \text{ and } H_i < H_j)$$

2. Weighted sum approach, where we use a utility function that combines the individual criteria in a weighted sum, that is

$$f(C, D, H) = W_C C + W_D D + W_H H$$

where W_C , W_D , and W_H are the weights associated with cost, delay, and maximum hops. Usually $W_C + W_D + W_H = 1$. The problem with this approach is that it needs a careful tuning with respect to each objective that needs to be optimized. Furthermore, the results show good solutions only for the case on which the tuning has been made.

During the topology design process, some desirable objectives, such as the delay, can only be imprecisely estimated. Fuzzy logic provides a rigorous algebra for dealing with imprecise information. Furthermore, it is a convenient method of combining conflicting objectives and expert human knowledge. From the pseudocode of the SE algorithm given in Figure 3.1, it is clear that there are two phases of the algorithm which could be modeled to include multiple objectives. These phases are **evaluation** and **allocation**. We have used fuzzy logic based reasoning in these two phases. We have used two objectives in the evaluation phase and three objectives in the allocation phase.

In this chapter, the proposed scheme and implementation details are discussed.

This chapter is organized as follows. In Section 4.2, proposed schemes and step by step details of implementation of different stages of the SE algorithm are given. We first describe the generation of initial solution in Section 4.2.1. Following this, the proposed fuzzy evaluation scheme is presented in Section 4.2.2. After covering the selection process in Section 4.2.3, proposed fuzzy allocation scheme is described in Section 4.2.4. Stopping criteria is discussed in Section 4.3. A conclusion is given in Section 4.4.

4.2 Proposed Scheme and Implementation Details

This section describes our proposals of fuzzification of different stages of the SE algorithm. The description is combined with the implementation details of the SE algorithm for topology design.

4.2.1 Initialization

The initial topology, which is a spanning tree, is generated randomly, while taking into account the constraints mentioned in Chapter 3. Thus the initial solution is a valid solution. The initialization step works in the following manner. We start from the root node. Then, another node, say X is selected randomly and is trial connected to the root. If this connection does not violate any constraint, the link joining them

is made permanent, otherwise another node is selected randomly. Once this is done, a new node is randomly chosen, and can be tried to be connected either to the root, or to node X , which is also done randomly, while checking the feasibility constraints. This procedure is repeated until all the nodes are connected and a complete topology is formed. Note that only one link at a time is connected.

Some parameters are also initialized in this phase. These include the maximum number of iterations for which the algorithm has to be run, and the selection bias B . Selection bias B and its effect are discussed in Section 4.2.3.

4.2.2 Proposed Fuzzy Evaluation Scheme

The **goodness** of each individual is computed as follows. In our case, an individual is a **link** which interconnects two local sites (at the backbone level) or two network devices (at the local site level). In the *fuzzy evaluation scheme*, satisfaction of optimum monetary cost of a link and optimum depth of a link (with respect to the root) are combined using fuzzy logic. The following rule and the subsequent equation are used for this purpose.

Rule 4.1: **IF** a link is *near optimum cost* AND *near optimum depth*

THEN it has *high goodness*.

$$g_{li} = \mu^e(x) = \alpha^e \times \min(\mu_1^e(x), \mu_2^e(x)) + (1 - \alpha^e) \times \frac{1}{2} \sum_{i=1}^2 \mu_i^e(x) \quad (4.1)$$

The superscript e stands for **evaluation** and is used to distinguish similar notation in other fuzzy rules. In equation 4.1, $\mu^e(x)$ is the fuzzy set of *high goodness*, g_i is goodness value, and α^e is a constant. The $\mu_1^e(x)$ and $\mu_2^e(x)$ represent memberships in the fuzzy sets *near optimum monetary cost* and *near optimum depth*.

In order to find the membership of a link with respect to *near optimum monetary cost*, we proceed in following manner. From the cost matrix, which gives the costs of each and every link possible, we find the minimum cost of all the costs and maximum cost of all the costs. We take these minimum and maximum costs as the lower and upper bounds and call them “LCostMin” and “LCostMax” respectively and then find the membership of a link with respect to these bounds. Furthermore, In this work, we have normalized the monetary cost with respect to “LCostMax” as follows:

$$\mu_1(x) = \begin{cases} 1 & \text{if } \frac{LCost}{LCostMax} \leq \frac{LCostMin}{LCostMax} \\ \frac{LCostMax - LCost}{LCostMax - LCostMin} & \text{if } \frac{LCostMin}{LCostMax} \leq \frac{LCost}{LCostMax} \leq 1 \\ 0 & \text{if } \frac{LCost}{LCostMax} \geq 1 \end{cases} \quad (4.2)$$

In the same manner, we can find the membership of a link with respect to *near optimum depth*. The lower limit, which we call “LDepthMin” is taken to be a depth of 1 with respect to the root. This is logical to take as the lower bound since there can be no link which is connected to the root at a depth of zero. This is clarified in Figure 4.1. The upper bound, which we call “LDepthMax” is taken to be 1.5

times of the maximum depth generated in the initial solution or a maximum of 7. For example, if in the initial solution, the maximum depth turns out to be 4, then “LDepthMax” for the depth membership function would be 6. This is done to give flexibility to links which may have more depth than the one in the initial solution, to show that they are also important to be considered, to some extent. If we take the initial solution maximum depth as “LDepthMax”, then in the following iterations some links with higher depths will have a membership value of zero (with respect to depth membership function, of course) and thus they will not be able to play any role as far as depth is concerned. However, due to technological limitations, we have limited the maximum possible depth to 7, in the case when “LDepthMax” turns out to be more than 4. The reason for having the maximum depth of 7 is that the hop limit for RIP protocol is 15. This means that if a maximum depth of 7 is taken, then in the worst case we would have a total of 14 hops from a source to a destination. The membership function with respect to *near optimum depth* can be represented mathematically as follows:

$$\mu_2(x) = \begin{cases} 1 & \text{if } LDepth \leq LDepthMin \\ \frac{LDepthMax - LDepth}{LDepthMax - LDepthMin} & \text{if } LDepthMin \leq LDepth \leq LDepthMax \\ 0 & \text{if } LDepthMax \leq LDepth \end{cases} \quad (4.3)$$

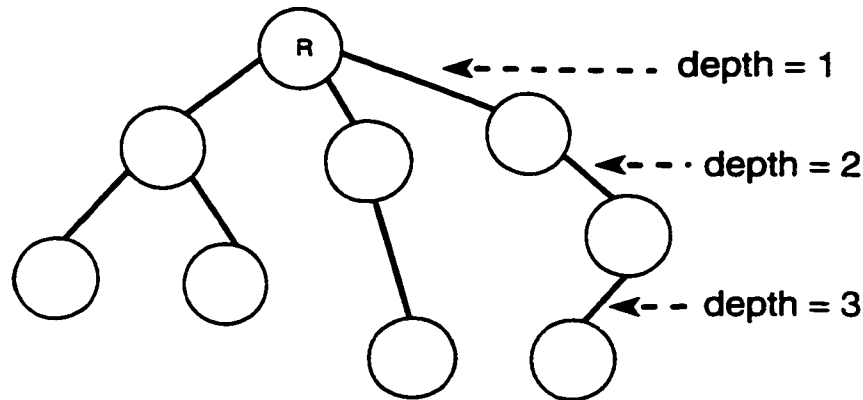


Figure 4.1: Depths of links with respect to the root.

The membership functions in graphical form for *near optimum monetary cost* and *near optimum depth* are given in Figures 4.2 and 4.3 respectively.

4.2.3 Selection

In this stage of the algorithm, for each link l_i , where $i = 1, 2, \dots, (n-1)$ currently present in the topology (at the backbone level) a random number in the range $[0,1]$ is generated and compared with $g_{c_i} + B$, where B is the selection bias. If the generated random number is greater than $g_{c_i} + B$ then link l_i is selected for allocation and considered removed from the topology. Thus the selection bias B is initially fixed so as to secure a selection set of a reasonable size.

In addition to the above approach, another bias methodology called *variable bias* has also been tried. This approach was suggested by Hussain et. al. [21]. The *variable bias* is a function of *quality of solution*. Average link goodness is a measure of how many “good” links are present in the topology. The bias value changes

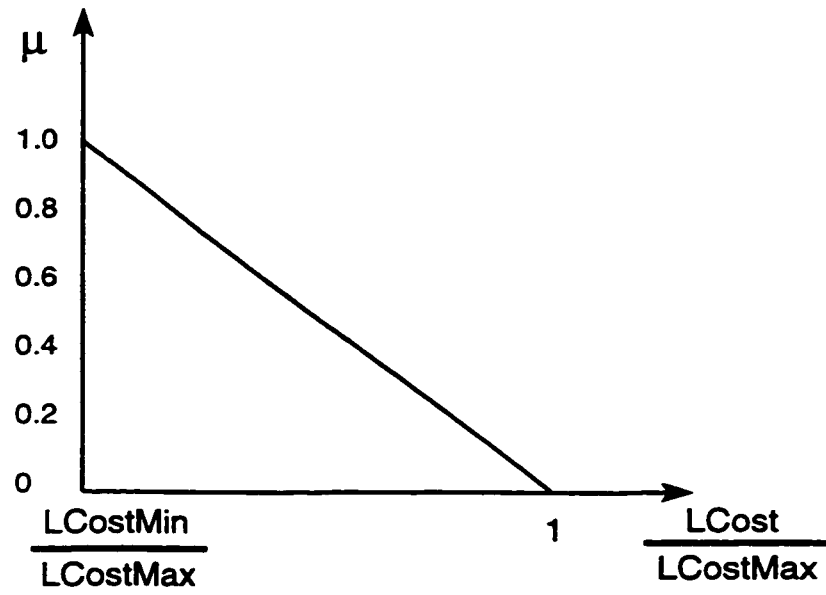


Figure 4.2: Membership function for monetary cost of a link.

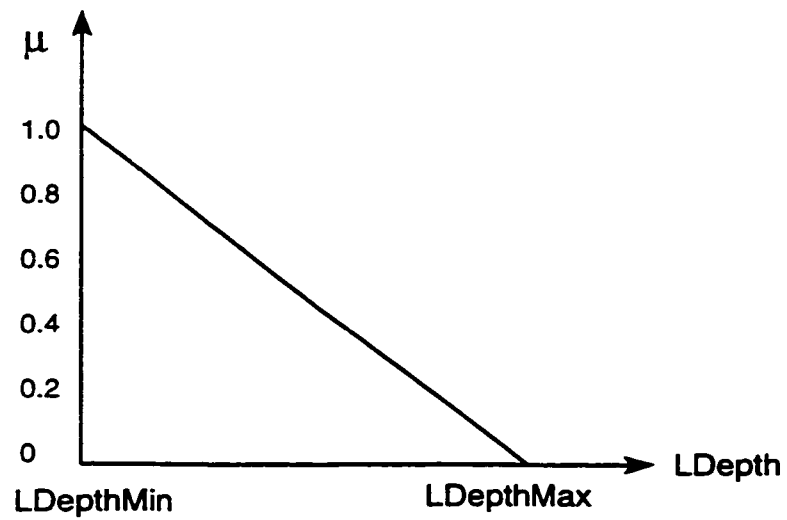


Figure 4.3: Membership function for depth of a link.

from iteration to iteration depending on the quality of solution. The *variable bias* is calculated as follows:

$$B_k = 1 - G_{k-1}$$

where B_k is the bias for k^{th} iteration and G_{k-1} is average goodness of all the links at the end of $(k - 1)^{\text{st}}$ iteration. Interested readers are referred to [21].

4.2.4 Proposed Fuzzy Allocation Scheme

During the **allocation** stage of the algorithm, the selected links are removed from the topology one at a time. For each removed link, new links are tried in such a way that they result in overall better solution. Before the allocation step starts, the selected links are sorted according to their goodness values, the link with the worst goodness being the head-of-line in the queue.

In earlier chapters we mentioned that there are three criteria to be optimized with respect to the overall solution. These criteria are *monetary cost*, *average network delay*, and *maximum number of hops between a source-destination pair*. In the *fuzzy allocation scheme*, these criteria are combined using fuzzy logic to characterize a good topology, as depicted in Figure 4.4.

The following rule and the subsequent equation are used for this purpose.

Rule 4.2: **IF** a solution has *low monetary cost* AND *low average network delay*
AND *low maximum number of hops between any source-destination pair*

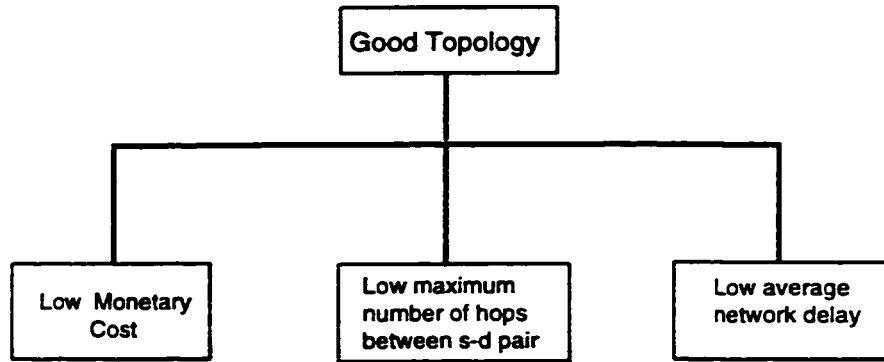


Figure 4.4: Basic components for a good topology.

THEN it is a *good topology*.

$$\mu^a(x) = \beta^a \times \min(\mu_1^a(x), \mu_2^a(x), \mu_3^a(x)) + (1 - \beta^a) \times \frac{1}{3} \sum_{i=1}^3 \mu_i^a(x) \quad (4.4)$$

where $\mu^a(x)$ is the membership value for solution x in the fuzzy set *good topology* and β^a is a constant in the range $[0,1]$. The superscript a stands for allocation. Here, μ_i^a for $i = \{1,2,3\}$ represents the membership values of solution x in the fuzzy sets *low monetary cost*, *low average network delay*, and *low maximum number of hops between any source-destination pair* respectively. The solution which results in the maximum value for Equation 4.4 is reported as the best solution found by the SE algorithm. Below we will see how to get the membership functions for the three criteria mentioned above.

Membership Function for Monetary Cost

First, we determine two extreme values for monetary cost, i.e., the minimum and maximum values. The minimum value, "TCostMin", is found by using the Esau-Williams algorithm, with all the constraints completely relaxed. This will surely give us the minimum possible monetary cost of the topology. The maximum value of monetary cost, "TCostMax", is taken to be the monetary cost generated in the initialization step. The monetary cost is normalized with respect to "TCostMax". The shape of the membership function is shown in Figure 4.5. The membership value for the normalized monetary cost is then computed using Equation 4.5.

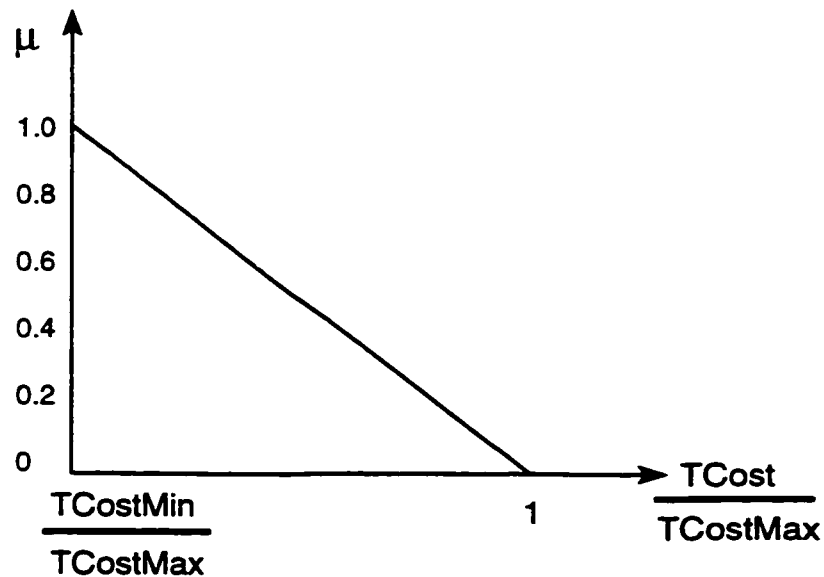


Figure 4.5: Membership function for monetary cost.

$$\mu_1(x) = \begin{cases} 1 & \text{if } \frac{TCost}{TCostMax} \leq \frac{TCostMin}{TCostMax} \\ \frac{TCostMax - TCost}{TCostMax - TCostMin} & \text{if } \frac{TCostMin}{TCostMax} \leq \frac{TCost}{TCostMax} \leq 1 \\ 0 & \text{if } \frac{TCost}{TCostMax} \geq 1 \end{cases} \quad (4.5)$$

Membership Function For Average Network Delay

We determine two extreme values for average network delay. The minimum value, “TDelayMin”, is found by connecting all the nodes to the root directly ignoring all the constraints and then calculating the average network delay using Equation 3.4. The maximum value of average delay, “TDelayMax”, is taken to be the average delay generated in the initialization step. The average delay is normalized with respect to “TDelayMax”. The shape of the membership function is shown in Figure 4.6. The membership value for the normalized average delay is computed using Equation 4.6

$$\mu_2(x) = \begin{cases} 1 & \text{if } \frac{TDelay}{TDelayMax} \leq \frac{TDelayMin}{TDelayMax} \\ \frac{TDelayMax - TDelay}{TDelayMax - TDelayMin} & \text{if } \frac{TDelayMin}{TDelayMax} \leq \frac{TDelay}{TDelayMax} \leq 1 \\ 0 & \text{if } \frac{TDelay}{TDelayMax} \geq 1 \end{cases} \quad (4.6)$$

Membership Function For Maximum Number of Hops Between any Source-Destination Pair

Again, two extreme values are determined. The minimum value, “THopsMin”, is taken to be 1 hop, which will be the minimum possible in any case. The maximum

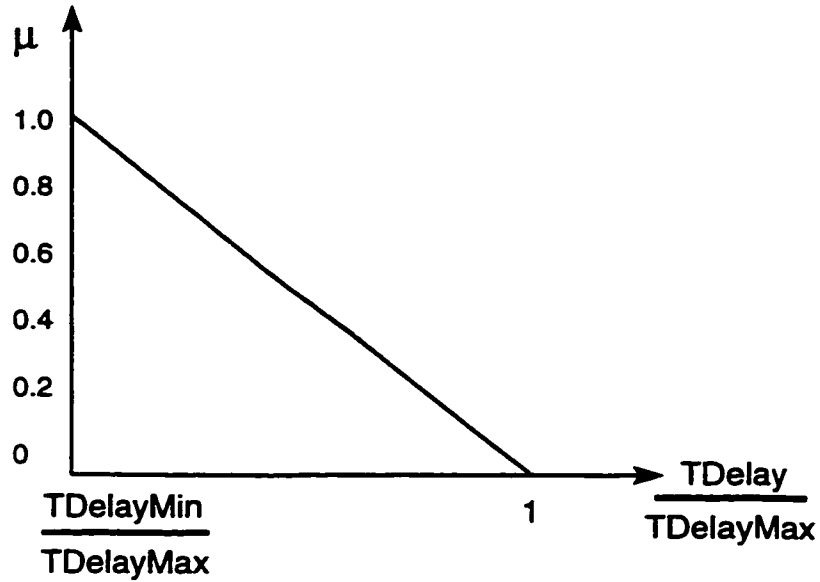


Figure 4.6: Membership function for average network delay.

value, “ $THopsMax$ ”, is taken to be the maximum number of hops between any source-destination pair generated in the initialization step. The shape of the membership function is shown in Figure 4.7. The membership value for the normalized average delay is computed using Equation 4.7

$$\mu_3(x) = \begin{cases} 1 & \text{if } THops \leq THopsMin \\ \frac{THopsMax - THops}{THopsMax - THopsMin} & \text{if } THopsMin \leq THops \leq THopsMax \\ 0 & \text{if } THopsMax \leq THops \end{cases} \quad (4.7)$$

In the proposed allocation scheme, all the selected links are removed one at a time and trial links are placed for each removed link. We start with the head-of-line

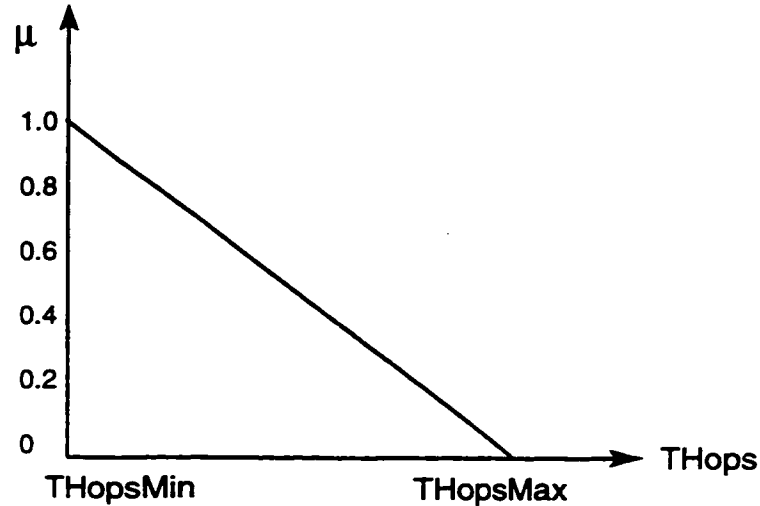


Figure 4.7: Membership function for maximum number of hops between a source-destination pair.

link, i.e. the link with the worst goodness. We remove this link from the topology. This divides the topology into two disjoint topologies, as depicted in Figure 4.8. Now the placing of trial links begins. In this work, the approach to place trial links is as follows. At most ten trial moves (i.e., trial links) are evaluated for each removed link. One point to mention is that for the ten moves, some moves may be invalid. However, we search for only four valid moves. Whenever four valid moves are found, we stop, otherwise continue until a total of ten moves are evaluated (whether valid or not). The removal of a link involves two nodes P and Q , of which node P belongs to the subtree which contains the root node and node Q belongs to the other subtree, as shown in Figure 4.8. For the ten moves we make, five of them are controlled and five are random. For the controlled moves, we start with node

Q and five *nearest* nodes in the other subtree are tried. For the random moves, we select any two nodes in the two subtrees and connect them. The process is shown in Figures 4.9 and 4.10.

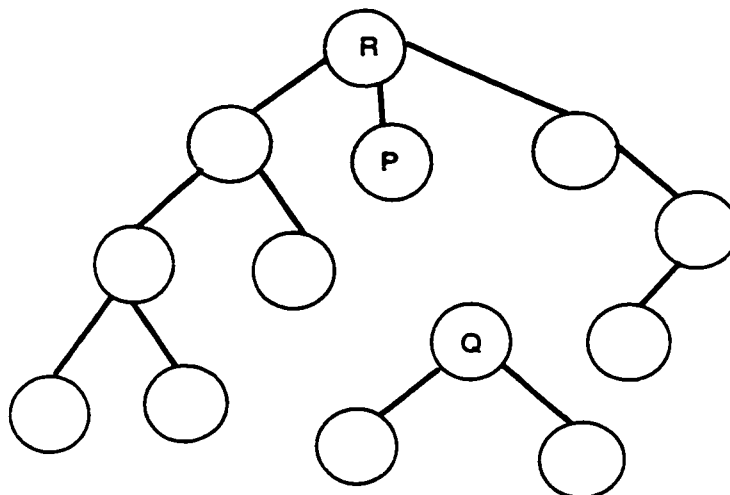


Figure 4.8: Two disjoint graphs containing nodes P and Q.

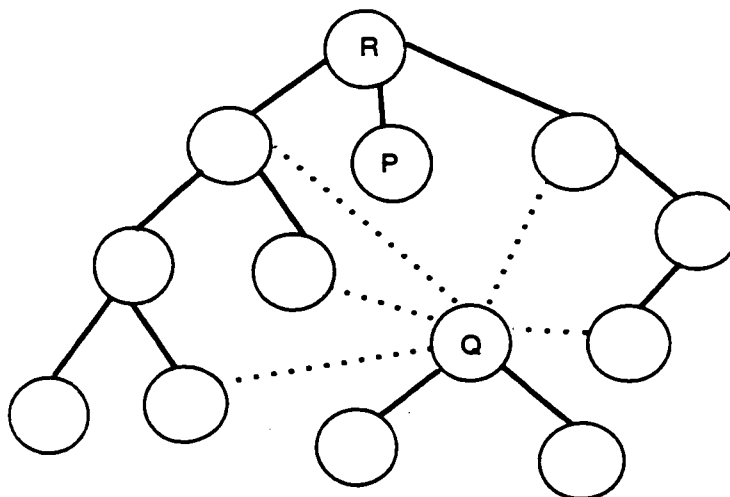


Figure 4.9: Controlled moves. The dotted lines show five controlled moves.

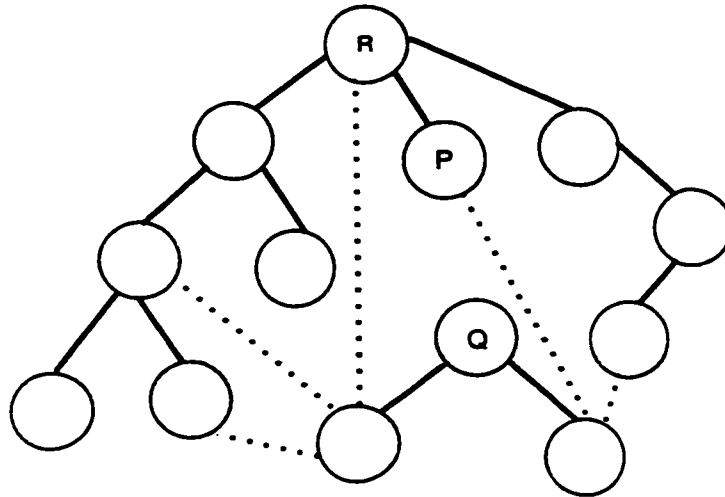


Figure 4.10: Random moves. The dotted lines show five random moves.

It may so happen that all the ten moves are invalid, in which case the original link is placed back in its position. The valid moves are evaluated based on Equation 4.4 and the best move among the ten moves is made permanent. This procedure is repeated for all the links that are present in the set of selected links.

4.3 Stopping Criterion

In our experiments, we have used a fixed number of iterations as a stopping criterion. We experimented with different values of iterations and found that for all the test cases, the SE algorithm converges within 4000 iterations or less. For smaller graphs, a major reduction in fuzzy cost occurs in the first 2000 iterations, while for bigger circuits, major reduction in fuzzy cost occurs in the first 2500 iterations.

4.4 Conclusion

The topology design of enterprise network is a hard and ill-defined problem with many conflicting objectives. In order to identify the best solution generated by the SE algorithm, we proposed fuzzification of **evaluation** and **allocation** stages of the simulated evolution algorithm. The proposed fuzzy evaluation scheme uses cost and depth of a link for the goodness computation. It combines three objectives, which are cost, average delay, and maximum number of hops between a source-destination pair, and optimizes these three objectives through the fuzzy function that combines them.

Chapter 5

Experiments and Results

5.1 Introduction

In this chapter we will discuss the experiments conducted and the results obtained. There are three main stages of topology design of enterprise networks. These stages are

- **Stage 1:** Assignment of LAN segments to network devices in each local site, where a local site will make up a single LAN.
- **Stage 2:** Design of the internal structure of each local site (i.e., in what topology are the network devices with LAN segments of a local site connected).
- **Stage 3:** Backbone design, where the local sites are connected to the backbone.

To achieve these objectives, we have used

1. Augmenting Path algorithm for stage 1.
2. Fuzzy Simulated Evolution algorithm for stage 2, and
3. Fuzzy Simulated Evolution Algorithm for stage 3.

The chapter is organized as follows. We will start with Backbone design (stage 3) in Section 5.2, and discuss the experiments and results obtained for this stage. Section 5.3 discusses the results for stage 1. Section 5.4 discusses the experiments and results for stage 2. A conclusion is given in Section 5.5.

5.2 Backbone Design Using Fuzzy Simulated Evolution Algorithm

This section summarizes the experimental study of variations of SE algorithms implemented in this work, namely

1. Fuzzy evaluation and fuzzy allocation based simulated evolution algorithm implementation with fixed bias controlled selection. This implementation is labelled as SE_FF.
2. Fuzzy evaluation and fuzzy allocation based simulated evolution algorithm implementation with variable bias controlled selection. This implementation

is labelled as SE_VB.

3. Fuzzy evaluation and fuzzy allocation based simulated evolution algorithm implementation where the allocation scheme incorporates Tabu Search characteristics. This implementation has variable bias controlled selection and is identified as SE_TS.

The characteristics of above listed algorithm variations are summarized in Table 5.1. The parameter values for different stages of the SE algorithm are summarized in Table 5.2. For example, the values of α^e and β^a were chosen after several trials of combinations of these.

We tested our implementations of the simulated evolution algorithm on five arbitrary test cases. For each test case, the traffic generated by each local site was collected from real sites. Other characteristics, such as the number of ports on a network device, its type, etc., were assumed. However, the costs of the network devices and links were obtained from vendors. The characteristics of test cases are listed in Table 5.3. The smallest test network has 15 local sites and the largest has 50 local sites. Also, Table 5.4 lists the characteristics of the equipment used in our experiments.

Following sets of experiments are carried out.

1. Comparing SE with fixed bias (SE_FF) and variable bias (SE_VB).

2. Analyzing the effect of Tabu Search approach based allocation in SE_TS and understanding the effect of Tabu List size.
3. Comparing SE_FF and SE_TS.
4. Comparing the variation of SE that gives the best results with Esau-Williams algorithm.
5. Comparing the variation of SE that gives the best results with Simulated Annealing (SA) algorithm.

Algorithm	Evaluation	Selection	Allocation
SE_FF	Fuzzy (link cost, depth)	Fixed bias	Fuzzy (cost, delay, hops)
SE_VB	Fuzzy (link cost, depth)	Variable bias	Fuzzy (cost, delay, hops)
SE_TS	Fuzzy (link cost, depth)	Variable bias	Fuzzy (cost, delay, hops) with tabu search approach

Table 5.1: Classification of our SE implementations.

Stage	Parameters
Fuzzy Allocation	$\beta^a = 0.5$
Fuzzy Evaluation	$\alpha^e = 0.5$
Stopping Condition	Fixed 4000 iterations
Initial Solution	Random
Bias (B)	Fixed, Variable

Table 5.2: Parameter values for different stages of the SE algorithm.

Name	# of Local Sites	LCostMin	LCostMax	TCostMin	TDelayMin	Traffic
n15	15	1100	9400	325400	2.14296	24.63
n25	25	530	8655	469790	2.15059	74.12
n33	33	600	10925	624180	2.15444	117.81
n40	40	600	11560	754445	2.08757	144.76
n50	50	600	13840	928105	2.08965	164.12

Table 5.3: Characteristics of test cases used in our experiments. LCostMin, LCostMax, and TCostMin are in US\$. TDelayMin is in milliseconds. Traffic is in Mbps.

Parameter	Characteristic
Cost of backbone switch with FO interface (8 ports)	\$ 30000
Cost of router with FO/TP interface(4 ports)	\$ 20000
Cost of switch with FO/TP interface(8 ports)	\$ 15000
Cost of hub or MAU with FO/TP interface(12 ports)	\$ 5000
Cost of fiber optic cable	\$ 5 per meter
Cost of Category 5 UTP	\$ 0.5 per meter
Delay per bit due to forwarding device	250 μ sec.
Maximum traffic on a link allowed	60 %
Average packet size	500 bytes

Table 5.4: Equipment parameters and their characteristics assumed in our experiments.

5.2.1 Comparison of SE_FF and SE_VB

In the original Simulated Evolution algorithm for VLSI Placement, Kling and Banerjee proposed the use of **fixed selection bias** [20]. The selection bias provides a mechanism of compensating any errors made in the estimation of the optimum cost used in the computation of goodness. Furthermore, it is used to limit the size of the selection set, i.e., controlling the magnitude of the perturbation of current solution. In case of a fixed bias, the user provides the bias value before the execution of the algorithm. The bias affects the execution time as well as the quality of solution. A low bias value takes more execution time than a high bias value [19]. It also poses one question. How to find the most suitable bias value in advance, which results in best solution quality and minimum execution time? One solution to this problem is to do several trial runs of the algorithm with different bias values. However, these trial runs will result in finding the best bias values for only that instance of the problem. As the problem changes, or even instance of problem changes, another set of trial runs are required to find the best bias value. Therefore, it cannot be used as a general rule for any set of problems. Furthermore, it also causes excessive execution time for all the trial runs. In order to overcome these issues, Hussain et. al. [21] proposed the *variable bias scheme* which we have recalled in Section 4.2.3.

Table 5.5 shows the best solution achieved with different bias values for SE_FF for our test cases. It is noticeable that the bias value that gives the best results

varies from case to case. For example, for n15, a bias value of 0.2 seems to give the best results considering all the three objectives. Similarly, a bias value of 0.2 for n25, a bias value of 0.0 for n33, and a bias value of 0.3 for n40 and n50 appear to give the best results. It is also clear that as the bias value increases, the execution time decreases. It is due to the fact that by increasing the bias value, fewer links are selected resulting in less amount of time spent in trial placing of links carried out in the allocation stage of the algorithm. This is also supported by Figure 5.1. In this figure, the cardinality of selection set (for n50) is shown against the frequency of solutions generated by SE_FF for different bias values. For low bias values like 0.0 and 0.1, more solutions have large selection sets, while for higher bias values like 0.3, the frequency of solutions with smaller selection sets increases.

The effect of bias on the quality of final solution can be described as follows. When we have a very low bias value, a large number of links are selected in each iteration. This will decrease the algorithm's ability to replace links optimally due to uncertainty about unconnected links. It will also increase the execution time of the algorithm. In contrast, when bias is very high, algorithm's ability to get out of local minima is restricted because "badly placed" links have reduced chances of selection. Thus the size of the selection set, execution time as well as the quality of solution space reached are reduced [19].

The results in Table 5.5 point to the fact that proper bias value is highly dependent on the test case under consideration. It also endorses a fact mentioned

Test Case	Bias	Monetary Cost	Avg. Delay	Max. Hops	Exec. Time
n15	0.0	310350	5.25	4	5
	0.1	301300	3.208	6	5
	0.2	314400	3.282	5	4
	0.3	311700	3.658	6	3
	0.4	300750	3.899	8	3
n25	0.0	494775	4.277	10	15
	0.1	507225	4.298	7	9
	0.2	509050	4.26	7	5
	0.3	517505	4.48	7	4.8
	0.4	520110	4.953	8	4.7
n33	0.0	687760	4.729	8	40
	0.1	682435	4.989	9	16
	0.2	663145	6.121	12	15
	0.3	697310	17.057	7	6.5
	0.4	691180	6.718	9	6
n40	0.0	813520	9.725	14	124
	0.1	871215	4.696	9	65
	0.2	859720	5.418	9	22
	0.3	866900	4.126	8	12
	0.4	874325	5.936	9	9
n50	0.0	989655	7.526	12	256
	0.1	1071700	5.689	11	120
	0.2	1080340	6.83	9	34
	0.3	1061900	5.32	9	8
	0.4	1087390	5.629	8	6

Table 5.5: Best solution for different bias values. Monetary cost is in US \$, delay is in milli seconds per packet, and execution time is in minutes.

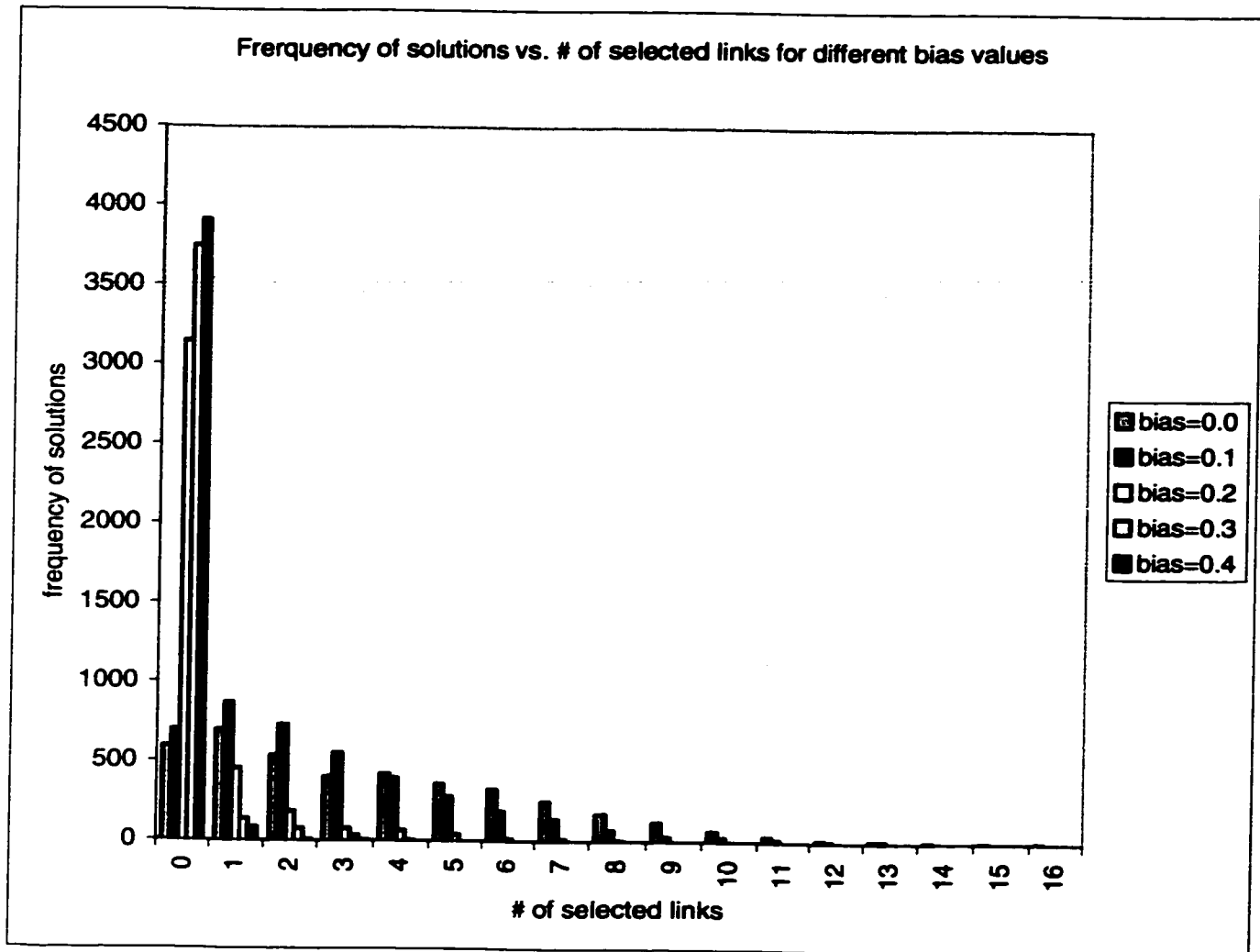


Figure 5.1: Caridinality of selection set for different bias values for test case n50.

earlier that it is not practical for the user to determine the best bias value in advance by carrying out several runs of the algorithm. The results of Table 5.5 confirm this assumption. For example, the best bias values for **n25** and **n40** are 0.2 and 0.3 respectively.

The *variable bias* (discussed in Section 4.2.4) was reported to have the following advantages:

1. Bias value is not arbitrarily selected and no trial runs are required to find the best bias value. The variable bias automatically adjusts according to the problem state.
2. For bad quality solutions, the average goodness is low, resulting in a high bias value. This will make sure that the size of selection set is not excessively large. It will save the algorithm from making big moves.
3. For good quality solutions, the average link goodness is high. Therefore, a low bias value is used. It will result in the selection of sufficient number of links and protect the algorithm from early convergence.

Table 5.6 compares the best solutions generated by best fixed bias SE_FF and SE_VB. The results for best fixed bias SE_FF are reproduced here for comparison purpose. It is clear from the table that, in general, SE_VB produces comparable results with SE_FF as far as “monetary cost” objective is concerned. For example, SE_VB produces a result 2.83 % better when compared with SE_FF in the case of

n15, while a result of 2.09 % inferior than SE_FF is attained with **n33**. The only exception where SE_VB gives a slightly significant improvement is with **n40** where a gain of 7.65 % is achieved compared with SE_FF.

As far as the “average network delay” and “maximum hops” objectives are concerned, a general trend is that SE_FF performs better than SE_VB. For example, in case of **n15**, SE_VB produces a solution of 25.99 % and 40 % inferior than SE_FF with respect to average network delay and maximum hops respectively. One deviation from this trend is seen in **n33**, where a gain of 12.5 % is achieved by SE_VB with respect to maximum hops. Another trend is that as the test case size increases, the performance of SE_VB deteriorates with respect to the average network delay.

Table 5.6 also gives the execution times for SE_FF and SE_VB. From this table it is clear that for small and medium size test cases (such as **n15**, **n25**, and **n33**), SE_VB has lower execution time than best fixed bias SE_FF, while for bigger cases (**n40** and **n50**), SE_VB has higher execution time than SE_FF. However, if we consider the time spent in trial runs of the SE_FF algorithm to find the best fixed bias, then SE_VB can be considered better than the fixed bias SE_FF. There were at least 3 trial runs with different bias values to identify the best value for each test case. The other advantage of SE_VB is that there is no need to run several trials.

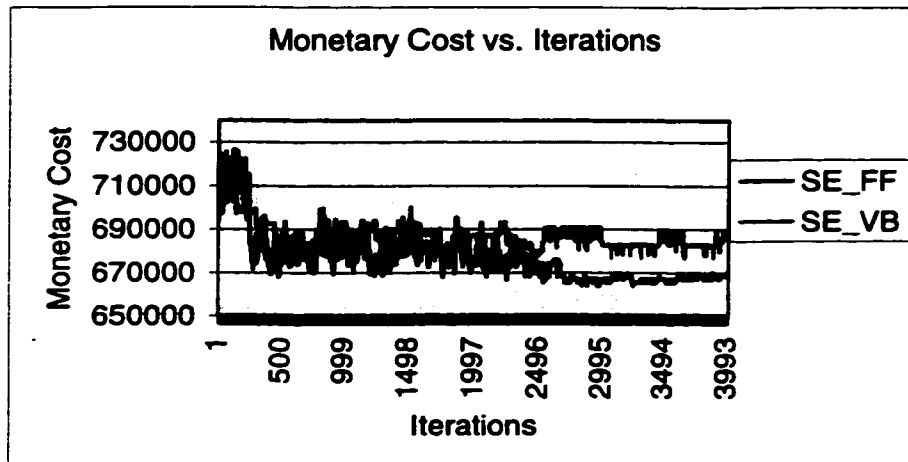
In order to compare the quality of search space of best fixed bias SE_FF and SE_VB, we plot different cost parameters versus iteration count of the algorithms for the test case **n33** (bias=0.0). Figure 5.2(a) and (b) respectively depicts the current

Case	SE_FF					SE_VB				% Gain			
	B	C	D	H	T	C	D	H	T	C	D	H	T
n15	0.2	314400	3.282	5	4	305500	4.135	7	1	2.83	-25.99	-40	75
n25	0.2	509050	4.26	7	5	512415	4.37	7	4.4	-0.657	-2.51	0	12
n33	0.0	687760	4.729	8	40	702815	5.319	7	17	-2.19	-12.48	12.5	57.5
n40	0.3	866900	4.126	8	12	800580	6.637	10	42	7.65	-60.85	25	-71.4
n50	0.3	1061900	5.32	9	8	1042080	8.236	10	62	1.87	-54.8	-11.1	-87.1

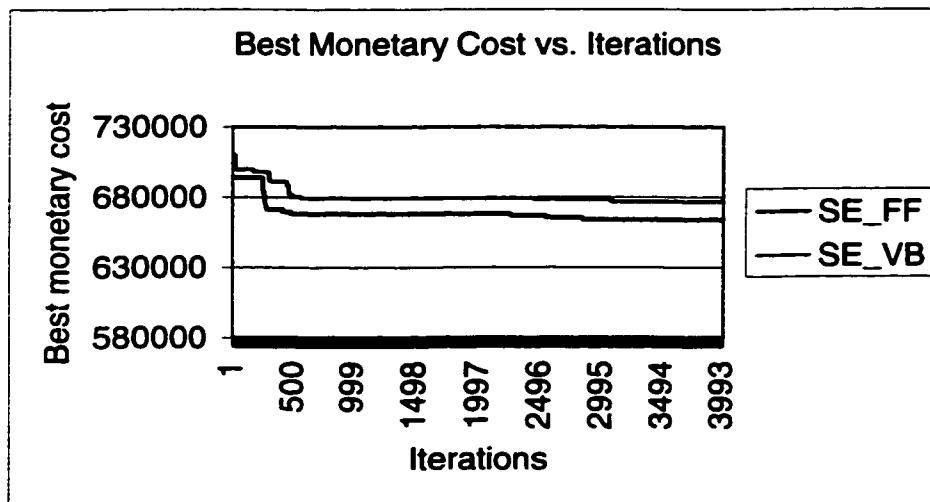
Table 5.6: Comparison of SE_FF and SE_VB. B = best bias, C = Cost in US \$, D = Delay in milli seconds per packet, H = hops, and T = execution time in minutes. The percentage gain shows the improvement achieved by SE_VB compared to SE_FF

and best monetary costs. From these plots, it is clear that SE_FF is able to achieve lower monetary cost than SE_VB. It is observed that SE_FF decreases the cost sharply and then carry out gradual refinements in rest of the iterations. However, SE_VB decreases cost smoothly and the trend continues till the end. Furthermore, SE_VB converges to a higher monetary cost than SE_FF. Figure 5.3(a) and (b) shows the current and best average network delay versus iterations for SE_FF and SE_VB. A similar behavior is observed here as with the monetary cost above. One notable thing is that the delay for SE_VB is smaller in earlier iterations and increases later on. However, there is smoothness in the behavior, in contrast to SE_FF, where there are some big variations till the middle of run. In Figure 5.4, we observe that SE_VB performs better than SE_FF with respect to maximum number of hops objective. Thus the inferiority of SE_VB in monetary cost and average delay is compensated, to some extent, by improvement in maximum number of hops.

We have seen that best fixed bias SE_FF has more execution time than SE_VB

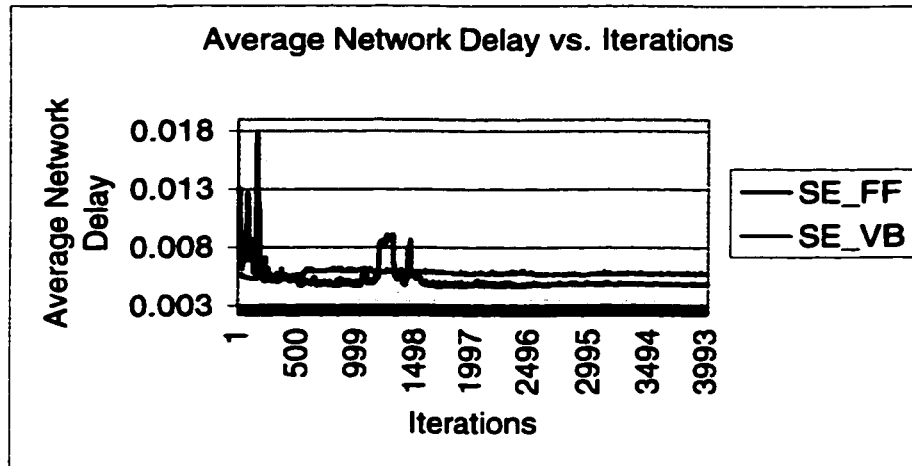


(a)

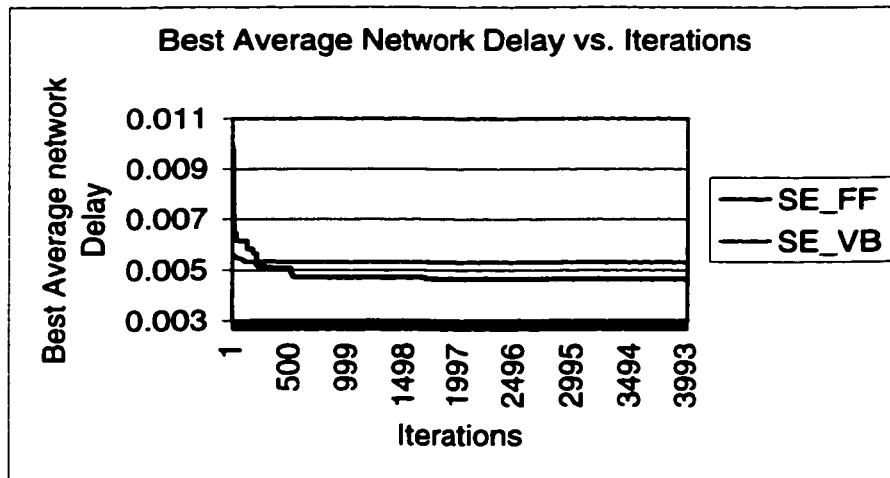


(b)

Figure 5.2: Comparison of SE_FF and SE_VB for n33 (a) current monetary cost (US \$) (b) best monetary cost.

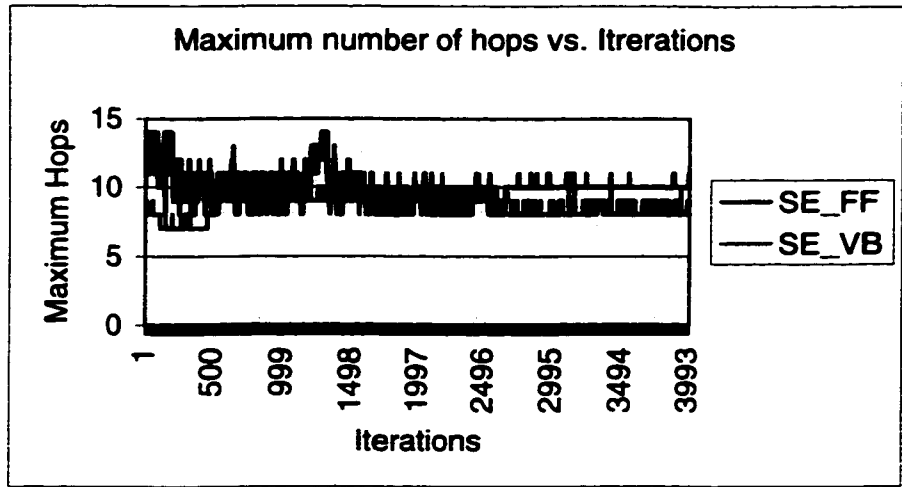


(a)

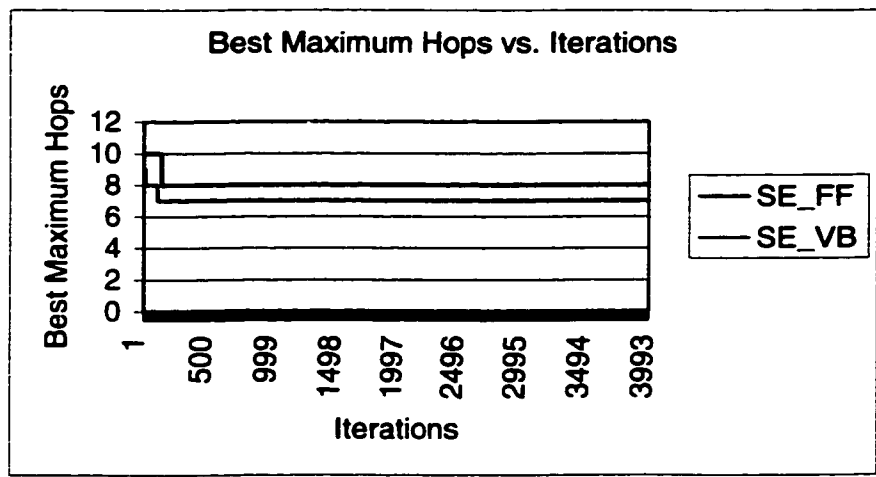


(b)

Figure 5.3: Comparison of SE_FF and SE_VB for n33 (a) current average delay (seconds) (b) best average delay.



(a)



(b)

Figure 5.4: Comparison of SE_FF and SE_VB for n33 (a) current maximum hops (b) best maximum hops.

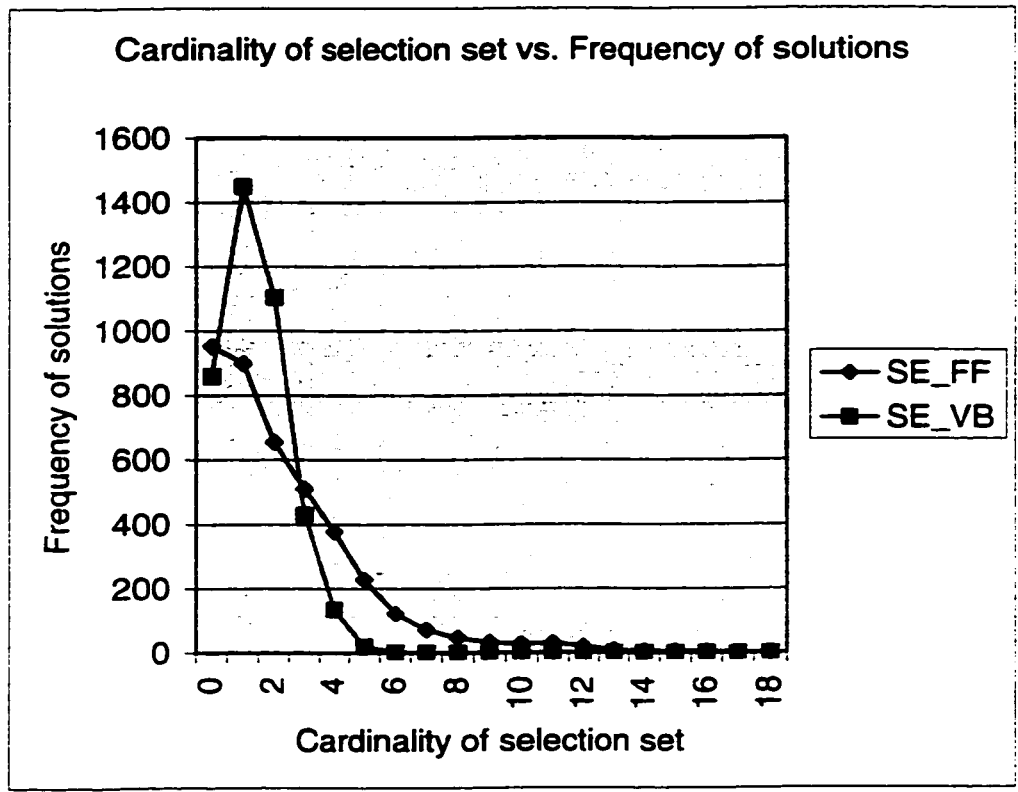
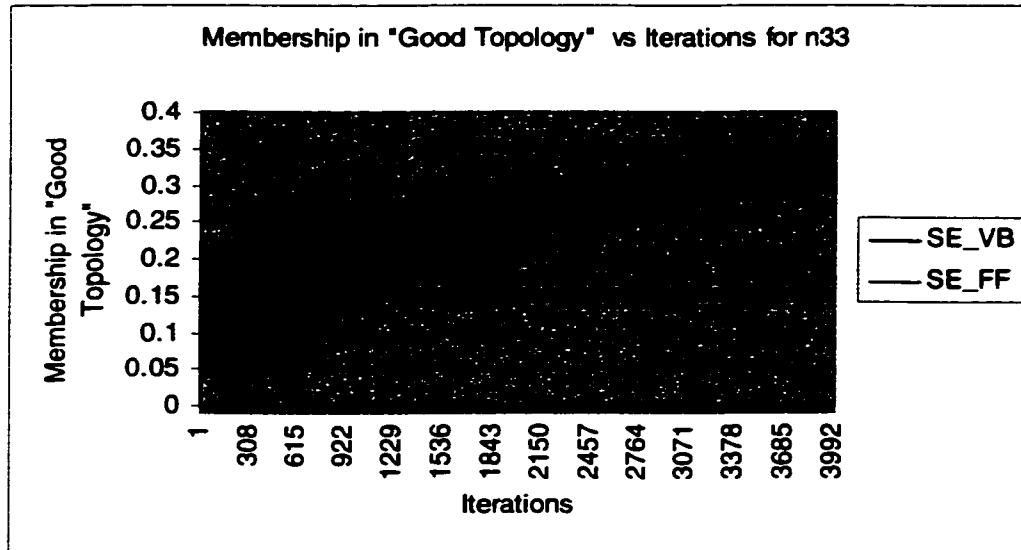
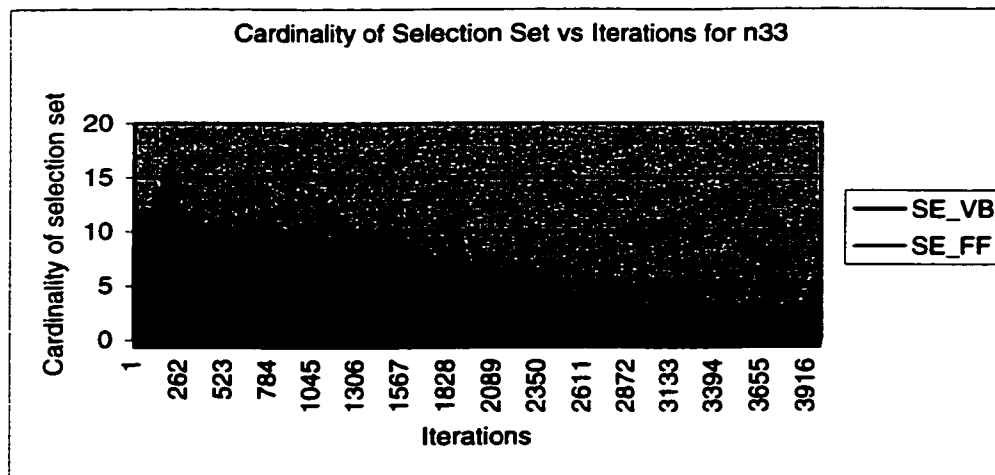


Figure 5.5: Cardinality of selection set against frequency of solutions for SE_FF and SE_VB for n33.



(a)



(b)

Figure 5.6: Comparison of SE_FF and SE_VB for n33 (a) Membership in "Good Topology";(b) Cardinality of selection set versus iterations.

for some test cases (e.g., **n33**). To see why this is happening, we plot the cardinality of the selection set against the frequency of solutions in Figure 5.5. From this figure, it is obvious that for SE_VB, the cardinality of selection set is in the range of 0 to 5 links. On the other hand, this range goes upto 18 links in case of SE_FF. This trend describes the reason for SE_FF having higher execution time, because more links are selected for perturbation, causing more time spent in different phases of the algorithm. The trend also describes the big variations in plots of SE_FF, also due to the high cardinality of selection set.

Figure 5.6(a) shows the membership in “Good Topology” versus iterations for **n33** for SE_FF and SE_VB. In this figure, it is seen that initially, SE_FF is inferior in solution quality than SE_VB, but towards the end converges to a better quality solution than SE_VB. However, in the same figure, bigger variations are observed with SE_FF than with SE_VB. The reason is that, as depicted in Figure 5.6(b), the number of links selected in SE_FF are more than those of SE_VB. This causes a bigger perturbation in the solution for SE_FF.

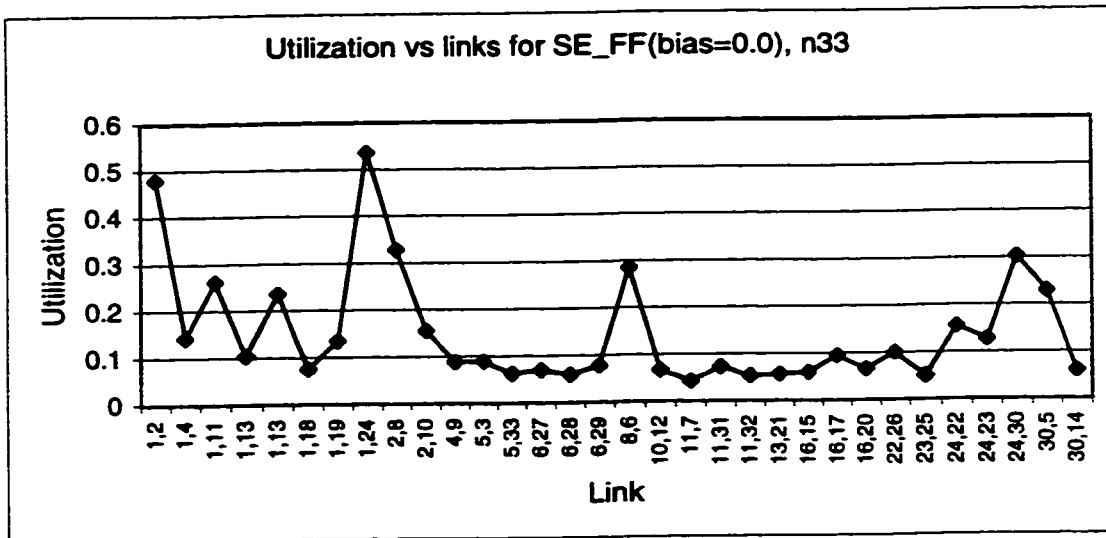
Figure 5.7 gives the links and their utilizations for best solution generated by best fixed bias SE_FF and SE_VB for **n33**. From the figure it is clear that traffic on any link is not exceeding the maximum allowed utilization, which is 60 Mbps, or 60 % (since we are using a 100 Mbps FO cable). It can also be seen that SE_FF generates a better topology than SE_VB since the former is availing all the 8 ports available on the backbone switch (denoted by 1), while the latter is only using 7 ports. Also,

there are more links in topology for SE_VB with high utilizations than in SE_FF. In this figure, all the links that have been used are shown with their respective utilizations. The figure also defines the topology of the best solution generated, with the hierarchy of the local sites. For example, in the case of SE_FF, the link 1,2 means that local site 1 (where the backbone switch is located) is connected to local site 2 and that local site 1 is preceding local site 2 (i.e., local site 1 is the parent of local site 2). Similarly local site 2 is the parent of local site 8. Also, local site 8 is the parent of local site 6 and so on. If this pattern is followed, the complete topology can be visualized.

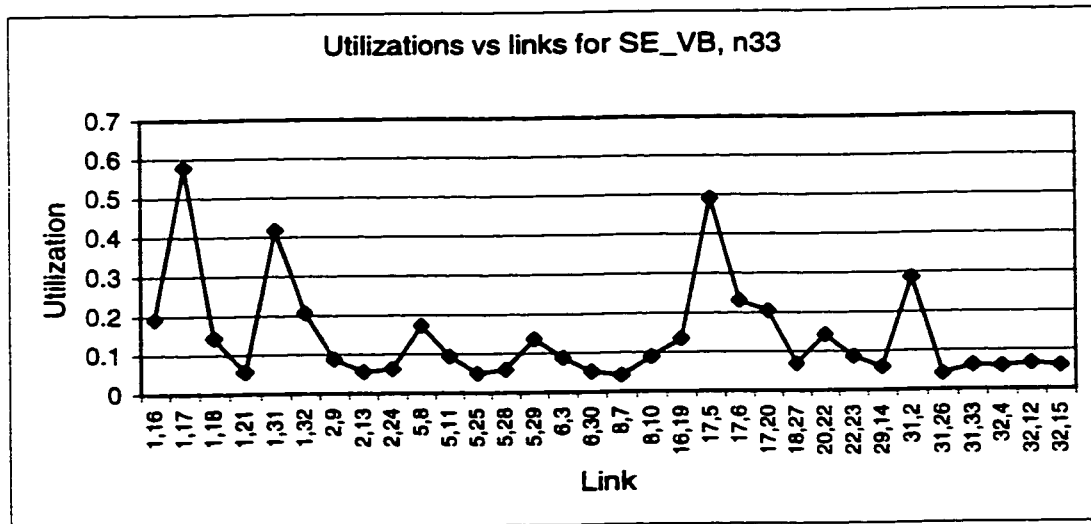
5.2.2 Effect of Tabu Search Based Allocation and Tabu List Size

We have seen in Section 5.2.1 that although SE_VB has lesser execution time than SE_FF, it is not able to reach the same quality of solution as the best fixed bias SE_FF. In this section, we propose a scheme in which Tabu Search characteristics are incorporated in the allocation phase of SE_VB. The purpose of this is to enhance the quality of search of SE_VB.

Tabu Search (TS) is a general iterative heuristic that is used for solving combinatorial optimization problems. The algorithm was first presented by F. Glover [42][43]. A key feature of TS is that it imposes restrictions on the search process, preventing



(a)



(b)

Figure 5.7: Links and their utilizations for best solution generated for n33 by (a) SE_FF (b) SE_VB. Traffic is in Mbps.

it from moving in certain directions to drive the process through regions desired for investigation [42]. It searches for the best move in the neighborhood of the current solution. As opposed to local search, TS does not get trapped in local optima because it accepts bad moves if they are expected to lead to unvisited solutions [43].

In its very basic operation, TS works as follows. It starts with an initial solution s that is selected randomly or using any constructive algorithm. It defines a subset $V^*(s)$ of its neighborhood $N(s)$. The algorithm evaluates the solution in $V^*(s)$ and finds the best (in terms of evaluation function) among them, call it s^* to be considered as the next solution. The algorithm maintains a list of *attributes* of accepted moves for the purpose of not cycling back to recently visited solutions. This list is called *Tabu List*. The size of tabu list specifies the number of coming iterations for which the move remains tabu. An *attribute* is some characteristic of a move which is saved in the tabu list, since it is not feasible to store the whole solution when the solution representation is large or complex. If the tabu list does not define the move leading to s^* as tabu, it is accepted as the new solution even if it is worse than the current solution in terms of the evaluation function. However, if the move leading to s^* is defined as tabu by the tabu list, the solution is not accepted until it has one or more features that makes the algorithm override its tabu status to accept it. *Aspiration criterion* is used to check whether the tabu solution is to be accepted or not. For more details, interested readers are referred to [42, 43, 7].

In this research work, we have modified SE_VB algorithm by introducing tabu

search characteristics in the allocation phase. We call this SE_TS algorithm. Recall that in the allocation phase (Section 4.2.4), certain number of moves are made for each link in the selection set and the best move is accepted, making the move (i.e., link) permanent. This newly accepted link is also saved in the tabu list. Thus our attribute is the link itself. The aspiration criterion is that if a link that had been made tabu produces a higher membership value than the current one in the membership function “good topology”, then we will override the tabu status of the link and make it permanent.

Table 5.7 shows the results obtained for the test cases using different tabu list sizes. In this table, monetary cost, average delay, and maximum hops of best solutions are reported along with the respective tabu list size. In the table we notice that as the test case size increases, the tabu list that gives the best solution also increases. For example, in **n15**, tabu list size of 2 gives the best solution. Similarly, best solutions are achieved by tabu list sizes of 5, 6, 7, and 7 in **n25**, **n33**, **n40**, and **n50** respectively.

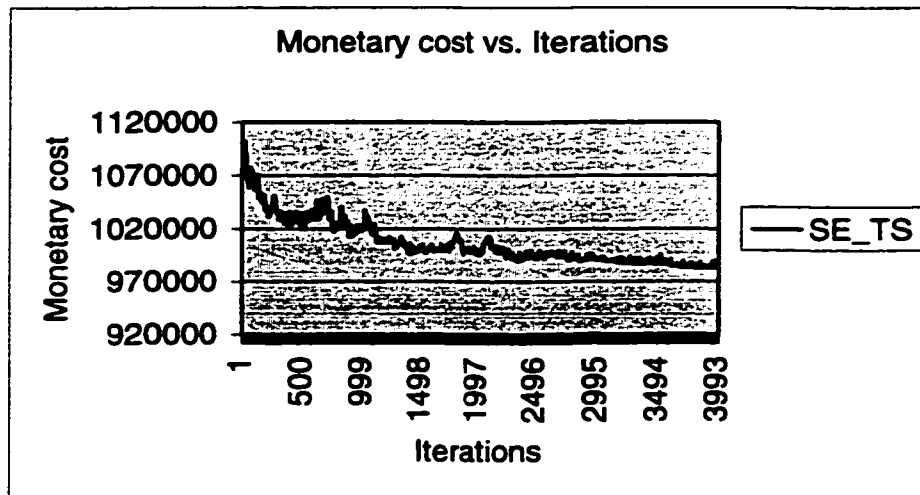
In order to see the behavior of our SE_TS algorithm, we plot different cost parameters versus iteration count for the test case **n50** (tabu list size=7). Figure 5.8 shows the monetary cost of solutions found after every iteration. The plots suggest that the algorithm converges smoothly without making big jumps. A similar behavior is seen in Figures 5.9 and 5.10 which plot the average network delay and maximum number of hops respectively. The stability of the algorithm becomes more

Test Case	Tabu list size	Monetary Cost	Avg. Delay	Max. Hops
n15	1	298200	2.935	4
	2	297100	2.78	4
	3	294350	3.448	6
	4	298100	3.037	5
	5	296900	3.278	6
n25	3	481745	4.219	8
	4	478690	4.189	9
	5	483210	3.537	6
	6	479915	4.275	9
	7	488400	4.608	9
n33	3	655715	5.772	11
	5	652785	4.77	8
	6	682465	4.19	6
	7	652310	5.95	10
	9	667100	5.087	7
n40	5	785795	4.746	10
	6	798695	8.019	12
	7	783970	4.441	9
	8	786950	5.478	9
	9	790645	5.136	8
n50	4	958995	6.739	14
	5	967110	9.279	14
	7	983020	5.245	11
	8	1075450	5.725	9
	9	971965	7.13	12

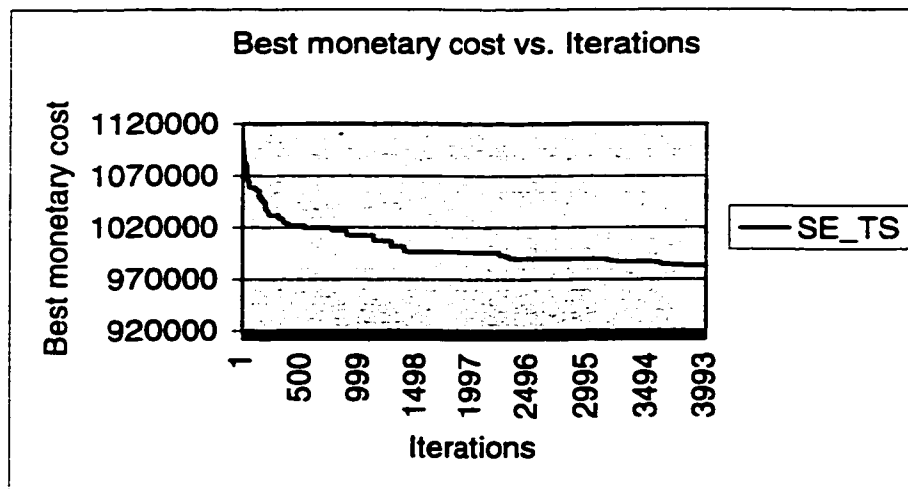
Table 5.7: Best solution for different tabu list sizes. Monetary cost is in US \$, delay is in milli seconds per packet, and execution time is in minutes.

obvious in Figure 5.11 where the membership in fuzzy function “good topology” is plotted against iterations. The figure shows small moves with constant convergence towards the optimum. Figure 5.12 shows the frequency of solutions having membership values in different ranges. According to this figure, out of 4000 solutions (since we run the algorithm for 4000 iterations), most of the solutions are falling in higher membership ranges, suggesting that the algorithm is concentrating in good solutions subspace. The convergence of the algorithm towards better solution is dependent on the average goodness of links. A high average goodness of links suggests that better links have been found and that there is a little room for further improvement. This is seen from Figure 5.13, where we see that the average goodness of links is increasing with each iteration. This improvement slows down towards the end, suggesting that better links have been found throughout the topology. To further strengthen our claim, we plot the minimum and maximum goodness of a link in each iteration in Figure 5.14, where the minimum goodness of a link gradually becomes nearer to average goodness of links, while the maximum goodness of link almost reaches the “optimum” value of 1. Figure 5.15 shows the solution generated by SE_TS with links and their utilizations for **n50**. In this figure we observe that the traffic is not exceeding the allowed peak utilization (60% or 60 Mbps) on any link and all the 8 ports on the backbone switch have been used.

Table 5.8 gives the results for different test cases considering the frequency of tabu moves, and the respective tabu list size that gave the best solutions with their

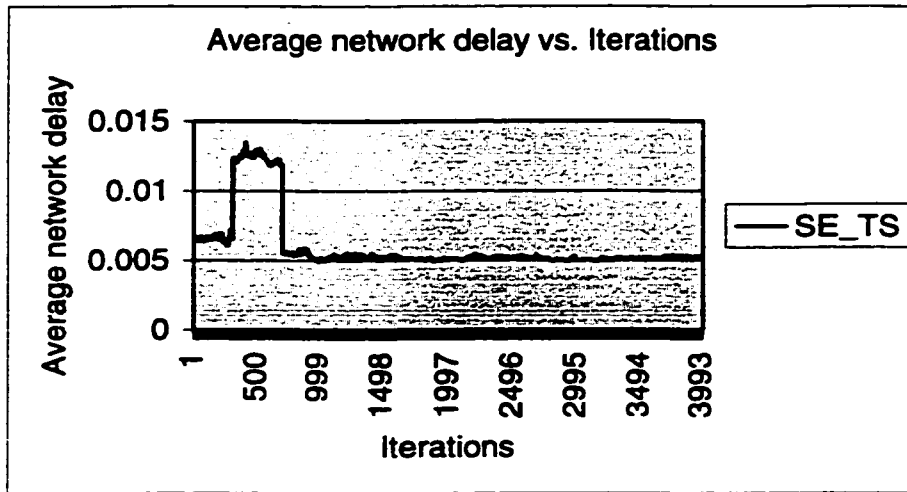


(a)

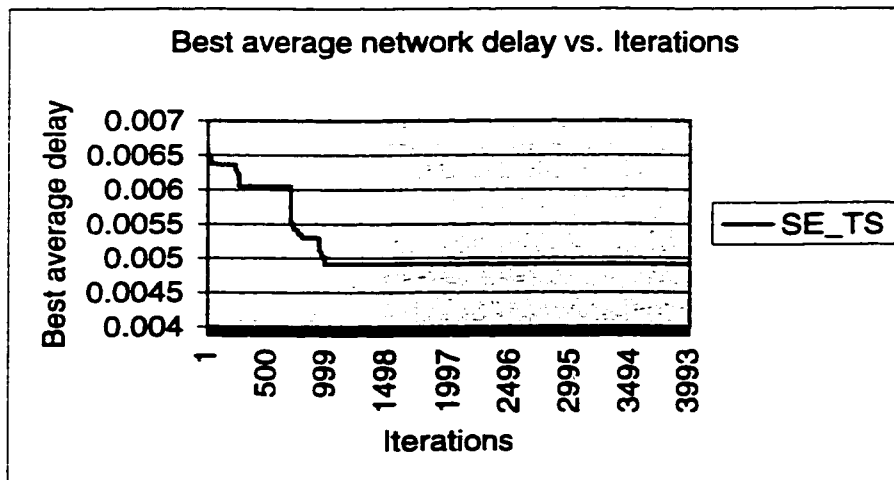


(b)

Figure 5.8: Variation of monetary cost with iterations for $n=50$, Tabu list size=7
 (a) current monetary cost (US \$); (b) best monetary cost.

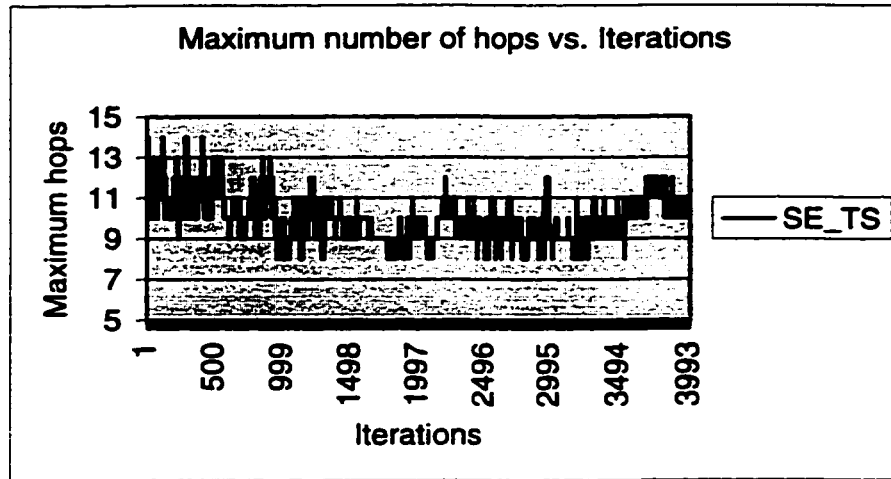


(a)

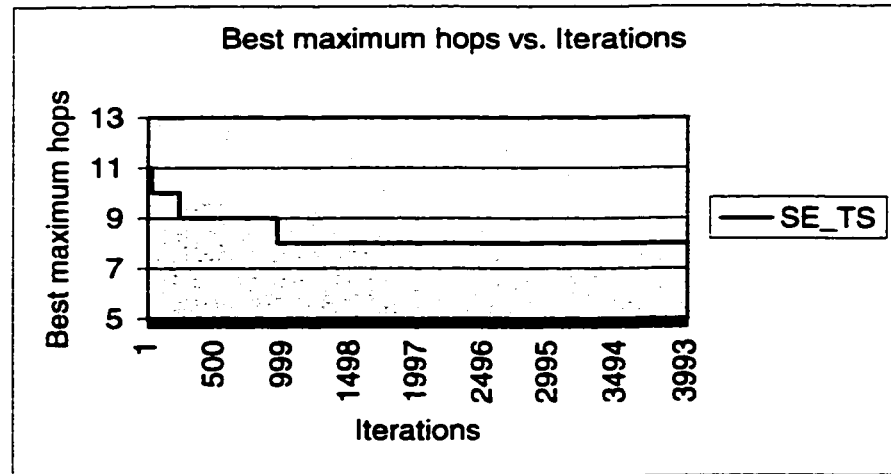


(b)

Figure 5.9: Variation of average delay with iterations for $n=50$, Tabu list size=7
 (a) current average delay (seconds); (b) best average delay.

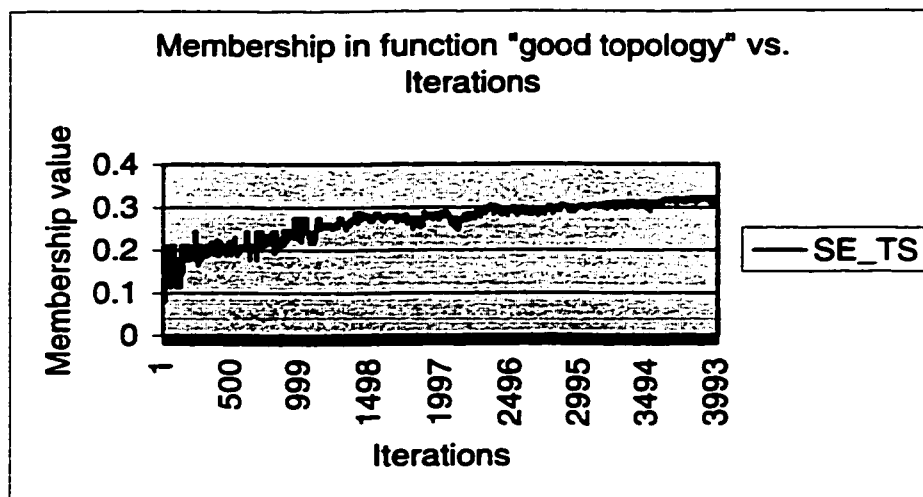


(a)

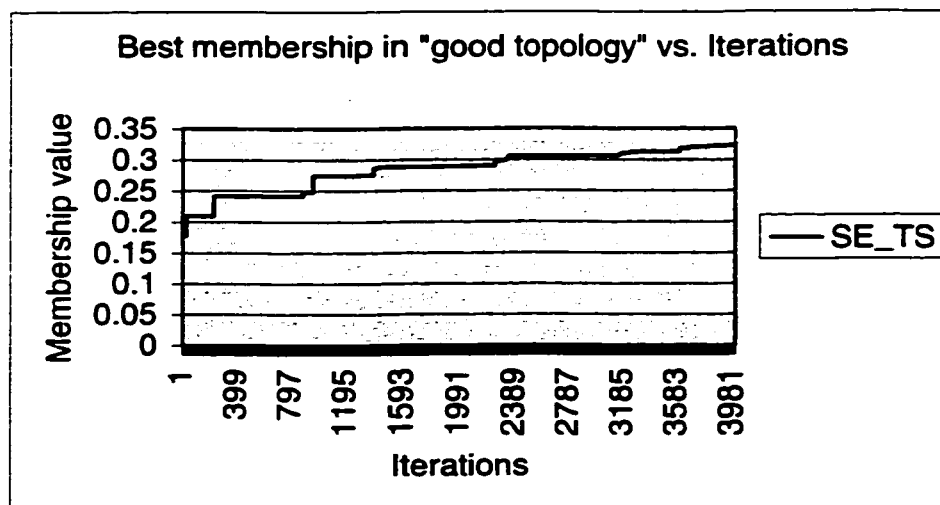


(b)

Figure 5.10: Variation of maximum hops with iterations for $n=50$, Tabu list size=7
 (a) current maximum hops; (b) best maximum hops.



(a)



(b)

Figure 5.11: Membership in function "good topology" for $n=50$, Tabu list size=7
 (a) current membership; (b) best membership.

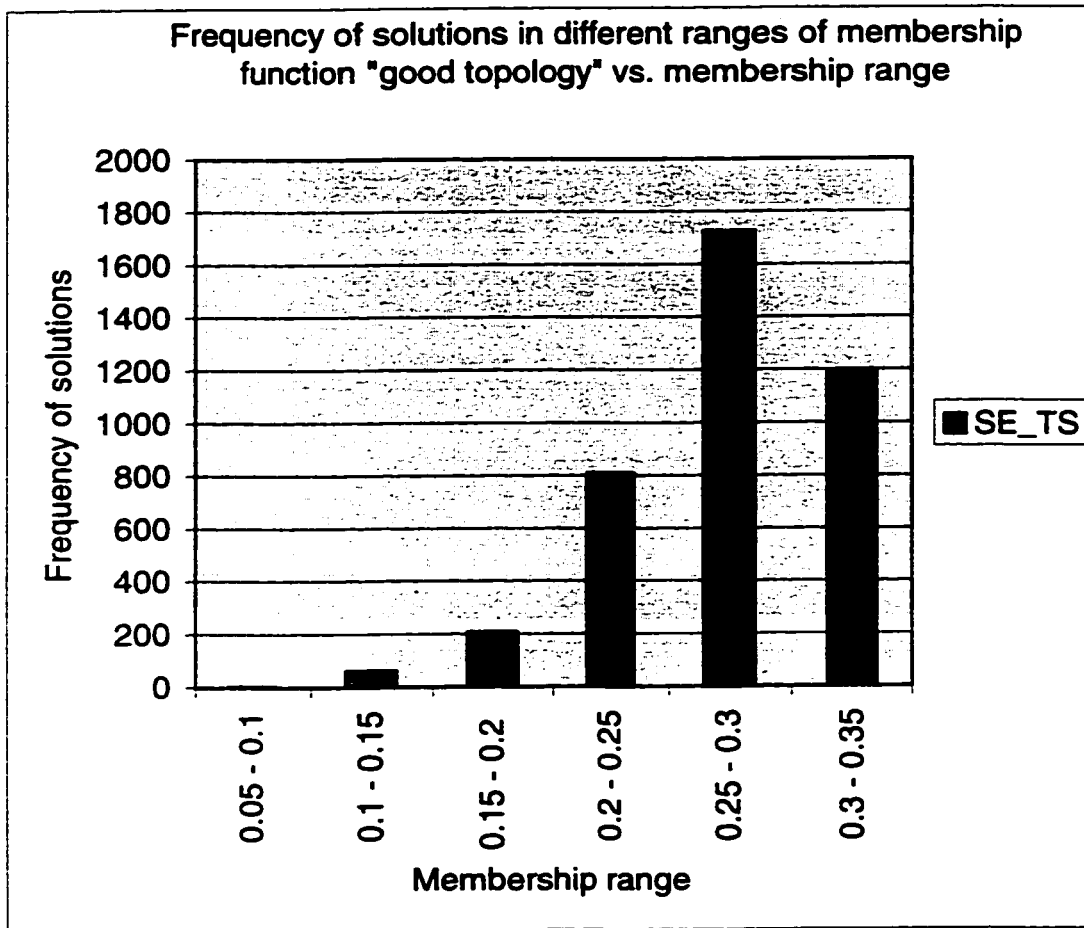


Figure 5.12: Frequency of solutions falling in different ranges of function "good topology" for $n=50$, Tabu list size=7.

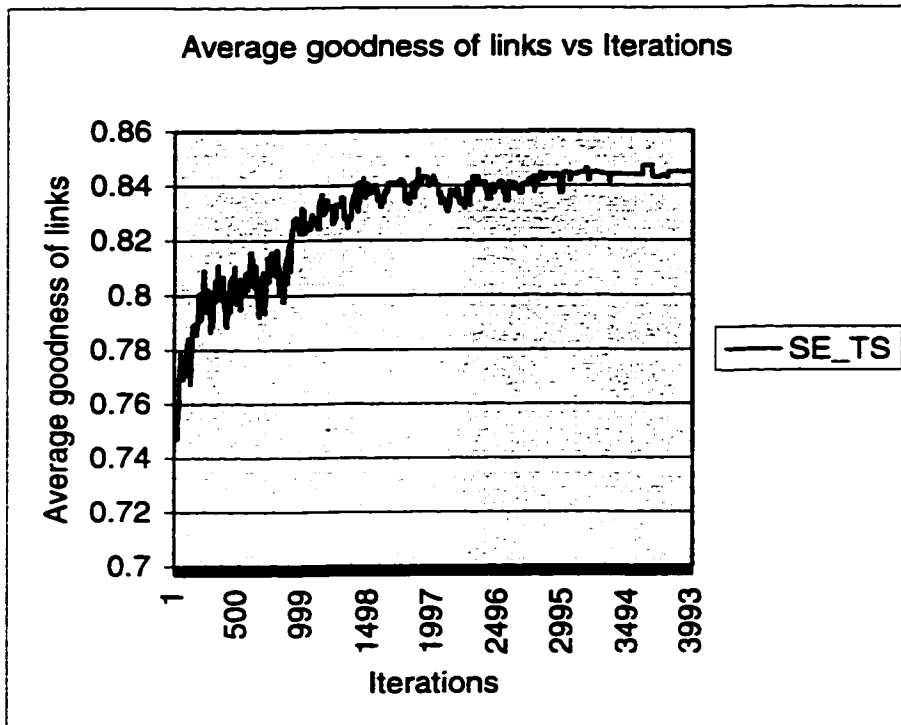
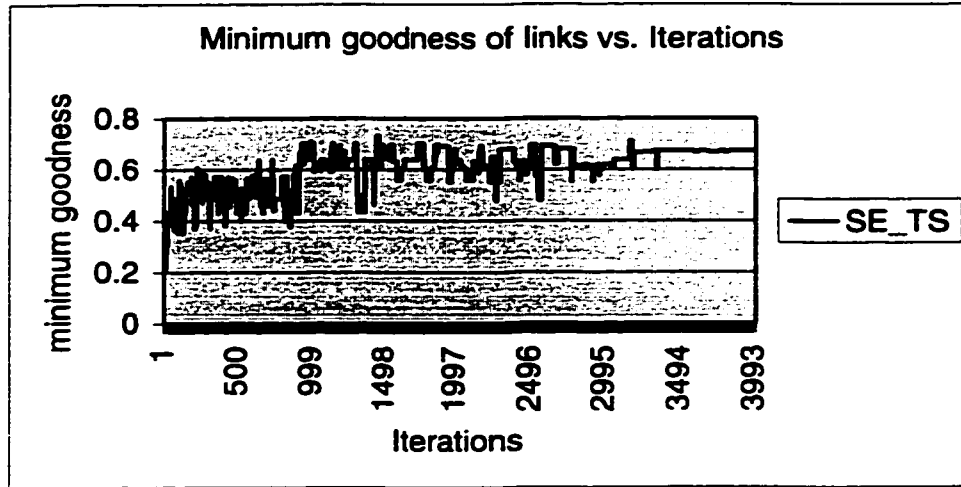
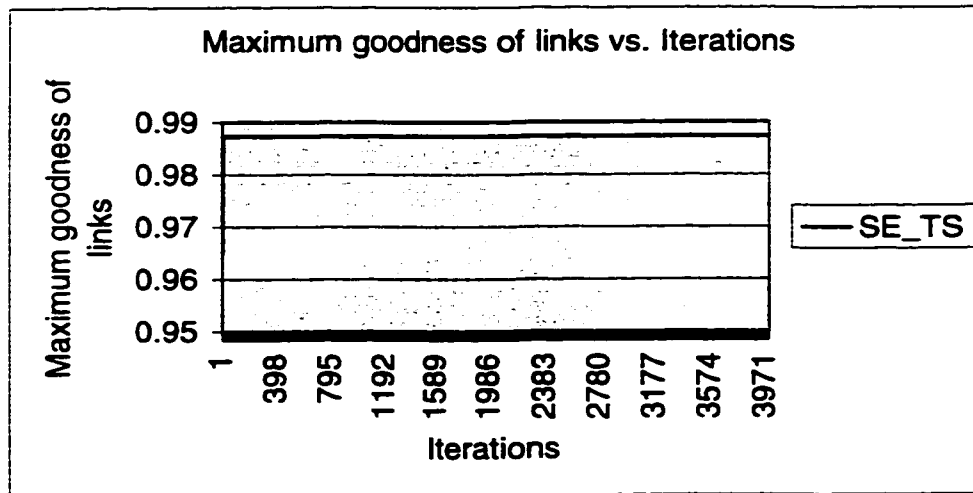


Figure 5.13: Variation of average goodness of links with iterations for n50, Tabu list size=7.



(a)



(b)

Figure 5.14: Variation of minimum and maximum goodness of a link with iterations for $n=50$, Tabu list size=7; (a) minimum goodness of a link (b) maximum goodness of a link.

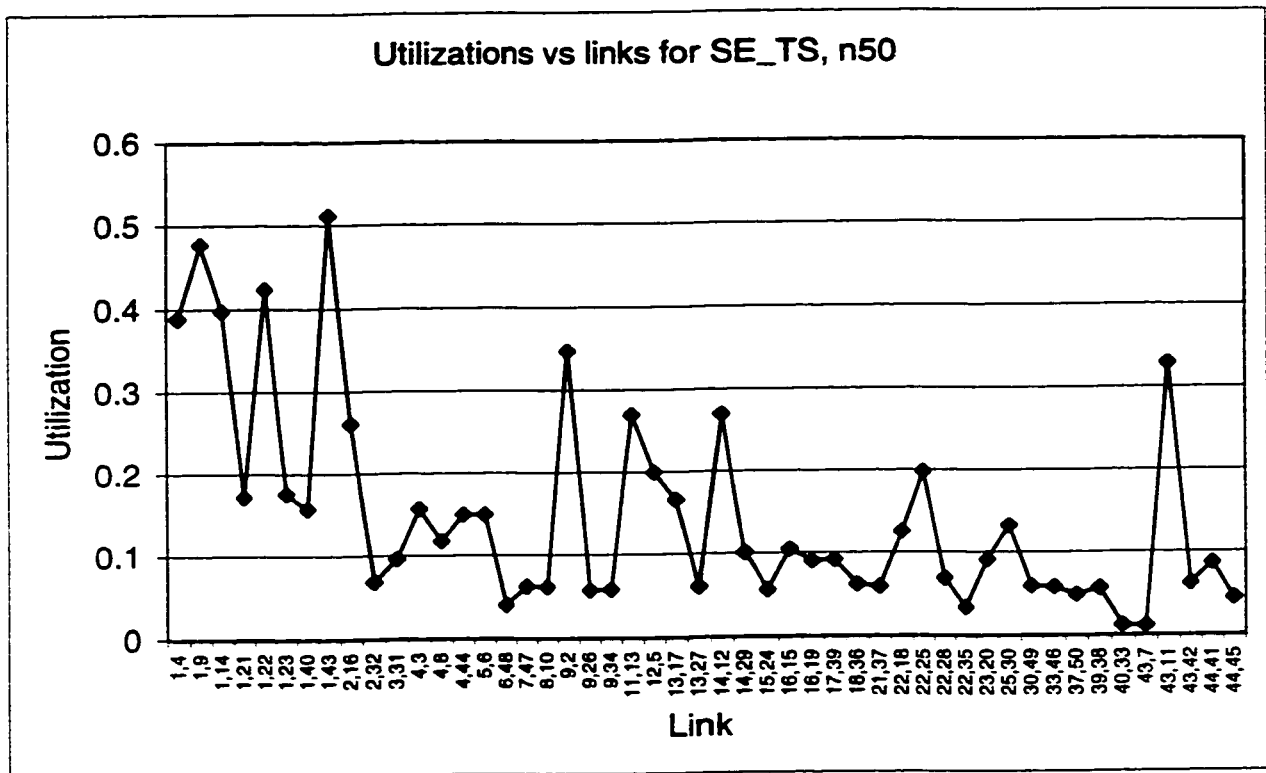


Figure 5.15: Links and their utilizations for best solution generated by SE_TS for test case n50. Traffic is in Mbps.

execution times. By frequency of tabu moves we mean the number of times a link was found tabu. We record this through a counter called *tabu counter*. The tabu counter only includes the number of tabu links which could not pass the aspiration criteria. It does not count the frequency of links which were actually tabu but managed to pass the aspiration criteria. From this figure, it can be seen that the percentage of tabu moves varies between 1% and 10%.

Test case	Tabu list size for best solution	Total moves	Tabu moves	% Tabu moves	Exec. time
n15	2	1241	45	3.62	2.25
n25	5	2496	39	1.56	4
n33	6	1352	93	6.878	8
n40	7	4223	233	5.51	26
n50	7	3995	328	8.21	65

Table 5.8: Results for best tabu list size. Execution time is in minutes.

5.2.3 Comparison of SE_FF and SE_TS

In Section 5.2.1, best fixed bias SE_FF and SE_VB were compared and it was found that overall, SE_FF produced better results than SE_VB. However, as mentioned earlier, one problem with fixed bias approach is that it needs many runs to find out the best fixed bias, which is not favorable in terms of time. In Section 5.2.2, the effect of introducing tabu search characteristics in variable bias based SE algorithm, the SE_TS algorithm, was studied. In this section, the results of SE_TS with best fixed bias SE_FF are compared to evaluate the performance of the former with respect

to the latter. Tables 5.9 and 5.10 show the results for best fixed bias SE_FF and best tabu list size SE_TS. These results have been reproduced here for the sake of comparison. The percentage gain shows the improvement achieved by SE_TS when compared to SE_FF. From these tables it is seen that SE_TS performs better than SE_FF as far as monetary cost objective is concerned. In all the test cases, a gain is achieved by SE_TS. For example, a gain of 7.43 % is achieved in case of **n50**. A similar behavior is seen for average network delay metric, where SE_TS achieves the gain in all the cases, with the exception of **n40**, where a loss of 7.63 % is observed. Similarly, for maximum number of hops metric, a gain is achieved for small (**n15** and **n25**) and medium (**n33**) size test cases, with a loss in performance for **n40** and **n50**. However, the loss in maximum hops and average delay metric for **n40** is compensated by the improvement in the monetary cost metric. Also, the loss for **n50** in maximum hops metric is compensated by improvement in monetary cost and average delay metric. However, if we compare the membership values in fuzzy function "good topology" for SE_FF and SE_TS, we see that in all the test cases, SE_TS converges to a better overall solution than SE_FF. As far as the execution time is concerned, a similar behavior is seen as was observed for comparison of SE_FF and SE_VB, where a gain is achieved for some test cases as well as the losses. However, it can still be said that SE_TS is better than SE_FF with respect to execution time if we take into account the time spent in trial runs to find the best bias in SE_FF. It is also seen that SE_TS performed better than SE_VB, if the

Case	SE_FF						SE_TS					
	B	C	D	H	M	T	TL	C	D	H	M	T
n15	0.2	314400	3.282	5	0.25	4	2	297100	2.78	4	0.318251	2.25
n25	0.2	509050	4.26	7	0.26	5	5	483210	3.537	6	0.352056	4
n33	0.0	687760	4.729	8	0.339005	40	6	682465	4.19	6	0.352941	8
n40	0.3	866900	4.126	8	0.2714	12	7	783970	4.441	9	0.337554	26
n50	0.3	1061900	5.32	9	0.264286	8	7	983020	5.245	11	0.32212	65

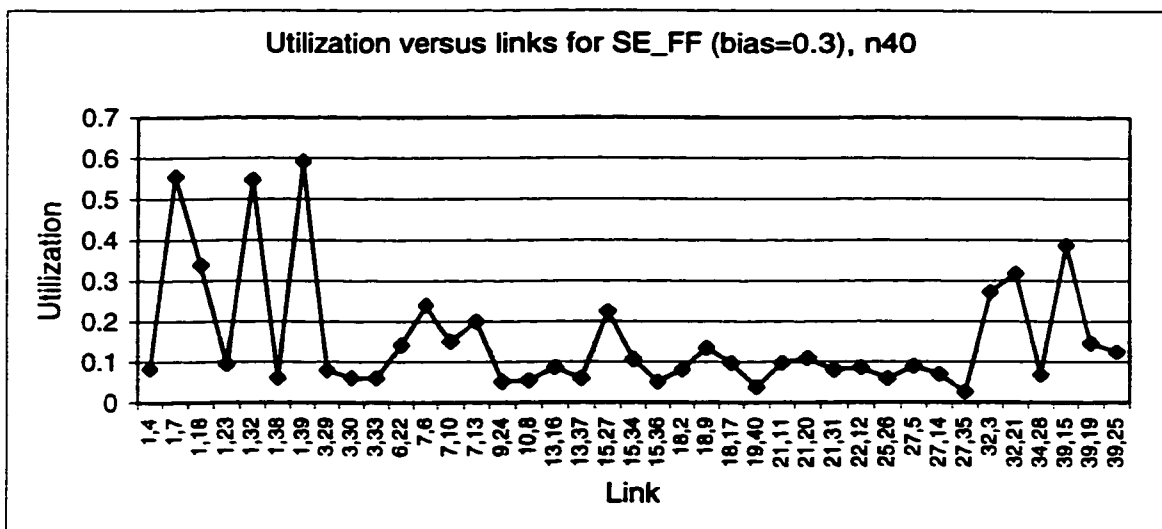
Table 5.9: Comparison of SE_FF and SE_TS. B = best bias, C = Cost in US \$, D = Delay in milli seconds per packet, H = hops, M = Membership in "Good Topology", T = execution time in minutes, TL= Tabu list size.

Case	% Gain		
	C	D	H
n15	5.5	15.29	20
n25	5.07	16.97	14.29
n33	0.755	11.4	25
n40	9.57	-7.63	-12.5
n50	7.43	1.41	-22.2

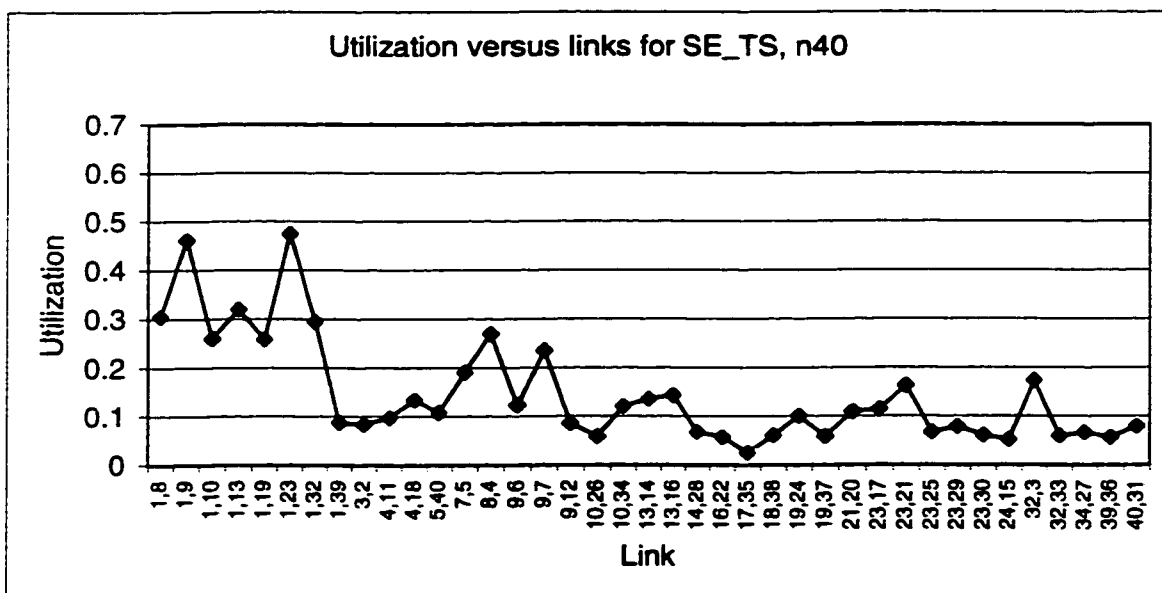
Table 5.10: Percentage gain in improvement achieved by SE_TS compared to SE_FF. C = Cost in US \$, D = Delay in milli seconds per packet, H = hops.

results of Tables 5.6, 5.9, and 5.10 are compared. Figure 5.16 gives the selected links and their traffics for the best solution for SE_FF and SE_TS for n40.

To compare the quality of search space between best fixed bias SE_FF and SE_TS, we plot different cost parameters versus iteration count of the algorithms for the test case n40 (best bias=0.3 in SE_FF and best tabu list size=7 in SE_TS). Figure 5.17(a) and (b) respectively give the current and best monetary costs. From these plots it is clear that SE_TS converges faster towards a better solution. On the other hand, SE_FF stops improving the cost after around 2000 iterations. Same



(a)

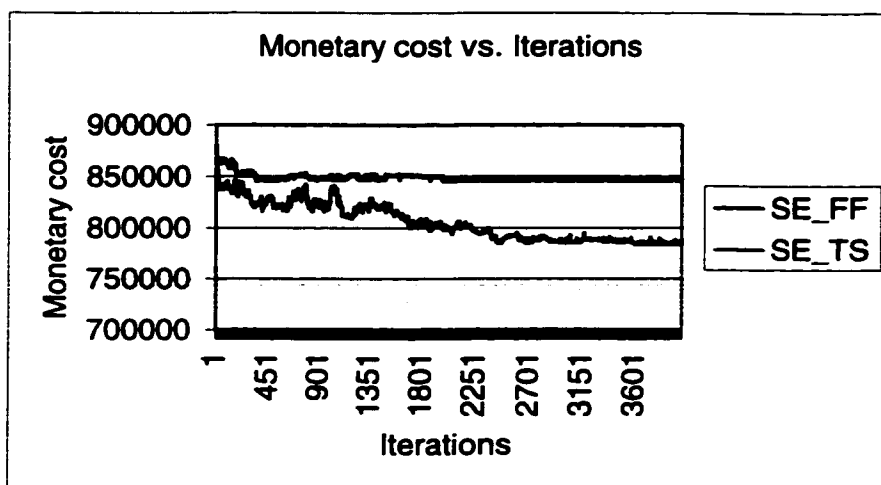


(b)

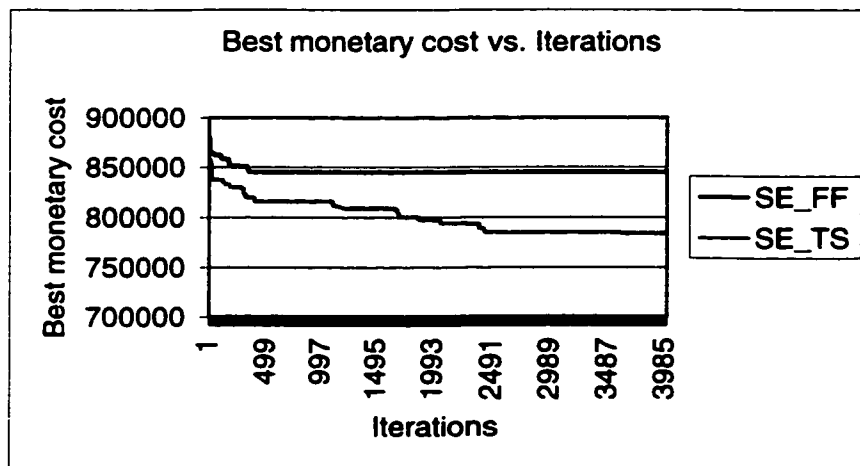
Figure 5.16: Links and their utilizations for best solution generated for n40 by (a) SE_FF (b) SE_TS. Traffic is in Mbps.

behavior is seen with respect to average network delay and maximum hops parameters in Figures 5.18 and 5.19, where we see that SE_TS performs better than SE_FF throughout the run. The reason SE_TS has better performance than SE_FF is the following. In SE_FF, since the search space for valid solutions is limited, it happens that after some iterations, same moves are repeated and thus the algorithm keeps searching in the same search space most of the time, while in SE_TS, although the search space is same, more search space is covered because previous moves remain tabu for some time, causing the algorithm to diversify the search into another sub-area. Recall that in the allocation phase, four valid moves are evaluated for each link in selection set and the best move (link) is made permanent. This new link is also saved in the tabu list simultaneously. However, it may happen that this new link may become “bad” (in terms of evaluation function) in the following iterations, upon which it is removed. But it may be possible that this link may become good again after one or more iterations, but since it is in the tabu list, it will not be chosen to be placed again, thus giving room to other links to be chosen. This increases the search space for SE_TS.

In general we see that SE_TS achieves the same quality of solution as best fixed bias SE_FF with lesser execution time than SE_FF. The reason for this is that SE_TS is based on SE_VB, which visits respectively lesser search space than SE_FF, but by introducing the tabu search characteristics in SE_TS, we explore a larger search space than SE_VB.

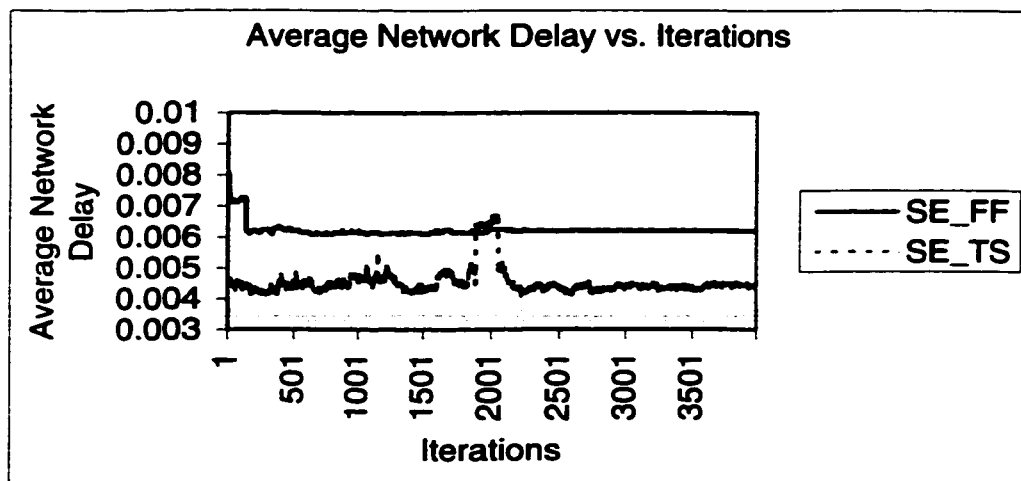


(a)

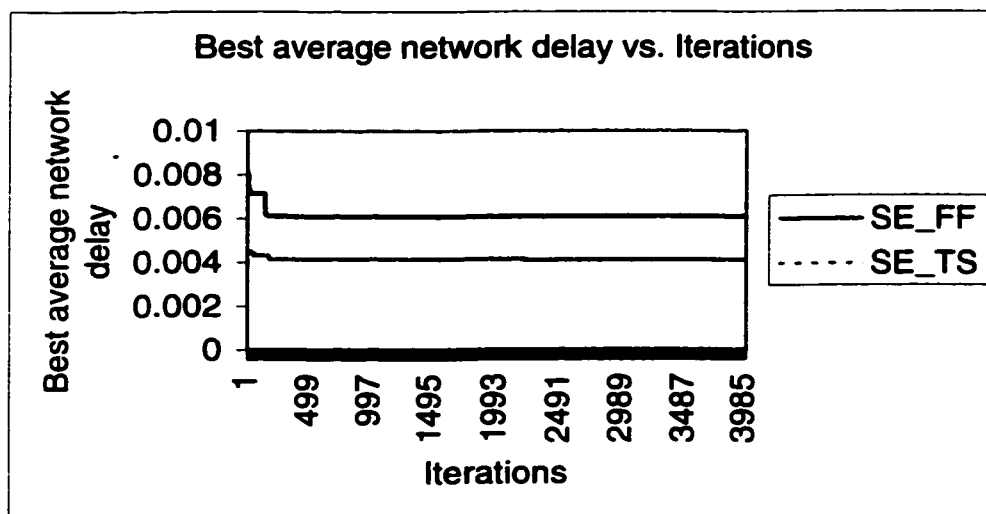


(b)

Figure 5.17: Comparison of SE_FF and SE_TS for n40 (a) current monetary cost (US \$)(b) best monetary cost.

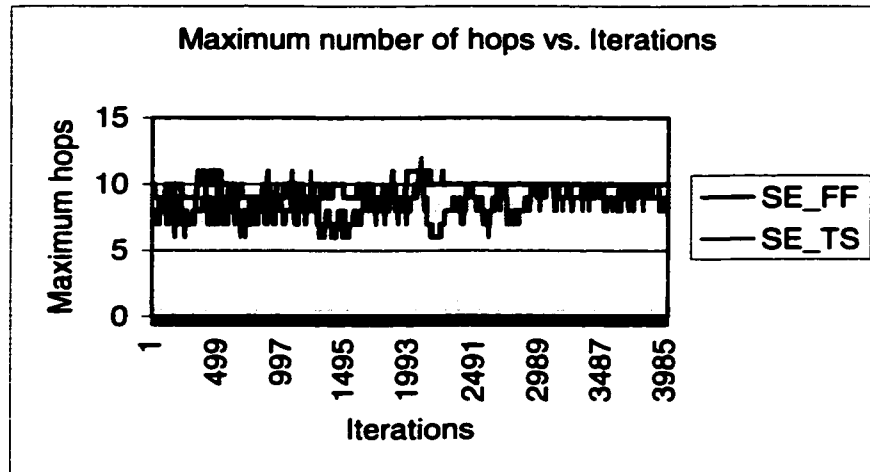


(a)

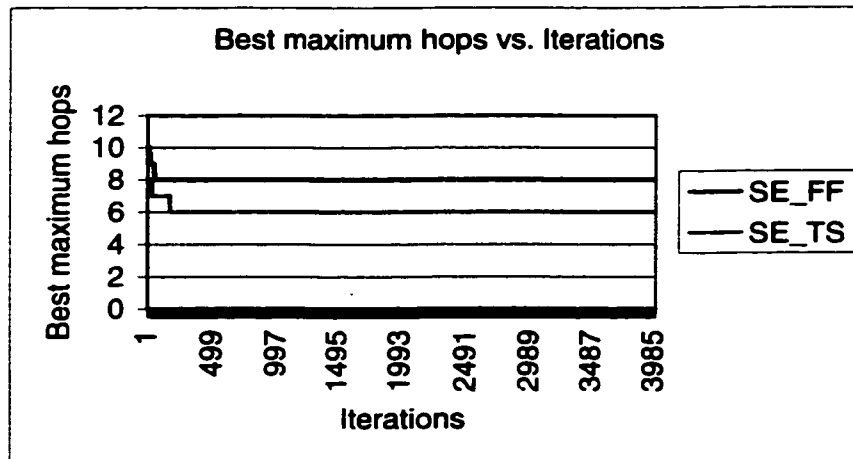


(b)

Figure 5.18: Comparison of SE_FF and SE_TS for n40 (a) current average delay (seconds) (b) best average delay.



(a)



(b)

Figure 5.19: Comparison of SE_FF and SE_TS for $n=40$ (a) current maximum hops (b) best maximum hops.

5.2.4 Comparison of SE_TS and Esau-Williams algorithm

In Section 2.1, we discussed Esau-Williams (EW) algorithm, which is the most widely used algorithm for centralized network design. The original algorithm had the flow of traffic on the links as the design constraint with the objective of optimizing link cost. In order to analyze the performance of the suggested SE_TS algorithm, we compare it with the EW algorithm. Since there are more constraints in our case than the original Esau-Williams algorithm, we have adapted it according to the constraints considered in this research. Table 5.11 presents the results for EW algorithm and best tabu list size SE_TS. From this table it is quite clear that SE_TS has far better overall performance than EW algorithm. We observe that although SE_TS has a little inferior performance than EW with respect to monetary cost, yet if we consider the other two objectives, we see a significant improvement by SE_TS. The inferiority in monetary cost and superiority in average delay and maximum hops of SE_TS is due to the fact that EW algorithm is considering only the monetary cost as the objective to be optimized, therefore it reduces only the monetary cost, ignoring the other two objectives. We also see that although EW algorithm is optimizing only the monetary cost, it does not have a very significant improvement than SE_TS with respect to monetary cost objective. On the other hand, SE_TS has a significant gain in delay and hops objectives. Furthermore, EW algorithm has a local partial view of the search and always makes a move greedily, while SE_TS works always with

Case	EW				SE_TS					% Gain		
	C	D	H	T	TL	C	D	H	T	C	D	H
n15	292700	5.0012	9	0.017	2	297100	2.78	4	2.25	-1.48	44.4	55.56
n25	469790	5.002	9	0.033	5	483210	3.537	6	4	-2.78	29.29	33.3
n33	624180	5.642	10	0.033	6	682465	4.19	6	8	-8.54	25.74	40
n40	754445	13.491	14	0.033	7	783970	4.441	9	26	-3.77	67.1	35.71
n50	928105	7.167	14	0.083	7	983020	5.245	11	65	-5.58	26.82	21.42

Table 5.11: Comparison of EW and SE_TS. C = Cost in US \$, D = Delay in milli seconds per packet, H = hops, T = execution time in minutes, TL= Tabu list size. The percnetage gain shows the improvement achieved by SE_TS compared to EW.

complete solution and accepts bad moves to get out of local minima. The difference in execution times of the two algorithms are due to the reason that EW algorithm is a constructive heusristic, while SE_TS is an iterative heuristic.

Time Analysis of Different Phases of SE_TS

We also studied the time spent in each phase of SE_TS algorithm, namely, initial solution, evaluation and selection, and allocation. Table 5.12 shows the results for n50 and n33 for best tabu list size. In this table we see that the allocation phase is taking a significant percentage of the total runtime. The reason is obviously that it is the most complex phase of the algorithm, where most of the manipulation of the solution takes place. It is also due to the fact that we are having very tight constraints due to which a larger number of invalid moves take place. Thus to have valid moves, more time is spent in each iteration. Also, the feasibility constraints are checked after each move is made. All these factors make the allocation phase a

very complex one.

Test case	% Init. Soln.	% Eval.+ Select.	% Alloc.
n33	0.208	14.472	85.32
n50	6.15	11.75	82.1

Table 5.12: Time analysis for different phases of SE_TS algorithm.

5.2.5 Comparison of Simulated Annealing and SE_TS

In Chapter 2, Simulated Annealing [7, 8] (SA) algorithm was discussed as applied to problems related to topology design of networks. In this section, we compare SA with SE_TS. As known, SA has four important parameters which need to be tuned very carefully. These are: cooling rate α , constant β , initial temperature T_0 , and M which represents the time until next parameter update. After trial runs, appropriate values of these parameters were found to be $\alpha=0.9$, $\beta=1.0$, $T_0=10$, and $M=10$.

Table 5.13 presents the results for SA and SE_TS. From this table, it is observed that SE_TS performs better than SA as far as monetary cost objective is concerned. In all the test cases, a gain is achieved by SE_TS. For example, a gain of 12.6 % is achieved in case of **n50**. A similar behavior is seen for average network delay metric, where SE_TS achieves gain in all the cases, e.g. in case of **n40**, where a gain of 14.81% is observed. Similarly, for maximum number of hops metric, a gain is achieved for small (**n15**) and medium (**n33**) size test cases, with a loss in performance for **n40** and **n50**. However, the loss in maximum hops for **n40** and

Case	SA				SE_TS					% Gain		
	C	D	H	T	TL	C	D	H	T	C	D	H
n15	318000	3.469	6	1.17	2	297100	2.78	4	2.25	6.57	19.86	33.3
n25	511320	3.725	6	3	5	483210	3.537	6	4	5.497	5.047	0
n33	708135	5.189	9	5.5	6	682465	4.19	6	8	2.76	19.25	33.3
n40	903735	5.213	8	27.5	7	783970	4.441	9	26	13.25	14.81	-11.1
n50	1124720	5.943	10	57	7	983020	5.245	11	65	12.6	11.74	-9.09

Table 5.13: Comparison of SA and SE_TS. C = Cost in US \$, D = Delay in milli seconds per packet, H = hops, T = execution time in minutes, TL= Tabu list size. The percentage gain shows the improvement achieved by SE_TS compared to SA.

n50 is compensated by the improvement in the monetary cost and average network delay metrics. As far as the execution time is concerned, SE_TS has a slightly higher execution time than SA in most of the cases. It is due to the fact that SA has only one move in an iteration while SE_TS has multiple moves in a single iteration.

In order to compare the quality of search space of SA and SE_TS, frequency of solutions for different membership ranges is plotted against the membership value in Figure 5.20 for n25. It is known that in SA, only one individual (link) is selected at a time, and only one move is allowed for that selected link. If the new link is not a feasible one or does not pass the Metropolis criterion, then the original link is placed back to its position. This implies that there may be iterations where original links are placed back and no alteration takes place to the currently existing solution. Therefore, such iterations are not included as giving a new solution for SA. This in turn gives a total of 2200 iterations (out of 4000 iterations) where an alteration took place in the solution. Thus, only these 2200 iterations (solutions) are plotted

in Figure 5.20. In this figure, it is observed that SE_TS has more solutions falling in higher membership ranges than SA, suggesting that SE_TS is concentrated in good solution subspace. For example, SA has most of the solutions in the membership range 0.2-0.25, whereas SE_TS has most of the solutions in the range 0.3-0.35.

Figure 5.21 plots the cumulative frequency of solutions for different membership ranges after each 200 iterations for **n25**. In Figure 5.21(a) shows the plot of SA, where we observe that there are no solutions in the membership ranges 0.3-0.35 and 0.35-0.4. The only ranges in this figure where solutions exist are 0.2-0.25 and 0.25-0.3. It is also seen that even in the membership range 0.2-0.25, solutions start to appear after around 350 iterations. This suggests that before 350 iterations, the solutions found have all membership values of less than 0.2. Another observation is that for the membership range 0.25-0.3, there are only a few solutions, which start to appear after 1000 iterations. The overall effect of Figure 5.21(a) is that although there is improvement in the quality of search space with the passage of iterations, yet it is mainly limited to a low membership range of 0.2-0.25.

Figure 5.21(b) shows the plot of the cumulative frequency of solutions for SE_TS for the same ranges as in Figure 5.21(a). In this figure, we notice that there are solutions in the membership ranges (such as 0.3-0.35) for which SA had no solution at all. One important trend here is that solutions with high membership range start to appear earlier than in SA. For example, we saw above that solutions in the range 0.2-0.25 start to appear after 350 iterations, but with SE_TS, solutions

in the same range appear just after 150 iterations. This behavior suggests that SE_TS starts converging toward better quality solutions faster than SA. Another important trend is that as the search progresses, SE_TS zooms on better solutions. For example, in Figure 5.21(b) we see that the cumulative frequency for membership range 0.2-0.25 increases upto 600 iterations, after which there is no solution in this range. However, after 400 iterations, solutions in the range 0.25-0.3 begin to appear and keep accumulating (with a slower pace) until around 1800 iterations. Similarly, solutions in the membership range 0.3-0.35 begin to appear after 700 iterations and keep accumulating with a high rate upto 2200 iterations, after which no additional solutions are found in this range. A point where the number of solutions are nearly equal in the above three membership ranges appears to be at 1100 iterations. There are also a few solutions (2 in fact) in the range 0.35-0.4 but they are not visible in this figure. The net effect of these graphs is that SE_TS is zooming on better quality solutions as the search progresses. Thus it is observed from Figure 5.21 that SE_TS achieves solutions with better quality than SA.

5.2.6 Quality of Final Solution with Different Initial Solutions

An experiment was run to see the quality of final solutions with different initial solutions for SE_TS. For this purpose, two initial solutions were used; one was

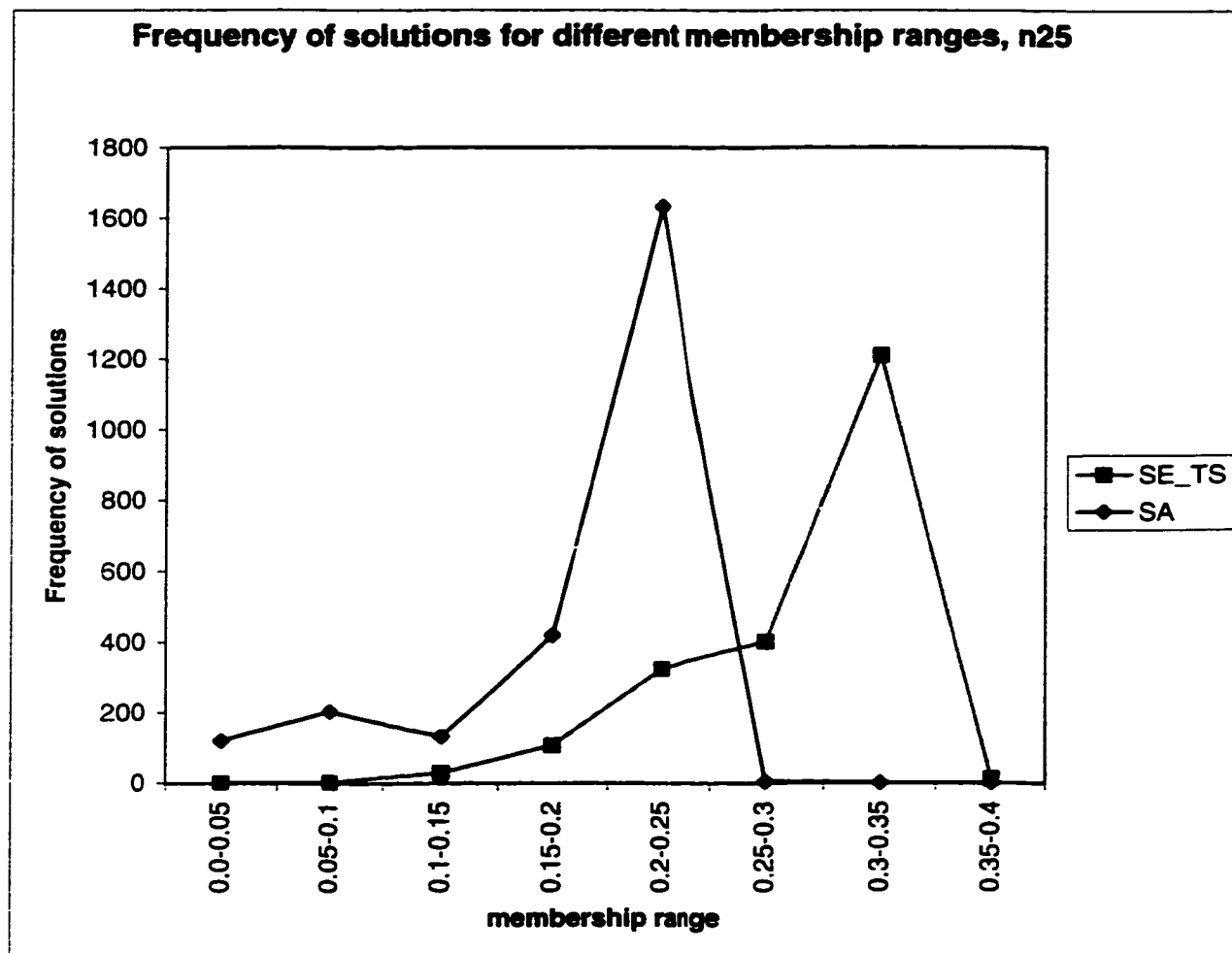
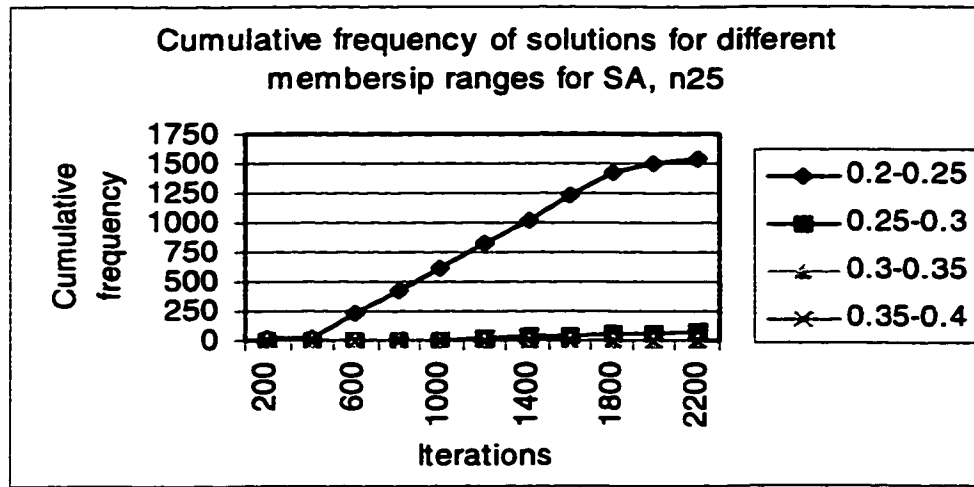
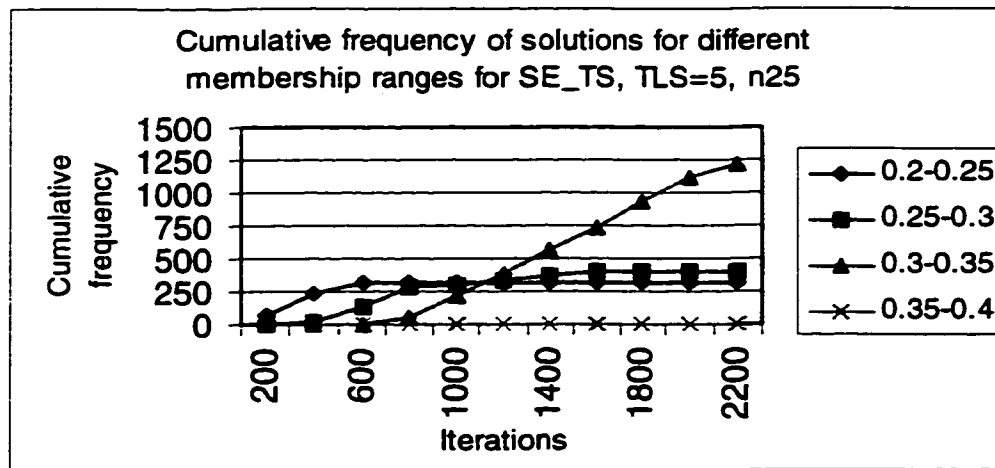


Figure 5.20: Frequency of solutions for different membership ranges for SA and SE_TS for n25.



(a)



(b)

Figure 5.21: Cumulative frequency of solutions for different membership ranges for n25 (a) SA (b) SE_TS.

Case	TL	EW			Random		
		C	D	H	C	D	H
n25	5	503715	3.377	6	483210	3.537	6
n33	6	654865	4.298	7	682465	4.19	6

Table 5.14: Quality of final solution with different initial solutions for SE-TS. TL= Tabu list size, C = Cost in US \$, D = Delay in milli seconds per packet, H = hops.

generated randomly and the other was as generated by the Esau-Williams algorithm.

Table 5.14 gives the final solutions for test cases **n25** and **n33**. It is clear from this table that the final solutions reached with both the initial solutions are of the same quality. This suggests that the quality of the final solution using SE-TS is not sensitive to the initial solution; the algorithm reaches the same quality of solution regardless of the quality of initial solution.

5.3 Assignment of LAN Segments to Network Devices using Augmenting Path Algorithm

In Section 2.1, we discussed the augmenting path (AP) algorithm. The algorithm was designed to solve the “terminal assignment problem” where the terminals (computers) were assigned to concentrators (networking devices). In this work, we have assumed that terminals have already been assigned to local concentrators, which make up a “segment” (see Section 3.2). The AP algorithm is also suitable to assign the segments (hubs) to upper level concentrators (workgroup switch), which when

interconnected will create a communication path between any two stations in the same or different work areas.

An experiment to show the AP algorithm works was conducted. Figure 5.22 depicts a typical local site. In this figure, the segments are denoted by S , and the concentrators are denoted by C . We see that there are fifteen segments and four concentrators located at four different places within the local site. Each concentrator (a hub or a switch in this case) has 4 ports. The task is to assign these fifteen segments to the four concentrators. When the AP algorithm is given the input, which is basically the distances of each segment from each concentrator, as shown in Table 5.15, the topology in Figure 5.23 is achieved. Table 5.15 also shows the concentrator to which the segment has been assigned.

5.4 Design of the Internal Structure of a Local Site using Fuzzy Simulated Evolution Algorithm

When the segments are assigned to the concentrators, the next step is to join these concentrators in a minimum spanning tree topology. For this purpose, we have used the SE_TS algorithm, which has been described in Section 5.2 and was found to be the best among the approaches introduced in that section. We conducted an

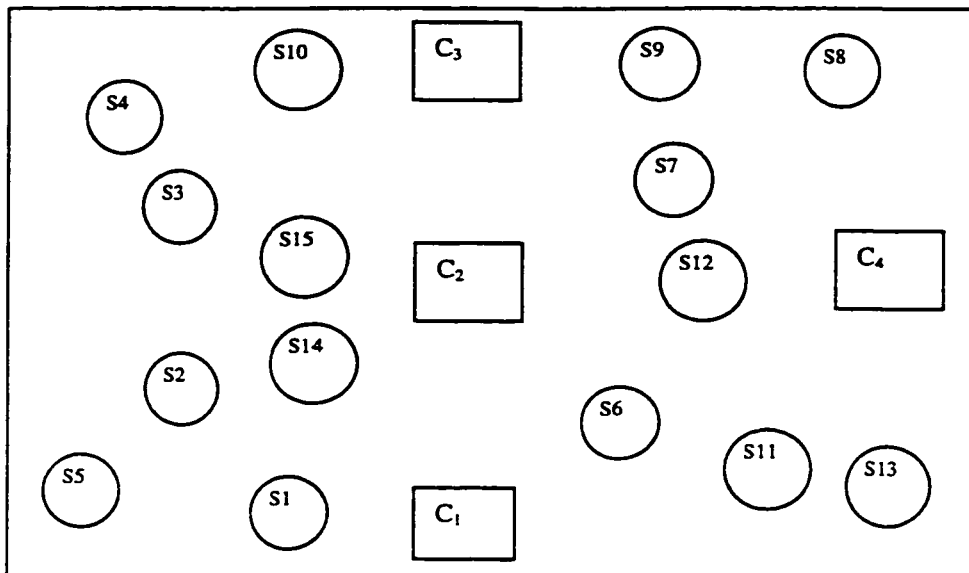


Figure 5.22: Segments and concentrators before assignment.

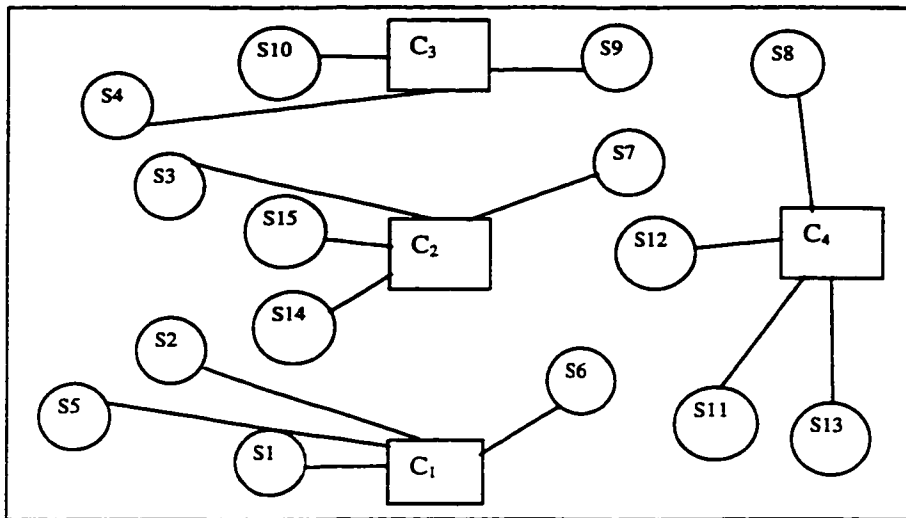


Figure 5.23: Segments and concentrators after assignment.

	C_1	C_2	C_3	C_4	Segment assigned to
S_1	30	38	54	74	C_1
S_2	46	46	52	84	C_1
S_3	50	36	38	78	C_2
S_4	64	54	52	96	C_3
S_5	56	60	70	98	C_1
S_6	24	24	42	26	C_1
S_7	40	22	26	24	C_2
S_8	60	46	38	28	C_4
S_9	52	30	20	50	C_3
S_{10}	56	36	28	72	C_3
S_{11}	32	40	54	22	C_4
S_{12}	36	30	40	16	C_4
S_{13}	50	54	66	22	C_4
S_{14}	22	20	40	60	C_2
S_{15}	30	24	36	44	C_2
C_1		28	52	50	
C_2			28	46	
C_3				50	

Table 5.15: Segment-concentrator and concentrator-concentrator distances in meters. The concentrator to which the segment has been assigned is also shown.

experiment for this. In this experiment, the only difference is the cost of equipment. Since we have assumed that we are using Category 5 cable within a local site, we will include the cost of this (mentioned in Table 5.4) instead of cost of fiber optic cable. Costs of the networking devices are the same as before. For this experiment, we have assumed that the local site is running on 10baseT Ethernet, and the maximum allowed traffic on a link should not exceed 6 Mbps (or 60%). When we apply the SE_TS algorithm to the local site considered in Section 5.3, we get the topology as shown in Figure 5.24. In this topology, we have

- Total traffic = 4.2 Mbps.
- Traffic on link 1,2 = 2 Mbps.
- Traffic on link 1,3 = 2.6 Mbps.
- Traffic on link 3,4 = 2.7 Mbps.
- Monetary Cost = 80065 dollars.
- End-to-end average delay per packet = 2.894 milli seconds.
- Maximum number of hops = 3.

Note that these statistics include the cost, delay, and hops only of the topology that includes the concentrators C_1 to C_4 (and not the segments).

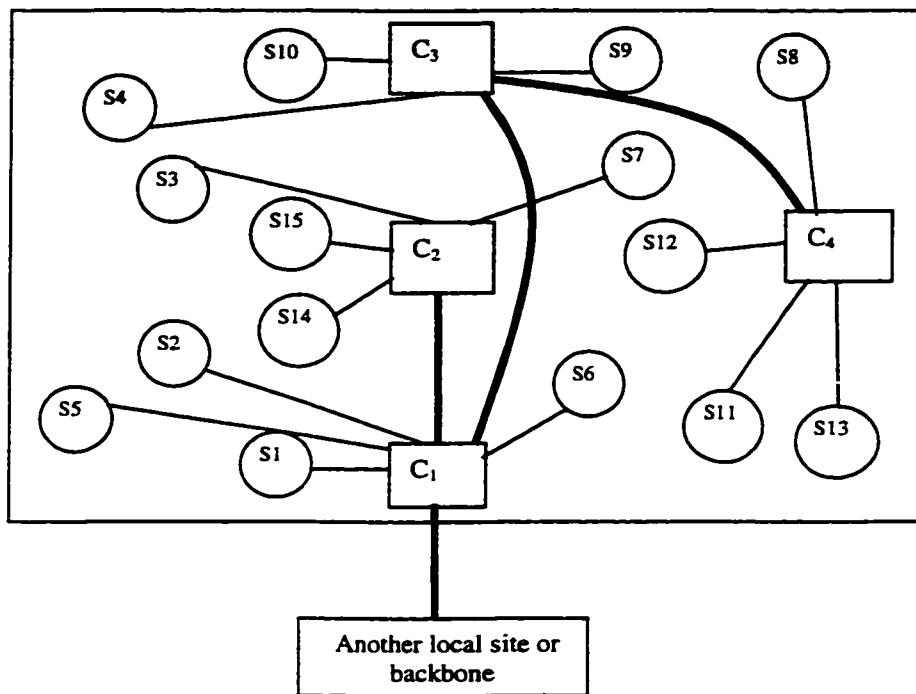


Figure 5.24: Topology generated for concentrators.

5.5 Conclusion

In this chapter, we presented and discussed various experiments that were performed using the approaches based on SE algorithm, namely SE_FF, which used fuzzy evaluation and fuzzy allocation with fixed bias, SE_VB, which used fuzzy evaluation and fuzzy allocation with variable bias, and SE_TS, which used variable bias, fuzzy evaluation, and tabu search characteristics in fuzzy allocation phase. The performance of SE_FF with respect to SE_VB was studied and it was observed that SE_FF performs better than SE_VB. Also, the effect of tabu search characteristics in fuzzy allocation phase of SE_TS was analyzed along with the variation of quality of solution with tabu list size. The performance of SE_TS with respect to SE_FF was also compared and it was found that SE_TS performs better than SE_FF. We also compared SE_TS with Esau-Williams algorithm, which is a greedy algorithm, and found that in general, SE_TS performs far better than Esau-Williams algorithm. The SE_TS algorithm was also compared to simulated annealing (SA) algorithm and it was observed that SE_TS has a better performance. We also used the augmenting path algorithm to assign segments to networking devices within a local site and then used SE_TS algorithm to connect these devices.

Chapter 6

Conclusion

6.1 Summary

Topology Design of Enterprise Networks (TDEN) consists of connecting a large number of computers and peripherals to networking devices and then interconnect these networking devices while satisfying a number of design constraints and optimizing a given objective function. The problem is subdivided into many stages and design at each stage is carried out independently. Being a hard problem, this problem cannot be solved using full enumeration methods. Therefore, intelligent heuristics are used to get sub-optimal solutions for this problem. Simulated Evolution (SE) is a general purpose iterative stochastic heuristic to solve such problems. It combines the iterative improvement and constructive perturbation. We have used SE for optimization of multiple objectives in TDEN problem. We are optimizing *monetary cost*, *aver-*

age network delay, and *maximum number of hops between any source-destination pair*. We have used fuzzy logic based reasoning to combine conflicting objectives. In this thesis, we have fuzzified *evaluation* and *allocation* stages of SE algorithm. Schemes for *fuzzy evaluation* and *fuzzy allocation* are proposed. The effect of fixed bias and variable bias on quality of solution and execution time of the SE algorithm have also been investigated. Also, the effects of incorporating tabu search characteristics in fuzzy allocation phase of variable bias based SE algorithm have been studied. Along with this, we have also studied the performance of variable bias and tabu search based allocation in SE with respect to Esau-Williams algorithm. Augmenting path algorithm has also been used in one stage of the problem.

Following are the conclusions of our research.

- *Fuzzy Evaluation* and *fuzzy allocation* based algorithm, the SE_FF, combines satisfaction of monetary cost and depth of a link in the evaluation phase and satisfaction of monetary cost, average delay, and maximum hops in the allocation phase and uses fixed bias in selection phase. This approach gives better overall results than SE_VB, the variable-bias-based SE_FF, at the expense of higher runtimes.
- The variation of SE_VB that incorporates the tabu search characteristics in fuzzy allocation phase, the SE_TS algorithm, gives the effect of variation of tabu list size and suggests that the tabu list size for best solution increases

with problem size.

- The results obtained for comparison of best fixed bias SE_FF and best tabu list size SE_TS suggest that SE_TS is able to give better overall results than SE_FF, with the observation that SE_TS gives lower monetary cost than best bias SE_FF.
- SE_TS algorithm gives a far better overall performance than Esau-Williams algorithm. With the observation that EW algorithm gives slightly lower monetary cost than SE_TS; SE_TS gives a much lower average delay as well maximum hops than EW algorithm.

In Chapter 1, topology design of enterprise networks (TDEN) problem has been briefly described.

In Chapter 2, literature relevant to TDEN have been reviewed. This review is classified into “constructive heuristics” and “iterative heuristics”.

In Chapter 3, terminology and notation necessary to understand subsequent chapters is included. We have also formally defined the TDEN problem. Objective values computation for monetary cost, average network delay, and maximum number of hops between any source destination pair are also given. In Chapter 3, simulated evolution algorithm and fuzzy logic have also been reviewed.

In Chapter 4, we have described fuzzy logic based SE algorithm for multiobjective optimization of TDEN. For the *evaluation* stage of SE algorithm, a *fuzzy*

evaluation scheme has been proposed which tries to identify links which do not fulfil bounds for monetary cost as well as depth. For the *allocation* phase of the SE algorithm, a *fuzzy allocation* scheme is proposed. This scheme tries to find a topology considering the three objectives, namely, monetary cost, average delay, and maximum hops, by combining them through a fuzzy rule.

In Chapter 5, experimental results have been presented. We started with the backbone design. For this stage, we have studied the effect of bias on the quality of solution. In this chapter, we have compared the best fixed bias SE algorithm (SE_FF) with variable bias SE algorithm (SE_VB). The results show an overall improvement by SE_FF on the quality of solutions, but at the expense of higher execution time. We have also introduced in SE_VB algorithm characteristics of *tabu search* in allocation phase (SE_TS) and observed that as the test case size increases, tabu list size that gives the best solution also increases. We have also compared the best tabu list size SE_TS algorithm with best fixed bias SE_FF algorithm and found that SE_TS gives better results in over all performance, specially with respect to monetary cost. In the same section we have compared SE_TS algorithm with Esau-Williams algorithm (EW) and noted that although SE_TS has a slightly inferior performance than EW algorithm with respect to monetary cost, SE_TS performs much better with respect to the other two objectives. Also, SE_TS has been compared with simulated annealing (SA) algorithm, which is a well-known heuristic. It was found that SE_TS performs better than SA. In the same chapter, we have

used augmenting path algorithm for assignment of segments to concentrators at the local site level, and also used SETS to interconnect these concentrators within the local site.

6.2 Future Research

Our work can be extended to address the following issues.

- Effect of dynamic bias on convergence of the SE algorithm.
- Implementation of a graphical user interface for our program. This interface should allow the users to interactively change parameters online as well as view topology characteristics from time to time.
- Addition of more objectives, such as the average number of hops, in the optimization along with the three objectives we have used.
- Parallelization of the algorithm, especially the allocation phase, as it is the most complex and time consuming phase.

Bibliography

- [1] Habib Youssef, Sadiq M. Sait, and Osama A.Issa. Computer-Aided Design of Structured Backbones. In *15th National Computer Conference and Exhibition*, pages 1–18, October 1997.
- [2] R. Elbaum and M. Sidi. Topological Design of Local-Area Networks Using Genetic Algorithm. *IEEE/ACM Transactions on Networking*, pages 766–778, October 1996.
- [3] M. Gen, K. Ida, and J. Kim. A Spanning Tree-Based Genetic Algorithm for Bicriteria Topological Network Design. In *IEEE International Conference on Evolutionary Computation*, pages 164–173, May 1998.
- [4] C. Ersoy and S. Panwar. Topological Design of Interconnected LAN/MAN Networks. *IEEE Journal on Selected Area in Communications*, pages 1172–1182, October 1993.
- [5] S. Even. *Graph Algorithms*. Computer Science Press, Rockville, MD, 1979.

- [6] Peter C. Fetterolf. Design of Data Networks with Spanning Tree Bridges. In *IEEE International Conference on Systems, Man, and Cybernatics*, pages 298–300, 1990.
- [7] Sadiq M. Sait and Habib. Youssef. *Iterative Computer Algorithms and their Application to Engineering*. IEEE Computer Science Press, Dec. 1999.
- [8] S. Kirkpatrick, C. Gelatt Jr., and M. Vecchi. Optimization by Simulated Annealing. *Science*, pages 498–516, May 1983.
- [9] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, 1989.
- [10] Sadiq M. Sait and Habib. Youssef. *VLSI Physical Design Automation: Theory and Practice*. McGraw-Hill Book Company, Europe, 1995.
- [11] A. Kumar, M. Pathak, and Y. Gupta. Genetic Algorithm-Based Reliability Optimization for Computer Network Expansion. *IEEE Transactions on Reliability*, 24:63–72, 1995.
- [12] B. Dengiz, F. Altiparmak, and A. Smith. Local Search Genetic Algorithm for Optimal Design of Reliable Network. *IEEE Transactions on Evolutionary Computation*, pages 179–188, September 1997.
- [13] B. Dengiz, F. Altiparmak, and A. Smith. Efficient Optimization of All-Terminal Reliable Networks, Using an Evolutionary Approach. *IEEE*

Transactions on Reliability, pages 18–25, March 1997.

- [14] L. Tao and Y. Zhao. Multi-way graph partition by stochastic probe. *Computers Ops Res*, 20(3):321–347, 1993.
- [15] Fogel D.B. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, 5(1):3–14, Jan 1994.
- [16] Ralph Michael Kling. *Optimization by Simulated Evolution and its Application to cell placement*. Ph.D. Thesis, University of Illinois, Urbana, 1990.
- [17] Charles C. Palmer. *An Approach To A problem In Network Design Using Genetic Algorithms*. Ph.D Dissertation, 1994.
- [18] R. Kling and P Banerjee. Optimization by Simulated Evolution with Applications to Standard Cell Placement. In *Proceedings of 27th Design Automation Conference*, pages 20–25, 1990.
- [19] Ralph M. Kling and Prithviraj Banerjee. Empirical and Theoretical Studies of the Simulated Evolution Method Applied to Standard Cell Placement. *IEEE Transactions on Computer-Aided Design*, 10(10):1303–1315, October 1991.
- [20] Ralph M. Kling and Prithviraj Banerjee. ESP: Placement by Simulated Evolution. *IEEE Transactions on Computer-Aided Design*, 8(3):245–255, March 1989.

- [21] Ali S. Hussain. *Fuzzy Simulated Evolution Algorithm for VLSI Cell Placement*. MS Thesis, King Fahd University of Petroleum and Minerals, 1998.
- [22] Dimitri Bertsekas and Robert Gallager. *Data Networks*. Prentice-Hall Inc., second edition, 1992.
- [23] R. C. Prim. Shortest connection networks and some generalizations. *BSTJ*, 36:1389–1401, 1957.
- [24] Aaron Kershenbaum. *Telecommunications Network Design Algorithms*. McGraw-Hill.
- [25] L. R. Esau and K. C Williams. On teleprocessing system design. A method for approximating the optimal network. *IBM System Journal*, 5:142–147, 1966.
- [26] D. Bachmann, M. Segal, M. Srinivasan, and T. Teorey. NetMod: A Design Tool for Large-Scale Heterogeneous Campus Networks. *IEEE Journal on Selected Areas in Communications*, 9(1):15–23, Jan 1991.
- [27] J. Hammond and P. O'Reilly. *Local Computer Networks*. Addison Wesley, 1988.
- [28] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.
- [29] S. Pierre and G. Legault. A Genetic Algorithm for Designing Distributed Computer Network Topologies. *IEEE Transactions on Systems, Man,*

Cybernetics, 28(2):249–258, April 1998.

- [30] R. Boorstyn and H. Frank. Large-Scale Network Topological Optimization. *IEEE Transactions on Communications*, 25(1):29–46, Jan 1977.
- [31] Mischa Schwartz. *Telecommunications Networks: Protocols, Modeling, and Analysis*. Addison-Wesley Publishing Company, 1987.
- [32] Andrew S. Tanenbaum. *Computer Networks*. Prentice-Hall Publishing Company, 1996.
- [33] William L. Schweber. *Data Communications*. McGraw-Hill, Singapore, 1988.
- [34] Gerd. E. Keiser. *Local Area Networks*. McGraw-Hill Book Company., 1989.
- [35] William Stallings. *Data and Computer Communications*. Macmillan Publishing Company, 1994.
- [36] J. M. Mendel. Fuzzy logic systems for engineering: A tutorial. *Proceeding of the IEEE*, 83(3):345–377, March 1995.
- [37] H. J. Zimmerman. *Fuzzy Set Theory and Its Applications*. Kluwer Academic Publishers, third edition, 1996.
- [38] Ronald Y. Yager. On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decisionmaking. *IEEE Transactions on Systems, Man, and Cybernetics volume =*.

- [39] L. A. Zadeh. Fuzzy Sets. *Information Contr.*, 8:338–353, 1965.
- [40] L. A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning. *Information Sciences*, 8:199–249, 1975.
- [41] Hakim Adiche. *Fuzzy Genetic Algorithm for VLSI Floorplan Design*. MS Thesis, 1997.
- [42] F. Glover. *Tabu Search: A Tutorial. Technical Report*. Graduate School of Business Administration, University of Colorado at Boulder, 1990.
- [43] F. Glover. *Tabu Search Fundamentals and Uses. Technical Report*. University of Colorado at Boulder, 1995.

Vitae

- Salman Ahmad Khan
- Born in Karachi, Pakistan.
- Received Bachelor of Science (B.S.) degree in Computer Engineering from King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia in December 1996.
- Joined Computer Engineering Department, KFUPM, as a research assistant in February 1997.
- Received Master of Science (M.S.) degree in Computer Engineering from KFUPM, Dhaharan, Saudi Arabia, in December 1999.