

## INFORMATION TO USERS

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

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

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

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

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

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

UMI<sup>®</sup>





# **QoS-DRIVEN MULTICAST ROUTING ALGORITHMS**

BY

**MUHAMMAD ATIF TAHIR**

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**

**JULY 2001**

UMI Number: 1406102



---

UMI Microform 1406102

Copyright 2001 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 & MINERALS

DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

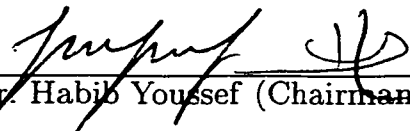
This thesis, written by

**MUHAMMAD ATIF TAHIR**

under the direction of his thesis advisor and approved by his thesis committee,  
has been presented to and accepted by the Dean of Graduate Studies, in partial  
fulfillment of the requirements for the degree of

**MASTER OF SCIENCE IN COMPUTER ENGINEERING**

Thesis Committee

  
Dr. Habib Youssef (Chairman)

  
Dr. Abdulaziz S. Almulhem (Co – chairman)

  
Dr. Sadiq M. Sait (Member)

  
Department Chairman

  
Dean of Graduate Studies

4/7/1401  
Date



Heartily dedicated to my family and especially to

**my dear mother and father**

whose prayers, love, and guidance  
led to this accomplishment.

# Acknowledgements

This thesis would have never been completed without the will and blessing of Allah, the most gracious, the most merciful. AL HAMDU LELLAH..

I acknowledge the support and facilities provided by King Fahd University of Petroleum and Minerals, Dhahran, Kingdom of Saudi Arabia.

I would like to express my profound gratitude and appreciation to my thesis committee chairman Dr. Habib Youssef and co-chairman Dr. AbdulAziz AlMulhem, for their guidance, patience, and sincere advice throughout this thesis. Thanks are also due to my thesis committee member, Dr. Sadiq M. Sait for his comments and critical review of the thesis.

All my family members, especially my parents and grandfather, 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, Aunt, and my only loving brother for their kind support.

Last, but not least, I would like to thank my childhood friends from Karachi, Pakistan. I would also like to thank the friends I made in KFUPM, Dhahran, especially Salman, Junaid, Mahmood, Ahmer, Raslan, Shazli, AbdulAziz. Wasif, Moin, Fareed, Asif, Sajid, Saad, Aamir, Ajmal, and Wasiq who provided a wonderful company. Thanks are also due to Yassir for helping me in the Arabic abstract of this thesis.

# Contents

<b>Acknowledgements</b>	<b>iv</b>
<b>List of Tables</b>	<b>x</b>
<b>List of Figures</b>	<b>xi</b>
<b>Abstract (English)</b>	<b>xvii</b>
<b>Abstract (Arabic)</b>	<b>xviii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Point-to-Point Unicast and Multicast Data Flow . . . . .	2
1.2 IP Multicast Applications and Development . . . . .	4
1.3 Multicast Routing . . . . .	5
1.4 Research Objectives . . . . .	9
<b>2 Problem Formulation</b>	<b>11</b>
2.1 Definitions . . . . .	12



2.2	Minimum Steiner Tree (MST) Problem . . . . .	14
2.3	Delay Constrained Minimum Steiner Tree (CMST) Problem . . . . .	14
2.4	Multiobjective Steiner Tree Optimization (MOST) Problem . . . . .	15
2.5	Multiobjective Steiner Tree Optimization with Dynamic Membership (DMOST) Problem . . . . .	16
2.6	Conclusion . . . . .	16
<b>3</b>	<b>Literature Survey</b>	<b>17</b>
3.1	Minimum Steiner Tree Algorithms . . . . .	17
3.2	Delay Constrained Minimum Steiner Tree Algorithms . . . . .	18
3.3	Dynamic Steiner Tree Algorithms . . . . .	31
3.4	Conclusion . . . . .	33
<b>4</b>	<b>Tabu Search Based Multicast Tree Design</b>	<b>34</b>
4.1	Tabu Search . . . . .	35
4.1.1	Overview of Tabu Search . . . . .	36
4.2	Delay Constrained Minimum Steiner Tree (CMST) Design . . . . .	38
4.2.1	Initial Solution . . . . .	38
4.2.2	Neighborhood Solutions . . . . .	40
4.2.3	Tabu Moves . . . . .	42
4.2.4	Aspiration Criterion . . . . .	43
4.2.5	Termination Rule . . . . .	43

4.3	Unconstrained Minimum Steiner Tree (MST) Design . . . . .	43
4.4	Multiobjective Steiner Tree Optimization (MOST) Design . . . . .	44
4.4.1	Membership Function for Cost . . . . .	48
4.4.2	Membership Function for End-to-End Delay . . . . .	49
4.4.3	Membership Function for Number of Steiner nodes . . . . .	50
4.4.4	Membership Function for Delay Variance . . . . .	51
4.4.5	An Example . . . . .	52
4.5	Time Complexity Analysis of Tabu Search . . . . .	56
4.6	Conclusion . . . . .	56
<b>5</b>	<b>Experiments and Simulation Results</b>	<b>58</b>
5.1	The Experimental Setup . . . . .	59
5.2	Simulation Results for Unconstrained Multicast Tree Design Problem	62
5.2.1	Comparison of Tabu Search and KMB . . . . .	63
5.2.2	Comparison of Tabu Search and RPM . . . . .	66
5.3	Simulation Results for Delay Constrained Multicast Tree Design Problem . . . . .	69
5.3.1	Comparison of Tabu Search with SC and MSC . . . . .	70
5.3.2	Comparison of Tabu Search with KPP1 and KPP2 . . . . .	73
5.3.3	Comparison of Tabu Search and CAO . . . . .	77
5.3.4	Comparison of Tabu Search and BSMA . . . . .	80

5.3.5	Quality of Solution by Tabu Search . . . . .	83
5.4	Simulation Results for Multiobjective Minimum Steiner Tree Problem	85
5.4.1	Comparison of KPP and Tabu . . . . .	85
5.4.2	Comparison of CAO and Tabu . . . . .	87
5.4.3	Comparison of BSMA and Tabu . . . . .	87
5.4.4	Quality of Solution by Tabu Search . . . . .	89
5.5	Conclusion . . . . .	94
<b>6</b>	<b>Dynamic Multicast Routing</b>	<b>95</b>
6.1	Proposed Approach . . . . .	96
6.1.1	Nodes Leaving . . . . .	96
6.1.2	Nodes Joining . . . . .	97
6.2	Experiment Setup and Simulation Results . . . . .	98
6.3	Conclusion . . . . .	102
<b>7</b>	<b>Conclusions and Future Work</b>	<b>103</b>
7.1	Summary . . . . .	103
7.2	Conclusions . . . . .	105
7.3	Future Work . . . . .	106
	Bibliography . . . . .	107

# List of Tables

5.1	Comparison of Tabu and KPP. N= Network size, D=Maximum end to end delay in <i>msec</i> , C=Cost in <i>Mbps</i> , V=Delay variance in $\mu sec$ , and S = Number of Steiner nodes. Group size = 10. . . . .	86
5.2	Comparison of Tabu and KPP. G= Group size, D=Maximum end to end delay in <i>msec</i> , C=Cost in <i>Mbps</i> , V=Delay variance in $\mu sec$ , and S = Number of Steiner nodes. Network size = 50. . . . .	86
5.3	Comparison of Tabu and CAO. N= Network size, D=Maximum end to end delay in <i>msec</i> , C=Cost in <i>Mbps</i> , V=Delay variance in $\mu sec$ , and S = Number of Steiner nodes. Group size = 10. . . . .	88
5.4	Comparison of Tabu and CAO. G= Group size, D=Maximum end to end delay in <i>msec</i> , C=Cost in <i>Mbps</i> , V=Delay variance in $\mu sec$ , and S = Number of Steiner nodes. Network size = 50. . . . .	88
5.5	Comparison of tabu and BSMA. N= Network Size, D=Maximum end to end delay in <i>msec</i> , C=Cost in <i>Mbps</i> , V=Delay variance in $\mu sec$ , and S = Number of Steiner nodes. Group size = 10. . . . .	89

5.6	Comparison of Tabu and BSMA. G= Group size, D=Maximum end to end delay in <i>msec</i> , C=Cost in <i>Mbps</i> , V=Delay variance in $\mu sec$ , and S = Number of Steiner nodes. Network size = 50. . . . .	90
6.1	Comparison of Dynamic Greedy and KPP. E= Event, J=Join, L=Leave. D=Maximum end to end delay in <i>msec</i> , C=Cost in <i>Mbps</i> , V=Delay variance in $\mu sec$ , and S = Number of Steiner nodes. Initial group size = 10. . . . .	101

# List of Figures

1.1	Unicast dataflow. . . . .	3
1.2	Multicast dataflow. . . . .	3
1.3	Multicast-Enabled components. . . . .	6
3.1	The graph of example 1. . . . .	19
3.2	The closure graph on $D = \{b,d,c,f\}$ with $U = 5$ and $s = a$ . . . . .	20
3.3	(a)-(d) The four stages in constructing the spanning tree using $f_{CD}$ . . . . .	22
3.4	(a)-(d) The four stages in constructing the spanning tree using $f_C$ . . . . .	22
3.5	The constrained spanning tree using $CST_{CD}$ . . . . .	23
3.6	The optimal constrained Steiner tree: cost = 20 with $U = 5$ . . . . .	23
3.7	Minimum delay Steiner tree: cost = 32 with delay $U = 5$ . . . . .	25
3.8	Example of Path replacement: path $ab$ in Figure 3.7 is replaced by path $bc$ . Cost = 24 with delay $U = 5$ . . . . .	25
4.1	Algorithmic description of a short-term Tabu Search (TS) [1]. . . . .	37
4.2	An example network. . . . .	39

4.3	Sink tree for source A, cost = 22 with $U = 5$ .	39
4.4	Sink trees for destinations $D = \{B, C, D, F\}$ .	41
4.5	Two possible neighbors from current solution.	41
4.6	Final solutions from current solution.	42
4.7	Membership function for a fuzzy set A.	47
4.8	Basic components for a good topology.	48
4.9	Membership function for cost.	49
4.10	Membership function for end-to-end delay.	50
4.11	Membership function for Steiner nodes.	51
4.12	Membership function for delay variance.	52
4.13	Minimum delay tree.	53
4.14	Minimum cost tree.	53
5.1	A randomly generated network, 20 nodes, average degree 4.	61
5.2	Symmetric load. Cost comparison of KMB and Tabu. Group size =10.	64
5.3	Symmetric load. Cost comparison of KMB and Tabu. Network size =80.	64
5.4	Asymmetric load. Cost comparison of KMB and Tabu. Group size =10.	65
5.5	Asymmetric load. Cost comparison of KMB and Tabu. Network size =100.	65
5.6	Number of multicast session versus cost between KMB and Tabu. Network Size =20, Group size=5.	66
5.7	Symmetric load. Cost comparison of RPM and Tabu. Group size =10.	67

5.8	Symmetric load. Cost comparison of RPM and Tabu. Network size =80. .	67
5.9	Asymmetric load. Cost comparison of RPM and Tabu. Group size =10. .	68
5.10	Asymmetric load. Cost comparison of RPM and Tabu. Network size=100.	68
5.11	Number of multicast session versus cost between RPM and Tabu. Network size =20, Group size=5. . . . .	69
5.12	Symmetric load. Cost comparison of SC, MSC and Tabu. Group size =10 and U=0.05 Sec. . . . .	71
5.13	Asymmetric load. Cost comparison of SC, MSC and Tabu. Group size =10 and U=0.04 Sec. . . . .	71
5.14	Asymmetric load. Cost comparison of SC, MSC and Tabu. Group size =10, Network size=100 nodes. . . . .	72
5.15	Asymmetric load. Cost comparison of MSC and Tabu. U = 0.05 sec, Network size=100 nodes. . . . .	72
5.16	Number of multicast session versus cost between SC, MSC, and Tabu. Network size =20, Group size=5. . . . .	73
5.17	Symmetric load. Cost comparison of KPP2, KPP1 and Tabu. Group size =10 and U=0.05 Sec. . . . .	74
5.18	Asymmetric load. Cost comparison of KPP2, KPP1 and Tabu. Group size =10 and U=0.04 Sec. . . . .	74
5.19	Asymmetric load. Cost comparison of KPP2, KPP1 and Tabu. Group size =10, Network size=100 nodes. . . . .	75



5.20 Asymmetric load. Cost comparison of KPP2, KPP1 and Tabu. $U = 0.05$ sec, Network size=100 nodes. . . . .	75
5.21 Number of multicast session versus cost between KPP1 and Tabu. Network size =20, Group size=5. . . . .	76
5.22 Number of multicast session versus cost between KPP2 and Tabu. Network size =20, Group size=5. . . . .	76
5.23 Symmetric load. Cost comparison of CAO and Tabu. Group size =10 and $U=0.05$ Sec. . . . .	77
5.24 Asymmetric load. Cost comparison of CAO and Tabu. Group size =10 and $U=0.04$ Sec. . . . .	78
5.25 Cost comparison of CAO and Tabu. Group size =10, Network size=100 nodes. . . . .	78
5.26 Cost comparison of CAO and Tabu. $U = 0.05$ Sec, Network size=100 nodes.	79
5.27 Number of Multicast Session versus cost between CAO and Tabu. Network size =20, Group size=5. . . . .	79
5.28 Symmetric load. Cost comparison of BSMA and Tabu. Group size =10 and $U=0.05$ Sec. . . . .	80
5.29 Asymmetric load. Cost comparison of BSMA and Tabu. Group size =10 and $U=0.04$ Sec. . . . .	81
5.30 Cost comparison of BSMA and Tabu. Group size =10, Network size=100 nodes. . . . .	81

5.31	Cost comparison of BSMA and Tabu. $U = 0.05$ Sec, Network size=100 nodes. . . . .	82
5.32	Number of multicast session versus cost between BSMA and Tabu. Net- work size =20, Group size=5. . . . .	82
5.33	Total number of solutions evaluated by tabu search during 500 iterations for different cost ranges. Network size = 100 nodes, Group size = 10 and $U=0.05$ Sec. . . . .	83
5.34	Total number of solutions evaluated by tabu search versus iteration number for different cost ranges. Network size = 100 nodes, Group size = 10 and $U=0.05$ Sec. . . . .	84
5.35	Cost of best solution found by tabu search versus iteration number for 3 networks. Group size = 10 and $U=0.05$ Sec . . . . .	84
5.36	Four different objective functions versus iterations. Network size = 50 Nodes. Group size = 20. . . . .	91
5.37	Four different objective functions versus iterations. Network size = 100 Nodes. Group size = 10. . . . .	92
5.38	Total number of solutions evaluated by tabu search during 1000 iterations for different membership ranges. Network size = 100 nodes, Group size = 10. . . . .	93

5.39	Total number of solutions evaluated by tabu search as a function of time (Iteration number) for different membership intervals. Network size = 100 nodes and Group size =10. . . . .	93
6.1	Network size = 30 nodes, Initial group size =10. (a) Cost versus Event number between KPP and Greedy. (b) Delay variance versus Event number between KPP and Greedy. (c) Steiner nodes versus Event number between KPP and Greedy. . . . .	100

## THESIS ABSTRACT

**Name:** Muhammad Atif Tahir  
**Title:** QoS-Driven Multicast Routing Algorithms  
**Major Field:** COMPUTER ENGINEERING  
**Date of Degree:** July 2001

*The research and industrial communities have recently been striving to make the Internet capable of supporting unicast and multicast traffic sources with Quality of Service (QoS) guarantees. QoS parameters targeted are guaranteed throughput, end-to-end delay, and delay variation. A multicast tree for a given multicast source is a tree rooted at the source and all its leaves being members in the multicast group. Tree cost is measured by the utilization of tree links. Also tree cost is highly correlated with the number of Steiner tree nodes, i.e., nodes which are not members of the multicast group. In this thesis, a tabu search algorithm is proposed for three different multicast routing problems involving the above QoS parameters. Our proposed algorithms are then compared with other proposed techniques on numerous sample networks. On all tests, the proposed tabu search algorithms were able to find better multicast trees than those reported in the literature. Also, a greedy algorithm is proposed for multicast routing problem with dynamic membership.*

## MASTER OF SCIENCE DEGREE

King Fahd University of Petroleum and Minerals, Dhahran.  
July 2001

## ملخص الرسالة

الاسم: محمد عاطف طاهر

عنوان الدراسة: خوارزميات توجيهها نوعية الخدمة لتحديد المسارات بشبكات البث الانتقائي

التخصص: هندسة الحاسب الآلي

تاريخ التخرج: يوليو 2001

إنصبت جهود العاملين في مجال البحث و الصناعة في الآونة الأخيرة لجعل شبكة المعلومات العالمية قادرة على دعم مصادر المعلومات في حالتى البث الاحادى و الانتقائى بنوعية الخدمة (QoS) المطلوبة. عناصر نوعية الخدمة المنشودة هي ضمان مقدار معين من: المخرج، زمن التأخير الكلى، و التغير فى زمن التأخير. شجرة البث الانتقائى لمصدر معين هي شجرة اصلها هو المصدر و كل فروعها أعضاء فى شجرة البث الانتقائى. تقاس كلفة شجرة البث بالمستخدم من موصلاتها. كما ان كلفة الشجرة ذات ارتباط وثيق بعقد الشبكة التى ليست جزءاً من شجرة البث الانتقائى. فى هذه الرسالة تم إقتراح خوارزمية بحث مرحل (Tabu Search) بهدف معالجة ثلاث مشاكل مختلفة فى إختيار المسارات فى شبكات البث الانتقائى ذات صلة بعناصر نوعية الخدمة المذكورة انفاً. ومن ثم تم مقارنة هذه الخوارزميات بالخوارزميات الأخرى المقترحة باستخدام عدد من الشبكات. فى كل الإختبارات كانت خوارزمياتنا ذات البحث المرحل قادرة على ايجاد شجرة بث إنتقائى افضل من تلك التى تم ايجاعها فى البحوث المنشورة. كذلك تم إقتراح خوارزمية أحادى الحل لمعالجة مشكلة إختيار المسارات فى شبكات البث الانتقائى ذات العضوية المتغيرة.

## درجة الماجستير في العلوم

جامعة الملك فهد للبترول والمعادن

الظهران ، المملكة العربية السعودية

يوليو 2001

# Chapter 1

## Introduction

Internet is widely recognized to deliver connectivity to the data world. The monetary investment in the Internet is enormous. Many Internet applications involve one-to-many or many-to-many (multipoint) communications, where one or more sources send data to multiple receivers. It is possible to provide transmissions to multiple receivers in three different ways: *unicast*, where a separate copy of the data is delivered to each recipient; *broadcast*, where a data packet is forwarded to all portions of the network even if only a few of the destinations are intended recipients; and *multicast*, where a single packet is addressed to all intended recipients and the network replicates packet only as needed. Internet applications that use unicast include Email and Web browsers. Potential applications of multicasting are in the business world, to help to increase the ability of organizations to communicate and collaborate, leveraging more value from network investment. Examples

of multicasting are the transmission of corporate messages to employees, video and audio conferencing for remote meetings and teleconferencing, replicating databases and web site information, live transmission of multimedia training and university courses, communication of stock quotes to brokers, updates on the latest election results, collaborative computing, transmission over networks of live TV or radio news and entertainment programs, and many more. These new applications are compelling the need for advances in traffic handling to overcome bottlenecks [2].

IP Multicast is an efficient, standard based solution with broad industry support. It is an extension of IP, the internetworking protocol that is used on the Internet. It is an important advancement in IP networking where applications are able to send a copy of the information to a group of addresses, reaching all registered recipients. Without multicasting, the same information must be either carried over the network multiple times, one time for each recipient, or broadcasted to everyone on the network, consuming unnecessary bandwidth and processing power. IP Multicast technologies address the needed mechanisms at different levels in the network and internetworking infrastructure to efficiently handle group communications.

## **1.1 Point-to-Point Unicast and Multicast Data Flow**

Figures 1.1 and 1.2 show the difference between unicast and multicast data flows [2].

In Figure 1.1, four copies of the same data (D) are sent point-to-point as D1, D2,

D3 and D4 to Receivers 1, 2, 3 and 4 in the same application. These are unicast transmissions, where point-to-point communication is set from one sender to a single receiver. In Figure 1.2, one copy of the same data (D) is sent to Receivers 1, 2, 3 and 4 using the same application. Copies of the data is generated only where necessary. In this example, the edge router makes four copies, one for each receiver. It is needless to mention the significant saving in the bandwidth locally and across the network. Even more bandwidth saving could be achieved if a very large number of receivers are involved in the communication session.

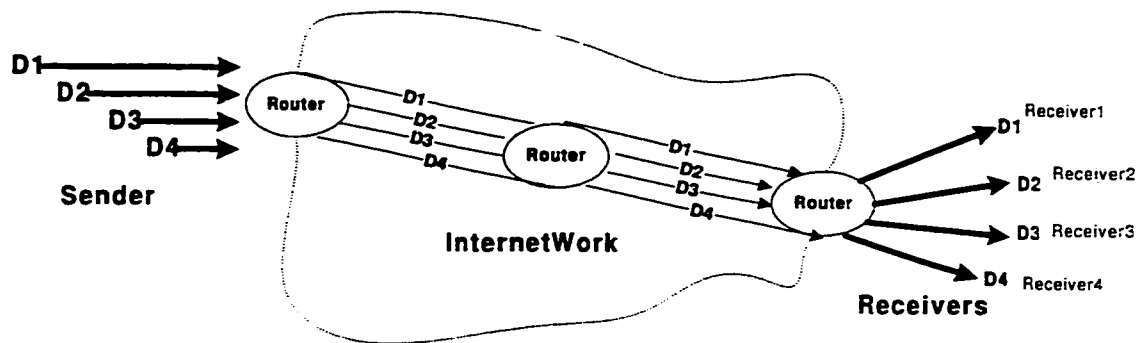


Figure 1.1: Unicast dataflow.

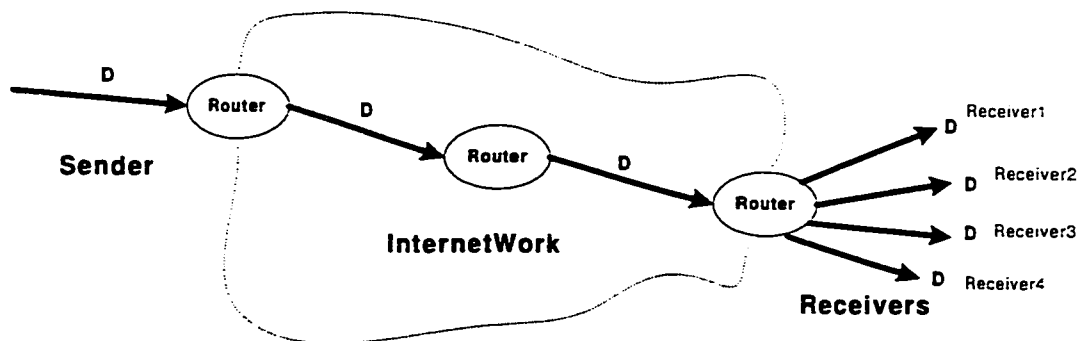


Figure 1.2: Multicast dataflow.



## 1.2 IP Multicast Applications and Development

Demand for multimedia, combining audio, video and data streams over a network, is rapidly increasing. Some of the most popular uses of multimedia are real-time interactive applications such as desktop video and audio conferencing, collaborative engineering, shared white boards, transmission of university lectures to a remote audience, and animated simulations. Even when data compression is used, multimedia applications occupy substantial portion of the available bandwidth. IP Multicast will help in the realization of the full potential of these exciting new applications. For example, consider the transmission of a corporate presentation to workers within a company. Using unicast transmission, it would be possible to support only a small numbers of recipients, because transmission of multiple copies of the multimedia stream would quickly strain the available network bandwidth besides it would linearly increase monetary cost. On the other hand, with IP Multicast it would be easy to support thousands of recipients, each sitting at his or her own desk. Another important type of multimedia application involves the transmission of stored data streams. Examples include updates of web caches, video server to video server updates, corporate announcements to employees, etc.

IP Multicast will enable these applications to scale to very large numbers of recipients. Although multimedia applications are bandwidth intensive, non-multimedia applications that involve the transfer of large databases of information will also

benefit immensely from IP Multicast. Examples of real-time applications of this type include stock/commodities quotes and trading information, and shared white boards. Non-real time applications include multicast file transfer and web caching.

IP multicasting depends on three components: (1) protocols for establishing and controlling multicast groups, (2) a router infrastructure for distribution of multicast traffic, and (3) application protocols and APIs (Application Programming Interfaces). Figure 1.3 depicts, at a high level, components that must be multicast-enabled which are marked with an asterisk [2]. The direction of traffic shown is for multicast datagrams. Traffic needed to communicate host group membership and routing information is not shown.

## 1.3 Multicast Routing

To support multicast communication efficiently, one of the key issues that needs to be addressed is routing. Routing primarily refers to the determination of a set of paths to be used for carrying the messages from source nodes to destination nodes. In multicasting, the routing function mainly finds the best route from source to all destinations in a multicast group. It is important that the routes used for such communications consume a minimal amount of resource. In order to use network resources as little as possible while meeting the network service requirements, the highly recommended solution involves the generation of a multicast tree spanning

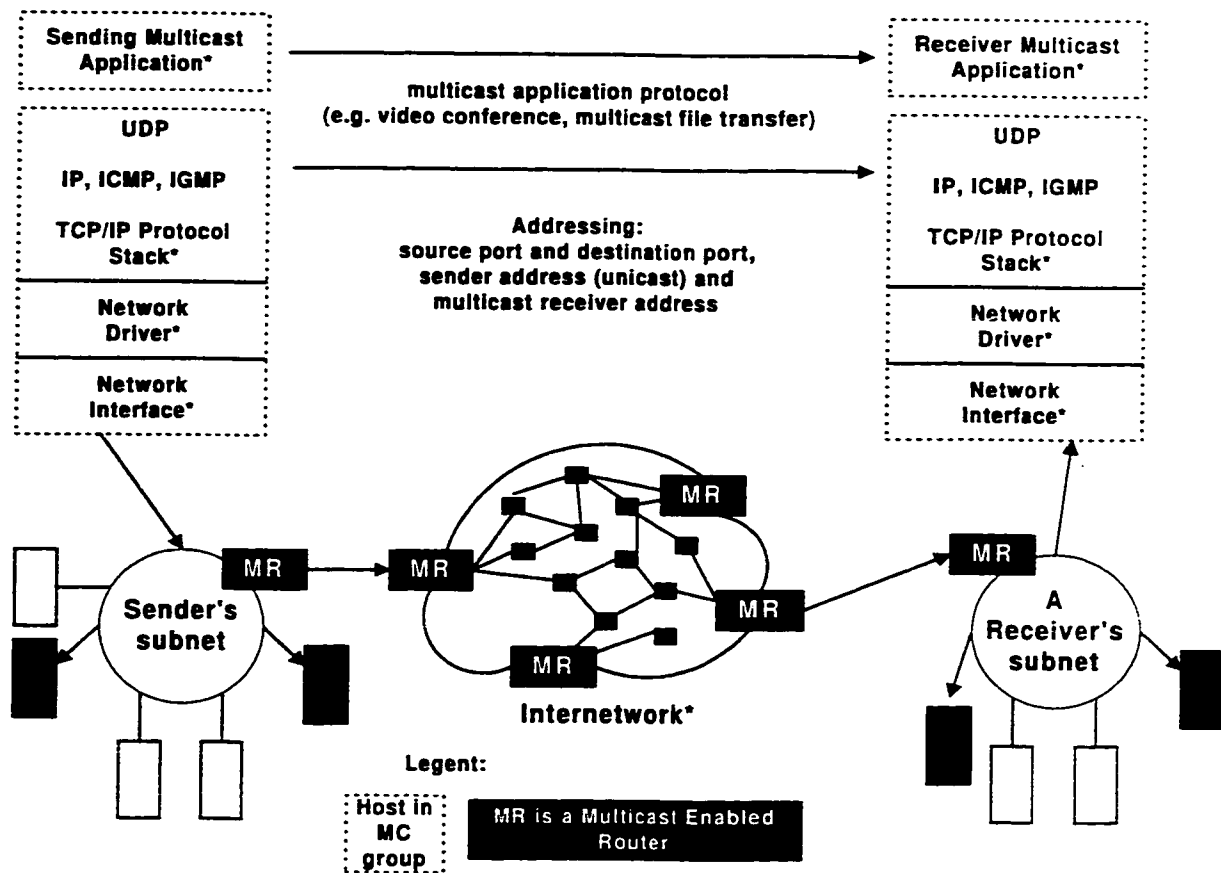


Figure 1.3: Multicast-Enabled components.

the source and destination nodes.

Minimum Steiner tree (MST) algorithms attempt to minimize the total cost of the multicast tree. Total cost of the multicast tree is generally defined as the sum of costs of all edges in the multicast tree. The cost is usually measured as the bandwidth consumed by the tree. The minimum Steiner problem is known to be NP-Complete.

With the advent of the real-time interactive applications, minimizing delay of the multicast tree is also an important objective along with minimizing cost. The delay corresponds to the time required to deliver a packet from the source to any member of the multicast group. These two goals can be used in the multicast routing algorithm to determine what constitutes a good tree. However, the cost and delay measures individually are insufficient to characterize a good multicast routing tree for interactive multimedia communication. For example, when the optimization objective is only to minimize the total cost of the tree, we will have a minimum cost tree. Although total cost as a measure of bandwidth efficiency is certainly an important parameter, it is not sufficient to characterize the quality of the tree as perceived by interactive multimedia and real time applications. This is because, networks supporting real-time traffic need to provide certain quality of service guarantees in terms of the end-to-end delay along the individual paths from sources to each destination nodes. Therefore, both cost optimization and delay optimization goals are important for the multicast routing tree construction. The

problem of minimizing tree cost under the constrained that all path delays are within a user-specified delay bound is referred to as the *delay constrained Steiner tree problem*.

There are also certain classes of applications in which minimizing delay variation and minimizing number of Steiner nodes are important parameters along with minimizing cost and minimizing delay. For example, with humans at the end-user. providing a minimum delay variation would be extremely helpful in maintaining the feeling of a face-to-face discussion for interactive applications. Also, with video and audio conferencing now available on the Internet, it is desirable for all the viewers to see and hear the speaker at (almost) the same time. A hop or relay node is defined as a node excluding the source and destination nodes in the multicast tree. Minimizing number of Steiner nodes is important in multicast routing. For certain applications, security is an important issue. By minimizing the number of Steiner nodes, unnecessary visits of traffic across the network can be avoided. We refer to this problem that minimizes cost, maximum end-to-end delay, number of Steiner nodes, and delay variation as the *multiobjective Steiner tree optimization problem*.

If nodes are allowed to join or leave the multicast group any time during the lifetime of the multicast connection, then the problem is called dynamic multicast routing problem. In graph theory, this problem is also known as the dynamic Steiner tree problem [3]. In the dynamic multicast routing problem, we are given a source node and a sequence of requests  $R = r_1, r_2, \dots, r_m$ . Each request  $r_i$  is for either

adding or removing a destination node to or from the multicast group. In response to request  $r_i$ , a multicast tree  $T_i$  is constructed by a dynamic multicast routing algorithm on  $S_i$  where  $S_i$  be the set of nodes in the multicast group. That is,  $T_i$  spans  $S_i$ . Given a sequence of requests  $R$  and an initial tree  $T_0$ , the dynamic multicast routing problem consists of finding a sequence of multicast trees  $T_i, i = 1, 2, \dots, m$  such that its cost is optimized [4, 5, 6]. If Quality of Service (QoS) parameter is included, such as a delay constraint, then it is called as delay-constrained minimum cost multicast-routing problem with dynamic membership [7, 8]. In this work, we seek to optimize the number of Steiner nodes and delay variation besides the bandwidth and delay QoS parameters. We refer to this problem as the *multiobjective multicast-routing problem with dynamic membership*.

## 1.4 Research Objectives

In this thesis, a tabu search based approach for the three previously mentioned problems namely: (1) the minimum Steiner tree (MST) problem, (2) the delay-constrained minimum Steiner tree (CMST) problem, and (3) the multiobjective minimum Steiner tree optimization (MOST) problem are proposed. A fuzzy logic based tabu search algorithm is proposed for the multiobjective optimization problem. The proposed algorithm is compared with many existing multicast algorithms. Results show that on almost all test cases, Tabu Search algorithm exhibits more

intelligent search of the solution subspace and is able to find better solutions than other reported multicast algorithms. A greedy approach for Dynamic Multicast routing problem is also proposed in this thesis.

## Organization of Thesis

The rest of this thesis is organized as follows. Chapter 2 gives a formal description of the problem. In Chapter 3, we review related literature. This chapter covers different approaches that have been attempted in Steiner tree problems.

In Chapter 4, we discuss our proposed fuzzy logic based tabu search algorithm for multiobjective optimization problem. Further, we describe different steps of our fuzzy tabu search algorithm, and the methodology to implement each of them. Also, a tabu search algorithm for Steiner tree and delay constrained Steiner tree problem is explained. In Chapter 5, we present and discuss experiments and results.

In Chapter 6, dynamic multicast routing problem is discussed. A greedy algorithm is proposed for multiobjective minimum steiner tree optimization with dynamic membership problem followed by experiments and simulation results. The thesis ends with conclusion and future work in Chapter 7.

# Chapter 2

## Problem Formulation

In this chapter, the formulation of minimum Steiner tree problem, delay constrained minimum Steiner tree problem, multiobjective minimum Steiner tree problem and multiobjective minimum Steiner tree problem with dynamic membership is given. The multiobjective minimum Steiner tree involves simultaneous optimization of cost, maximum end-to-end delay, delay variance, and the number of Steiner nodes.

This chapter is organized as follows. In Section 2.1, some basic definitions are given that are used throughout the thesis. Sections 2.2 - 2.5 describe mathematically the minimum Steiner tree problem, the delay constrained minimum Steiner tree problem, the multiobjective Steiner tree optimization problem, and the multiobjective Steiner tree optimization with dynamic membership problem respectively. Section 2.6 concludes the chapter.



## 2.1 Definitions

The communication network is modeled as a graph  $G = (V, E)$ , where  $V$  is a set of nodes and  $E$  is a set of edges (links). A link represents a physical connection between two nodes. The cost and delay of an edge  $e$  are denoted by  $cost(e)$  and  $delay(e)$  respectively, where  $cost(e)$  and  $delay(e)$  are positive real numbers. A directed network model is assumed, i.e., the links  $e = (u, v)$  and  $e = (v, u)$  cannot be used interchangeably.

- A node is a switch in a wide area network.
- A link represents a physical connection between two nodes. The existence of a link from node A to node B implies the existence of a link in the reverse direction from node B to node A. The network is not symmetric, thus cost and delay functions of the two links are not necessarily equal. Different capacities may be assigned to the different links in the network.
- A network consists of a set of nodes and a set of links interconnecting these nodes such that:

there is a path from any node to all other nodes in the network, and

there exists at most one link between any two nodes.

Thus from graph theory point of view a network is a simple connected directed graph.

- A multicast tree for a given multicast source is a tree rooted at the source and all its leaves being members in the multicast group.
- The link cost is a function of the amount of traffic traversing the link  $e$  and the expected buffer space needed for that traffic [9, 10].
- The delay of a link is the sum of the queuing delay, transmission delay and propagation delay of the link. Let  $D$  be the set of destination nodes, then, for each path from a source  $s$  to any destination  $d \in D, D \subseteq V$ , the delay of the path is defined as the sum of link delays along the path:

$$delay(\Pi_{s,d}) = \sum_{e \in \Pi_{s,d}} delay(e) \quad \text{where } d \in D$$

The maximum end-to-end delay of a multicast tree is the maximum delay from the source to any member in the multicast group, that is;

$$Max\_Delay(\Pi_{s,d}) = \max(\sum_{e \in \Pi_{s,d}} delay(e)) \quad \text{where } d \in D$$

- An upper bound  $U$  is assigned to the delay along every path from a source to any destination  $d \in D, D \subseteq V$ . The delay bound for each destination node can be different since each communication link in the network can have different delay constraints as specified by the multicast application.
- The delay variance is defined as the variance over the set of end-to-end delays along the paths  $\Pi_{s,d}$  where  $d \in D$ . This delay variance is used to address

the problem of delay variation in the tree. Mathematically, the delay variance function is defined as

$$Var(T) = variance(\sum_{e \in \Pi_{s,d}} delay(e)) \quad \text{where } d \in D$$

- The number of Steiner nodes is defined as the intermediate nodes in the multicast tree excluding the source and destination nodes.

## 2.2 Minimum Steiner Tree (MST) Problem

The minimum Steiner tree problem [4] is to determine a multicast tree connecting the source node  $s$  to every destination node  $d \in D$ , such that the total cost of this tree is minimal. Mathematically, the problem is to find a tree  $T = (V_T, E_T)$  where  $V_T \subseteq V$  and  $E_T \subseteq E$  such that the total cost of this tree  $\sum_{e \in E_T} cost(e)$  is minimized subject to the constraint  $\{s\} \cup D \subseteq V_T$ .

## 2.3 Delay Constrained Minimum Steiner Tree (CMST) Problem

The delay constrained minimum Steiner problem [11, 12] is to determine a multicast tree connecting the source node  $s$  to every destination  $d \in D$  node such that the cost of this tree is minimal while the total delay from the source node to any destination

node is smaller than  $U$ . Mathematically, the problem is to find a tree  $T = (V_T, E_T)$  where  $V_T \subseteq V$  and  $E_T \subseteq E$  such that the total cost of this tree  $\sum_{e \in E_T} cost(e)$  is minimized subject to the following two constraints:

1.  $\{s\} \cup D \subseteq V_T$ ;
2.  $\sum_{e \in \Pi_{s,d}} delay(e) \leq U \forall d \in D$ , where  $\Pi_{s,d}$  is the set of edges constituting the path from source node  $s$  to destination node  $d$  in the tree  $T$ .

## 2.4 Multiobjective Steiner Tree Optimization (MOST)

### Problem

The multiobjective Steiner tree optimization problem is to determine a multicast tree connecting the source node  $s$  to every destination  $d \in D$  satisfying the source bandwidth requirement and such that the total cost of the tree  $\sum_{e \in E_T} cost(e)$ , maximum end to end delay  $\max(\sum_{e \in \Pi_{s,d}} delay(e))$  where  $d \in D$ , the delay variance  $variance(\sum_{e \in \Pi_{s,d}} delay(e))$  where  $d \in D$ , and the number of Steiner nodes are minimized.

## 2.5 Multiobjective Steiner Tree Optimization with Dynamic Membership (DMOST) Problem

Given a graph  $G = (V, E)$  with two weighted functions  $cost(e)$  and  $delay(e)$  on edge  $e$ , a source  $s$ , a set of destinations  $D$ , and considering the dynamic condition with a sequence of connection requests  $R = r_1, r_2, \dots, r_m$ , the multiobjective minimum Steiner tree problem with dynamic membership is to determine a multicast tree connecting the source node  $s$  to every destination node  $d \in D$  for every request, satisfying the source bandwidth requirement and such that the total cost of the tree, maximum end-to-end delay, the delay variation, and the number of Steiner nodes are minimized.

## 2.6 Conclusion

In this chapter, we have formulated minimum Steiner tree problem, delay constrained minimum Steiner tree problem and multiobjective minimum Steiner tree problem and multiobjective minimum Steiner tree problem with dynamic membership problem. Important criteria for multicast routing algorithms are also discussed in this chapter.

# Chapter 3

## Literature Survey

The different multicast routing algorithms are presented in this chapter with their distinguishing features. It is beyond the scope of this survey to list the pseudo code of each individual algorithm. This chapter is organized as follows. Section 3.1 describes previously proposed algorithms for minimum Steiner tree problem. In Section 3.2, algorithms for delay constrained minimum Steiner tree problem have been discussed followed by algorithms for dynamic Steiner tree problem in Section 3.3. Section 3.4 concludes the chapter.

### 3.1 Minimum Steiner Tree Algorithms

The objective of the minimum Steiner tree problem is to minimize the total cost of the multicast tree. Very few algorithms have been proposed for the minimum

Steiner tree problem in directed network [13]. The minimum Steiner tree problem is known to be NP-complete. KMB [14] is an efficient unconstrained minimum Steiner tree heuristic for undirected networks. The KMB heuristic uses Prim's minimum spanning tree algorithm [15] during its computation. Prim's algorithm is optimal only for symmetric networks. Thus the cost performance of the KMB heuristic may be optimized if it is applied to asymmetric networks. The worst case time complexity of the KMB heuristic is  $O(MV^2)$ , where  $M$  is the size of the multicast group. The total cost of trees generated using KMB heuristic in symmetric networks is on the average 5% worse than the cost of the optimal minimum Steiner tree [16].

An efficient solution for the Steiner tree problem with application to multicast routing is presented by Ricudis [17]. The solution is based on a hybridized Genetic algorithm with a hill climbing technique that facilitates better local exploration of the solution search space.

## 3.2 Delay Constrained Minimum Steiner Tree Algorithms

The delay-constrained source-specific minimum Steiner tree problem was first formulated by Kompella, Pasquale, and Polyzos [11]. The authors proved the NP-completeness of the problem. In their algorithm, it is necessary to solve the delay constrained shortest path problem for  $k(k+1)/2$  number of times, where  $k$  is the

number of source and destination nodes. The algorithm uses source routing. The steps of the algorithm are illustrated with the following example.

**Example1:** Consider the network given in Figure 3.1 with  $U = 5$ . Each edge is labeled with its cost and delay.

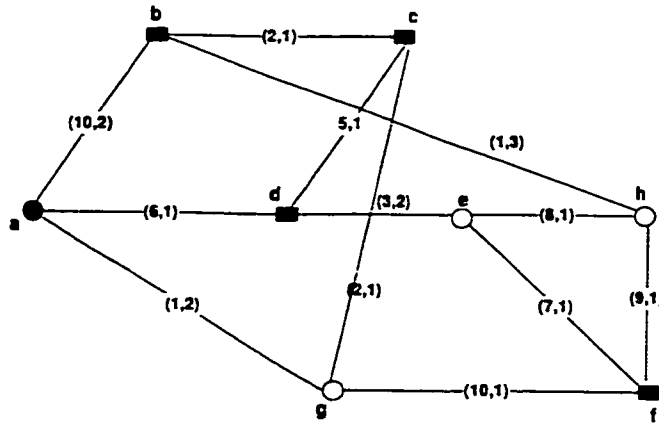


Figure 3.1: The graph of example 1.

**Step 1:** Compute all-pairs constrained least cost paths.

$$s = a, D = \{b, c, d, f\}$$

$cost(\Pi_{v,w}) =$  least cost constrained path from  $v$  to  $w$ .

$delay(\Pi_{v,w}) =$  path delay along least cost constrained path from  $v$  to  $w$ .

From Figure 3.1,

1.  $cost(\Pi_{a,b}) = 5, delay(\Pi_{a,b}) = 4, \Pi_{a,b} = [a, g, c, b]$
2.  $cost(\Pi_{a,c}) = 3, delay(\Pi_{a,c}) = 3, \Pi_{a,c} = [a, g, c]$
3.  $cost(\Pi_{a,d}) = 6, delay(\Pi_{a,d}) = 1, \Pi_{a,d} = [a, d]$



4.  $cost(\Pi_{a,f}) = 11, delay(\Pi_{a,f}) = 3, \Pi_{a,f} = [a, g, f]$
5.  $cost(\Pi_{b,c}) = 2, delay(\Pi_{b,c}) = 1, \Pi_{b,c} = [b, c]$
6.  $cost(\Pi_{b,d}) = 7, delay(\Pi_{b,d}) = 2, \Pi_{b,d} = [b, c, d]$
7.  $cost(\Pi_{b,f}) = 10, delay(\Pi_{b,f}) = 4, \Pi_{b,f} = [b, h, f]$
8.  $cost(\Pi_{c,d}) = 5, delay(\Pi_{c,d}) = 1, \Pi_{c,d} = [c, d]$
9.  $cost(\Pi_{c,f}) = 12, delay(\Pi_{c,f}) = 2, \Pi_{c,f} = [c, b, h, f]$
10.  $cost(\Pi_{d,f}) = 10, delay(\Pi_{d,f}) = 3, \Pi_{d,f} = [d, e, f]$

Then a closure graph is constructed where edges are labeled with the corresponding cost and delay of the enumerated least costs paths. The closure graph from Step 1 is shown in Figure 3.2.

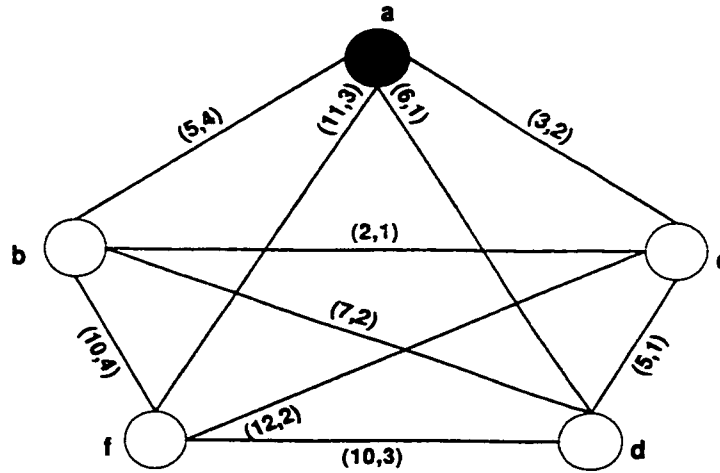


Figure 3.2: The closure graph on  $D = \{b, d, c, f\}$  with  $U = 5$  and  $s = a$ .

**Step 2:** Construct a constrained spanning tree of the closure graph. A greedy approach is used to add edges to a subtree of the constrained spanning tree until all the destination nodes are covered. The following two functions are defined for each path:

$$f_{CD}(v, w) = \frac{\text{cost}(\Pi_{v,w})}{U - (\text{delay}(\Pi_{s,v}) + \text{delay}(\Pi_{v,w}))} \quad (3.1)$$

where  $\text{delay}(\Pi_{s,v}) = \text{path delay from } s \text{ to } v \text{ in the tree.}$

$$f_C(v, w) = \text{cost}(\Pi_{v,w}) \quad (3.2)$$

Continuing with the same example,  $s = a, D = \{b, c, d, f\}$  and from Equation 3.1,

$$f_{CD}(a, c) = \frac{3}{5-(0+3)} = 1.5$$

$$f_{CD}(a, d) = \frac{6}{5-(0+1)} = 1.5$$

$$f_{CD}(a, f) = \frac{11}{5-(0+3)} = 5.5$$

$$f_{CD}(a, b) = \frac{5}{5-(0+4)} = 5$$

**Step 3:** Expand the edges of the constrained spanning tree into the constrained lowest paths that they represent.  $f_{CD}$  and  $f_C$  give rise to two source-based heuristics,  $CST_{CD}$ , and  $CST_C$ , respectively. The algorithm constructs a spanning tree using  $f_{C,D}$  and  $f_C$  as shown in Figures 3.3 and 3.4 respectively. Paths are added in increasing order of their cost until all destinations are covered.

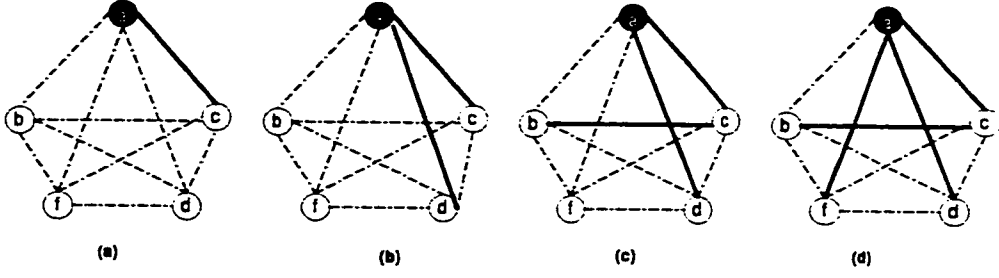


Figure 3.3: (a)-(d) The four stages in constructing the spanning tree using  $f_{CD}$ .

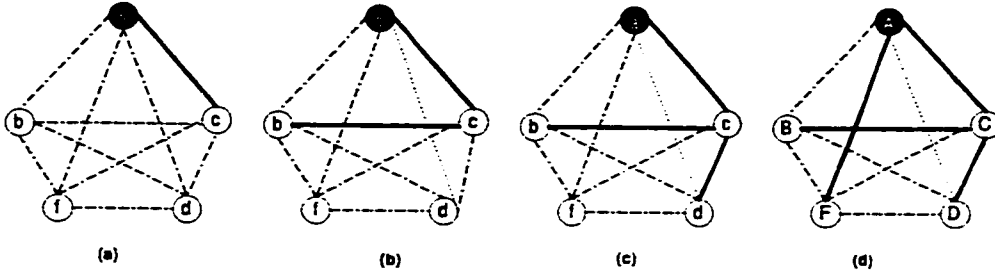


Figure 3.4: (a)-(d) The four stages in constructing the spanning tree using  $f_C$ .

$CST_{CD}$  uses the selection function  $f_{CD}$ , which explicitly uses both cost and delay in its functional form. It tries to choose low cost paths, but modulates the choice by trying to pick edges that maximize the residual delay as stated in Equation 3.1. This increases the chances of extending the path through this edge, and beyond to another destination. The idea is to reduce the cost of the tree through path sharing. Moreover, this heuristic has a tendency to optimize on delay. It may find paths with delays far lower than  $U$ , at the expense of added cost to the tree. Figure 3.5 shows the constrained spanning tree using  $CST_{CD}$ .  $CST_C$ , minimizes  $f_C$ , thereby trying to construct the lowest cost tree if possible, while meeting the delay bound. Figure 3.6 shows the constrained spanning tree using  $CST_C$ .

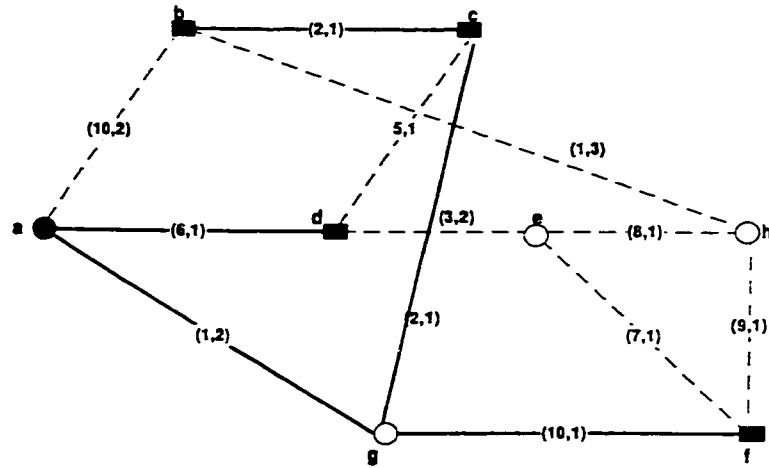


Figure 3.5: The constrained spanning tree using  $CST_{CD}$ .

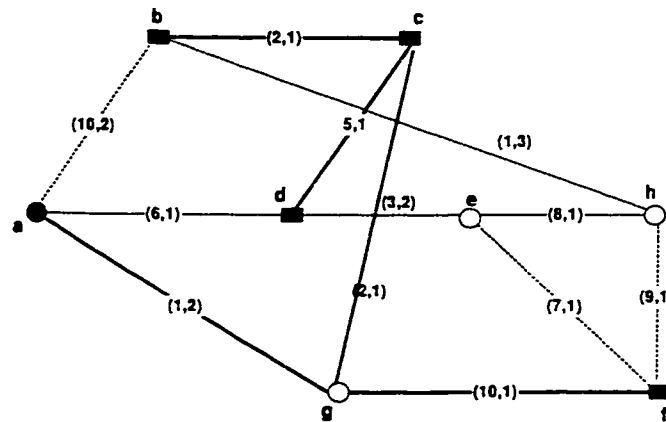


Figure 3.6: The optimal constrained Steiner tree: cost = 20 with  $U = 5$ .

Another heuristic algorithm called DDBMA (Dynamic Delay Bounded Multicast Algorithm) is reported by Hac [18] to construct a minimum cost multicast tree. The algorithm sets variable delay bounds on destinations and can be used to handle the network cost optimization goal; i.e., minimizing the total cost (total bandwidth utilization) of the tree. This algorithm can also be used to handle a dynamic delay-bounded minimum Steiner tree which is accomplished by updating the existing multicast tree when a destination node need to be added or deleted.

The DDBMA consists of four major steps:

1. Construct a minimum delay Steiner tree ( $T_0$ ) using Dijkstra's shortest-path algorithm with respect to the multicast source (Figure 3.7).
2. Iteratively refine the initial tree ( $T_0$ ) for lower cost using Path Replacement Technique.
3. Check for destination node either joining or leaving.
4. Update existing multicast tree if there are any requests to join or leave.

The algorithm terminates when there is no more requests. Consider the same network as shown in Figure 3.1. An important part of this algorithm is the delay-bounded path replacement iterative improvement strategy. Path replacement in the DDBMA means that a path in the tree  $T_i$  is replaced by another new path which is not in tree  $T_i$ , resulting in a new tree configuration  $T_{i+1}$  of a lower cost. Figure 3.8

shows the path replacement technique. The final tree from this algorithm is the same as that given in Figure 3.6.

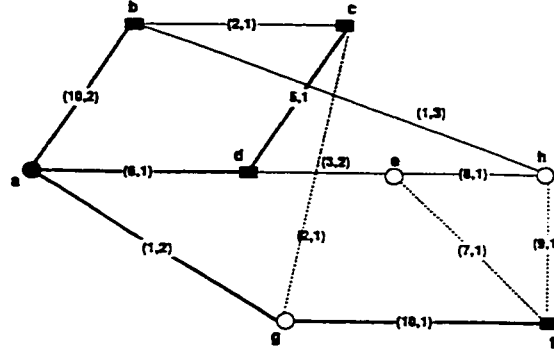


Figure 3.7: Minimum delay Steiner tree: cost = 32 with delay  $U = 5$ .

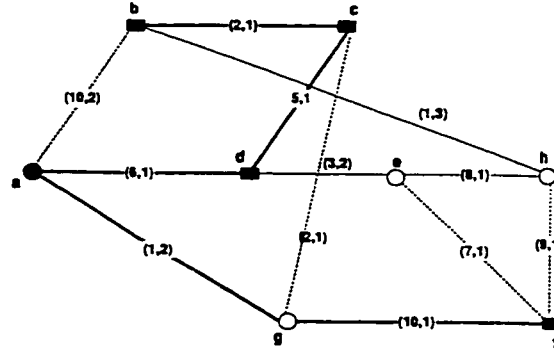


Figure 3.8: Example of Path replacement: path  $ab$  in Figure 3.7 is replaced by path  $bc$ . Cost = 24 with delay  $U = 5$ .

Evolutionary algorithms are robust heuristics for many hard optimization problems [1]. An orthogonal genetic algorithm is proposed by Zhang [12] for the design of delay constraint minimum cost multicast tree. A tree is encoded as a binary string. The edges of the graph are numbered from 1 to  $n$ . Thus any multicast tree  $T$  can

be represented as a binary String  $(e_1, e_2, \dots, e_n)$  where

$$e_i = \begin{cases} 1 & \text{if edge } i \text{ is included in the multicast tree } T \\ 0 & \text{otherwise} \end{cases}$$

Two fitness vectors are defined;  $f_1$  measures the cost of the multicast tree and  $f_2$  measures how tight the delay constraints are fulfilled. Thus a multicast tree is good if  $f_1$  is small (i.e., a small cost) and  $f_2$  is zero (i.e., the delay constraints are fulfilled). Given two multicast trees  $T_1$  and  $T_2$  with respective fitness vectors  $(C_1, D_1)$  and  $(C_2, D_2)$ ,  $T_1$  is better than  $T_2$  if and only if

1.  $D_1 < D_2$  or
2.  $D_1 = D_2$  and  $C_1 < C_2$ .

In this manner, the cost of the multicast tree is minimized enforcing the delay constraint. In genetic algorithms, crossover and mutation are adopted to search the solution space. The subgraph obtained from these operations may not be a tree connecting the source node to all the destination nodes. Thus a graph search method has been applied to find a spanning tree from the subgraph and then connect the unconnected destination nodes to the tree. This is called as Check and Repair Operation.

An orthogonal design has been used into the crossover operation [10]. The orthogonal design provides a series of orthogonal arrays for different number of factors

and a different number of levels.  $L_m(n^k)$  denotes an orthogonal array for  $k$  factors and  $n$  levels, where “ $L$ ” denotes a Latin square and  $m$  is the number of combinations of levels to be tested. The following table shows an example of orthogonal array  $L_4(2^3)$  for three factors and two levels. There are four combinations of factor Levels.

Combination	Factor 1	Factor 2	Factor 3
1st	1	1	1
2nd	1	2	2
3rd	2	1	2
4th	2	2	1

An example is shown below, in which the orthogonal array  $L_4(2^3)$  is used to sample the genes of two parents for crossover. Since  $L_4(2^3)$  has three factors, each parent string is divided into three substrings.

Example 2: Consider two Parents P1: 11001111 and P2: 00101001

$A_{1,1}$	$A_{1,2}$	$A_{1,3}$	$A_{2,1}$	$A_{2,2}$	$A_{2,3}$
110	011	11	001	010	01

Now sample the genes from the 2 parents to produce 4 binary strings based on  $L_4(2^3)$ .

$$o_1 = (A_{1,1}, A_{1,2}, A_{1,3}) = (1 \ 1 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1)$$

$$o_2 = (A_{1,1}, A_{2,2}, A_{2,3}) = (1 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1)$$

$$o_3 = (A_{2,1}, A_{1,2}, A_{2,3}) = (0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1)$$

$$o_4 = (A_{2,1}, A_{2,2}, A_{1,3}) = (0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1)$$



Among these four binary strings, some are selected to be offsprings. In orthogonal crossover, one of the main design choices is the orthogonal array. In many traditional genetic algorithms, global selection of parents is needed, and this is a serious bottleneck in parallel implementation. In the orthogonal genetic algorithm, selection is performed only within the orthogonal crossover operation but global selection is not needed. Thus, the orthogonal genetic algorithm can search the solution space effectively and is well suited for parallel implementation and execution.

A fast heuristic, called QDMR, is proposed by Guo [19] for generating low-cost multicast routing trees. A salient feature of QDMR is that it dynamically adjusts its low-cost tree construction policy based on how far the current on-tree node is from violating the QoS delay bound. This QoS dependent (adaptive) tree construction, together with the capability to merge least delay paths into the low-cost tree in case of constrained delay requirements, lead to the following properties:

1. QDMR guarantees to find a feasible multicast tree if such a tree exists;
2. This delay-bounded multicast tree is very rapidly generated; and
3. The tree has low cost.

QDMR is based on the DDMC algorithm [20]. DDMC algorithm does not consider delay constraints. The QDMR extends the DDMC algorithm by overcoming this problem. The DDMC is based on two well-known algorithms: Prim and Dijkstra's [21]. Both algorithms are greedy in nature. The DDMC algorithm treats

each destination node as a new "source" after that node is added to the tree. Thus the final tree has a low-cost multicast tree. However, this may lead to some destination nodes violating the delay bound. The pruning Phase in QDMR overcomes this problem of violating the delay bound.

The multicast routing is formulated as an Integer Programming problem by using Path variables in [22]. An associated problem reduction property is then characterized to reduce the solution space. Moreover, a polynomial time column generation procedure is exploited to solve the associated linear programming relaxation with such solution space reduced. Therefore, a branch-and-price algorithm is derived to obtain the optimal integer solution (tree) for the problem.

The bounded shortest multicast algorithm, BSMA is another multicast algorithm that is considered the best in terms of tree cost [23]. BSMA iteratively replaces the edges in the tree until the tree cost can not be further reduced. BSMA uses  $K$ th Shortest path algorithm to find lower cost edges. The time complexity for BSMA is  $O(K V^3 \log V)$  where  $K$  may be very large in the case of large, densely connected networks, and this may lead to the difficulty of achieving acceptable running times.

Another algorithm is constrained adaptive ordering, CAO [24]. In CAO, the constrained Bellman-Ford algorithm is used to connect one group member at a time to the source. After each run of the constrained Bellman-Ford algorithm, the unconnected member with the minimum cost constrained path to the source is chosen and is added to the existing subtree. The cost of the links in the existing

subtree is set to zero. CAO is always capable of constructing a constrained multicast tree, if one exists, because of the nature of the breadth-first search conducted.

A variant of the Steiner tree problem is proposed by Haberman [25]. A set of constraints namely delay and delay variation along the paths of the final tree have been imposed in this problem. The authors named this problem as *delay and delay variation constrained Steiner tree (DVCST) problem*.

The minimum Steiner tree heuristic proposed by Waters [26] to be semi-constrained, because it uses the maximum end-to-end delay from the source to any node in the network as the delay constraint. Thus, this constraint is not related directly to the application's QoS constraints, and that, depending on the network delays, this internally computed constraint may be too strict or too lenient as compared to the QoS requirements of the application. The heuristic then constructs a broadcast tree that does not violate the internal delay constraint. Finally, the broadcast is then pruned beyond the multicast nodes. In [27], the SC algorithm has been modified and denoted as MSC. Simulation results in [27] showed that MSC heuristic always performs better than the original heuristic with respect to tree costs and end-to-end delays.

### 3.3 Dynamic Steiner Tree Algorithms

In dynamic multicasting, destination nodes are joining and leaving the group during the communication period without any re-routing, and therefore the multicast tree may grow too large. A re-routing can produce an optimal cost tree, but is undesirable as it would cause disruption in transmission of continuous media. Thus, in [28], a centralized heuristic approach for dynamic multicast routing is proposed. The motivation is to minimize the total cost of Steiner Tree for the whole duration of a session without re-routing. Thus, this algorithm tries to minimize the spanning tree cost for the whole duration of a session, instead of individual tree cost for every time instant. It is assumed that the duration of participation of every member is known at the time of joining.

A non-rearrangeable Virtual Trunk Dynamic Multicast (VTDM) algorithm is proposed in [5] and is based on the following principle:

*“In the dynamic multicast routing problem, if a static multicast algorithm is applied to re-construct the tree for each request, some nodes and links may be frequently used. If a dynamic multicast routing can use these nodes and links, then it could conceivably construct low-cost trees matching the cost-performance of the static multicast algorithm”.*

In order to identify such nodes and links, the authors define the notion of a *Virtual Trunk (VT)*, which is a tree of the underlying network. The nodes of the

$VT$  are determined using a weight function that associates a positive real number with each node. The  $VT$ , for a given connection, is constructed by applying the KMB algorithm to a selected set of nodes [4]. It is then used as a template for modifying the tree in response to changes in group membership. New nodes are attached to the multicast tree by determining the least-cost path connecting the nodes to the Virtual Trunk. Nodes deletions are handled as greedy.

An efficient multicast routing algorithm for delay sensitive applications with dynamic membership is proposed by Hong [6]. The proposed algorithm utilizes an optimization technique called Lagrangian relaxation method. A source-based optimal dynamic multicast routing algorithm is proposed in [29], which satisfies the network conditions of delay constraints, cost minimization and adapts to a dynamic network events. An algorithm called CRCMD (Controlled Rearrangement for Constrained Dynamic Multicasting) for on-line update of multicast trees to adjust to changes in group membership is proposed in [7]. This algorithm is based on a concept called *Quality Factor (QF)* that represents the usefulness of a portion of the multicast tree to the overall multicast session.

An algorithm named as SBPT (Shortest Best Path Tree) is presented by Fujinoki [30]. This algorithm establishes and maintains dynamic multicast trees, maximize the bandwidth to be shared by multiple receivers and simultaneously guarantees the shortest paths for each receiver node. The SBPT algorithm is a distributed algorithm with a cost in the same order as the sum of the shortest path tree (SPT)

and Greedy Algorithm.

### 3.4 Conclusion

In this chapter, literature related to multicast routing algorithms has been reviewed. The most popular algorithms for unconstrained multicast routing problem are KMB [14] and RPM [31, 32]. Some algorithms for delay-constrained multicast routing algorithms are KPP [11], CAO [24], BSMA [23], SC [26], MSC [27]. BSMA is proved to be best and is within 7% of the optimal for symmetric and small networks [27]. For Dynamic Multicast Routing algorithms, Greedy [14], VTDM [5], and CRCDM [7] are the most popular. In short, this chapter covers different approaches that have been taken in the multicast routing problems and the objectives that they optimize.

## Chapter 4

# Tabu Search Based Multicast Tree Design

In Chapter 3, different heuristics for multicast routing algorithms were reviewed. These heuristics are applied only either to the minimum Steiner tree (MST) problem or the delay constrained minimum Steiner tree (CMST) problem. To the best of our knowledge, no algorithm has been proposed for the multiobjective Steiner tree optimization (MOST) problem. A tabu search based heuristic is proposed in this thesis, that can easily be applied to both minimum Steiner tree and delay Constrained minimum Steiner tree problem. Further, a fuzzy logic based tabu search algorithm is proposed for multiobjective Steiner tree optimization problem. Thus, in our scheme, we optimize four cost parameters in the multicast tree design: cost of the tree, end-to-end delay of the tree, delay variance of the tree and number of

Steiner nodes in the tree.

In this chapter, the proposed scheme and implementation details are discussed. This chapter is organized as follows. Section 4.1, some of the theory of tabu search is briefly explained. Section 4.2 describes proposed scheme for delay constrained minimum Steiner tree problem followed by proposed scheme for unconstrained minimum Steiner tree problem in Section 4.3. Section 4.4 describes multiobjective minimum Steiner tree problem. A conclusion is given in Section 4.6.

## 4.1 Tabu Search

The number of possible multicast trees in a computer network of a moderate size is extremely large. Further because of the constrained nature of the problem and the various cost parameters, it is not clear what constitutes the best tree. Modern iterative heuristics such as tabu search have been found effective in tackling this category of problems which have an exponential and noisy search space with numerous local optima [1]. These iterative algorithms are heuristic search methods which perform a nondeterministic but intelligent walk through the search space. In this chapter, we present a Tabu Search algorithm to find a multicast tree satisfying the given Quality of Service (QoS).



### 4.1.1 Overview of Tabu Search

Tabu Search (TS) was introduced by Fred Glover [33, 34] as a general iterative metaheuristic for solving combinatorial optimization problems. Tabu Search is conceptually simple and elegant. It is a form of local neighborhood search. Each solution  $S \in \Omega$  has an associated set of neighbors  $N(S) \subseteq \Omega$  where  $\Omega$  is the set of feasible solutions. A solution  $S' \in N(S)$  can be reached from  $S$  by an operation called a move to  $S'$ . TS moves from a solution to its best admissible neighbor, even if this causes the objective function to deteriorate. To avoid cycling, solutions that were recently explored are declared forbidden or tabu for a number of iterations. The tabu Status of a solution is overridden when certain criteria (aspiration criteria) are satisfied. Intensification and diversification strategies are used to improve the search. In the former case, the search is accentuated in promising regions of the feasible domain. In the latter case, an attempt is made to consider solutions in a broad area of the search space [1]. To produce good results, any implementation of TS must be engineered to suite the structure of the problem at hand [35]. The general Tabu Search algorithm is given in Figure 4.1.

### Algorithm Short-Term-TS

$\Omega$  : Set of feasible solution  
 $S$  : Current Solution  
 $S^*$  : Best admissible solution  
 $Cost$  : Objective function  
 $N(S)$  : Neighborhood of solution  $S$   
 $V^*$  : Sample of neighborhood solutions  
 $T$  : Tabu list  
 $AL$  : Aspiration Level

**Begin**

1. Start with an initial feasible solution  $S \in \Omega$ .
2. Initialize tabu lists and aspiration level.
3. For fixed number of iterations Do
4.     Generate neighbor solutions  $V^* \subset N(S)$ .
5.     Find best  $S^* \in V^*$ .
6.     If move  $S$  to  $S^*$  is not in  $T$  Then
7.         Accept move and update best solution.
8.         Update tabu list and aspiration level.
9.         Increment iteration number.
10.     Else
11.         If  $Cost(S^*) < AL$  Then
12.             Accept move and update best solution.
13.             Update tabu list and aspiration level.
14.             Increment iteration number.
15.         End If
16.     End If
17. End For

**End**

Figure 4.1: Algorithmic description of a short-term Tabu Search (TS) [1].

## 4.2 Delay Constrained Minimum Steiner Tree (CMST)

### Design

This problem is also called as QoS Driven Multicast Tree Generation Problem. Our Algorithm assumes that sufficient global information is available to the source, i.e., the source node of the network has complete information regarding all network links to construct a multicast tree.

#### 4.2.1 Initial Solution

The algorithm starts with an initial feasible solution  $S \in \Omega$  built in a greedy fashion as follows: A minimum cost delay constraint Steiner tree is constructed using Dijkstra's shortest-path algorithm starting from the source. This results in a set of superpaths. A superpath is defined as a set of edges whose starting and ending nodes are the root of the tree and any node in  $\{s\} \cup D$ . We call this tree as sink tree of source  $s$ . To illustrate how CMST works, we consider a network as shown in Figure 4.2, here  $s = \{A\}$  and  $D = \{B, C, D, F\}$ .

The sink tree for source  $s$  is shown in Figure 4.3. The solution encoding is path based, where a solution is encoded as an array of  $k$  elements where each element is a superpath representing a branch of the multicast tree and  $k = |D|$  i.e., cardinality of the set  $D$ , which contains the members of the multicast group.

The encoding for the initial solution corresponding to the sink tree of Figure 4.3

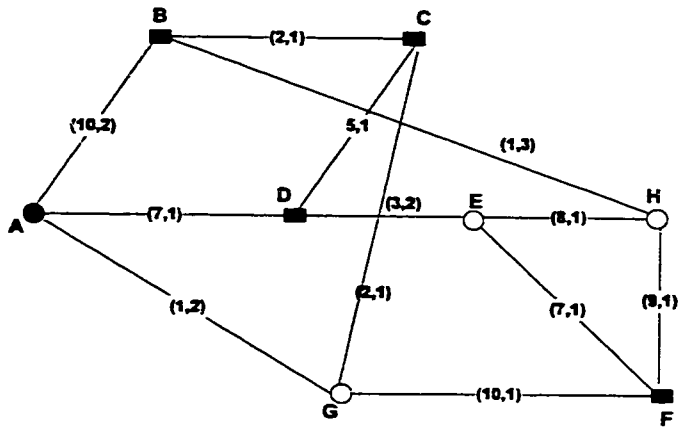


Figure 4.2: An example network.

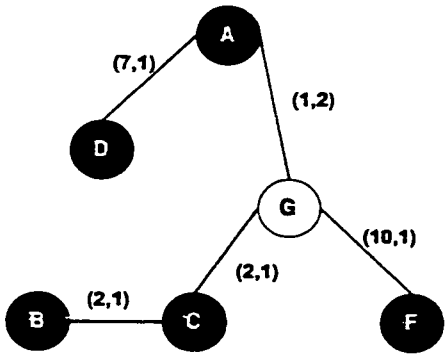


Figure 4.3: Sink tree for source A, cost = 22 with  $U = 5$ .

is as follows

0	1	2	3	
$A, D$	$A, G, F$	$A, G, C$	$A, G, C, B$	$Cost = 22$

#### 4.2.2 Neighborhood Solutions

For generating neighbors, we choose a neighborhood structure based on “delete and add” operations. To reduce the size of the solution space to be searched by Tabu, we construct a sink tree for each destination using Dijkstra’s Shortest Path Algorithm. The destination is the root of the tree and the remaining nodes out of  $\{s\} \cup D$  are the destinations of the new sink tree. Figure 4.4 shows the sink trees generated for the network of Figure 4.3.

Each iteration begins by generating a set  $V^*$  of neighboring solutions. In our algorithm, the set  $V^*$  is dynamic, i.e., it varies from one iteration to another. At each iteration, we randomly delete one superpath from the encoding of current solution and then generate different feasible solutions by adding superpaths from one of the destinations sink trees of Figure 4.4.

Among the neighbors, the one with the best cost is selected, and considered as new current solution for the next iteration. For example, the two new trees of Figure 4.4 are shown in Figure 4.5 and are encoded as follows

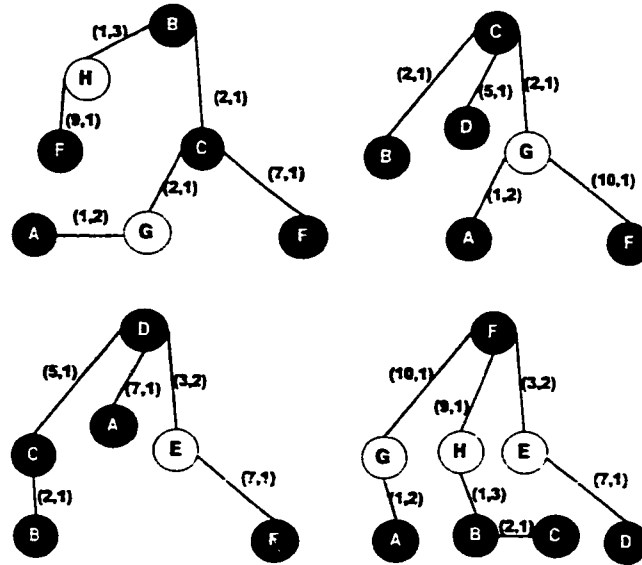


Figure 4.4: Sink trees for destinations  $D = \{B, C, D, F\}$ .

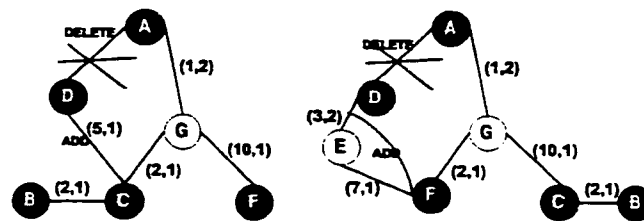


Figure 4.5: Two possible neighbors from current solution.

	0	1	2	3	Cost
Figure 4.6(a) Encoding	$D, C$	$A, G, F$	$A, G, C$	$A, G, C, B$	20
Figure 4.6(b) Encoding	$D, E, F$	$A, G, F$	$A, G, C$	$A, G, C, B$	$25 + \text{Penalty}$

Thus, the new multicast tree of Figure 4.6(a) is selected for the next iteration and considered as new current solution. As mentioned above, we randomly delete one superpath from the current solution in the next iteration. Thus, for all candidate solutions in the current iteration, it is guaranteed that at least one solution will give a multicast tree. It might happen that some of the trees violate delay constraint; in that case we assign an extra penalty by increasing its cost, so that it is less likely to be accepted in the candidate list as shown in the tree of Figure 4.6(b).

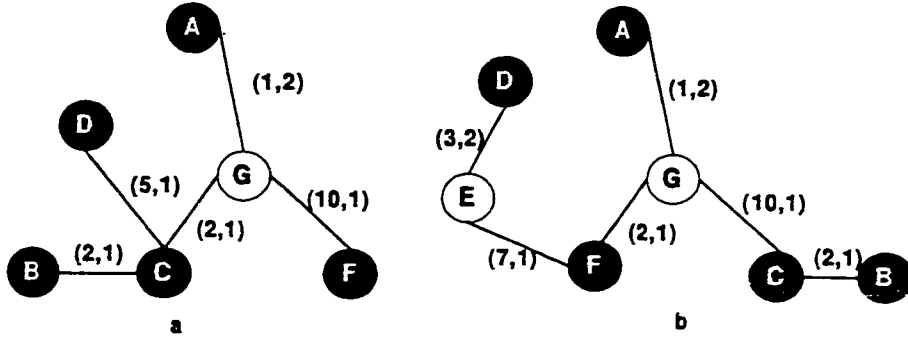


Figure 4.6: Final solutions from current solution.

### 4.2.3 Tabu Moves

If a superpath is deleted at iteration  $i$ , then reintroducing the same superpath in an add operation is tabu. The tabu list size is set to 7.

#### **4.2.4 Aspiration Criterion**

As is common in TS, a tabu status of a move can be overridden if implementing it results in a better cost. This means that a tabu superpath in this case will be inserted anyhow.

#### **4.2.5 Termination Rule**

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 tabu search algorithm converges within a maximum of 500 iterations. The choice of a maximum number of iteration is a function of network and group size. This issue has not been investigated in this work but could be future research work.

### **4.3 Unconstrained Minimum Steiner Tree (MST)**

#### **Design**

The algorithm explained above can also be used for Steiner tree problem also called as multicast tree generation problem. The only change in the initial solution where a minimum cost Steiner tree is constructed using Dijkstra's shortest path algorithm without delay constrained and also there is no penalty in the cost because there is no delay violation during the formulation of the candidate list solutions.



## 4.4 Multiobjective Steiner Tree Optimization (MOST)

### Design

In earlier chapters we mentioned that there are four criteria to be optimized with respect to the overall solution. These criteria are cost of the tree, maximum end-to-end delay, number of Steiner nodes and delay variance. 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 the other three criteria. Let  $C_i$ ,  $D_i$ ,  $H_i$ , and  $V_i$  be the cost, delay, maximum Steiner nodes and variance of a multicast tree solution  $S_i$  respectively. A solution  $S_i$  is said to dominate a solution  $S_j$ , denoted as  $S_i \prec S_j$  iff

$$C_i < C_j; D_i < D_j; H_i < H_j; V_i < V_j$$

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 prioritized against each other. For example cost would be number 1 objective, then delay, then Steiner nodes, then delay variance. Then a solution  $S_i$  is better than  $S_j$  if

$$C_i < C_j,$$

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

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

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

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

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

where  $W_C$ ,  $W_D$ ,  $W_H$ , and  $W_V$  are the weights associated with cost, delay, Steiner nodes, and delay variance respectively. Usually  $W_C + W_D + W_H + W_V = 1$ . The problem with this approach is that it needs a careful tuning with respect to each objective that needs to be optimized. Good solutions can only be obtained if tuning has been made.

During the multicast tree design process, some desirable objectives, such as the delay, can only be imprecisely estimated. To efficiently deal with such situation, Fuzzy logic can be considered. Fuzzy logic provides a rigorous algebra for dealing with imprecise information. Further, it is a convenient method of combining conflicting objectives and expert human knowledge.

Fuzzy set theory has been recently applied in many areas of science and engineering [36, 37, 38, 39]. But there are very few applications of fuzzy logic in data networking which are limited to specific problems.

In fuzzy logic, optimization of a vector-valued function is replaced by the optimization of a scalar function, which is constructed from levels of satisfaction of decision-makers by values of components of a vector-function. In practice, this approach is proven to be useful for finding of compromised solutions in different applications.

Another reason to consider fuzzy logic approach is its ability of treatment of uncertainties in network state information. Fuzzy logic provides a convenient framework for representation of such knowledge. The desirable properties of multicast trees are more naturally described in linguistic terms, which constitute the basis of fuzzy logic.

A fuzzy set  $A$  of a universe of discourse  $X$  is defined as  $A = \{(x, \mu_A(x)) | \text{all } x \in X\}$ , where  $\mu_A(x)$  is a membership function of  $x \in X$  being an element in  $A$ . Membership of data in a fuzzy set is defined using values in the range  $[0,1]$ . A value of 1 indicates full membership in the set. Figure 4.7 shows one example of a membership function.

For the problem under consideration, four objective functions are combined using fuzzy logic to characterize a good multicast tree, as depicted in Figure 4.8. The fuzzy subset of good multicast trees is defined by the following Fuzzy logic rule:

**IF** a tree has *small number of Steiner nodes* **AND** *low maximum end-end delay*  
*AND low delay variation* **AND** *low cost*, **THEN** it is a good tree

According to the andlike/orlike ordered-weighted-averaging logic [40], the above

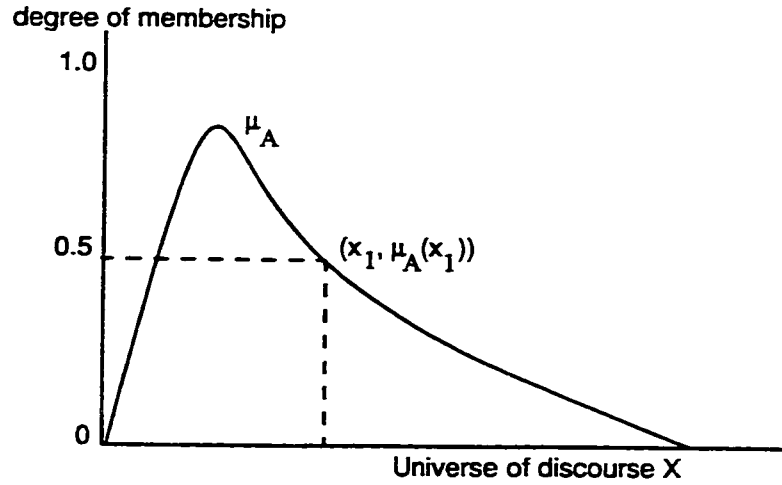


Figure 4.7: Membership function for a fuzzy set A.

rule evaluates to the following:

$$\mu(x) = \gamma \times \min(\mu_i(x)) + (1 - \gamma) \times \frac{1}{4} \sum_{i=1}^4 \mu_i(x). \quad (4.1)$$

where  $\mu(x)$  is the membership value for solution  $x$  in the fuzzy set *good tree* and  $\beta$  is a constant in the range  $[0,1]$ . Here,  $\mu_i$  for  $i = \{1,2,3,4\}$  represents the membership values of solution  $x$  in the fuzzy sets *small number of Steiner nodes*, *low delay*, *low delay variance*, and *low cost*, respectively. The solution which results in the maximum value for Equation 4.1 is reported as the best solution found by the TS algorithm. Below we will see how to get the membership functions for the four criteria mentioned above.

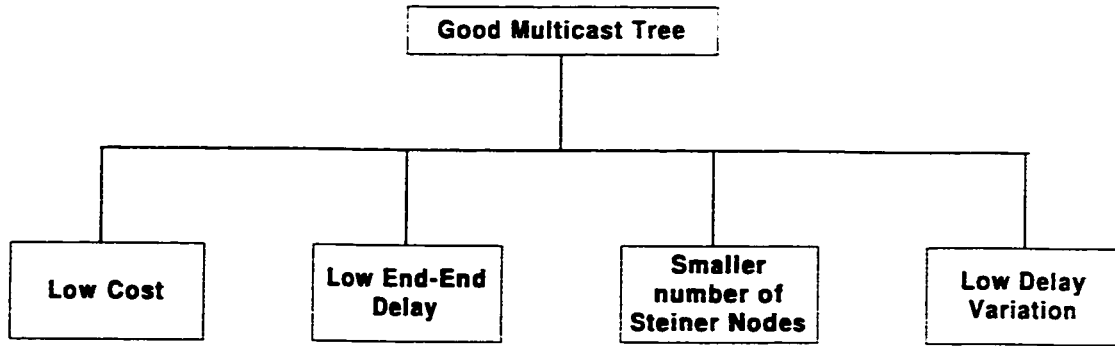


Figure 4.8: Basic components for a good topology.

#### 4.4.1 Membership Function for Cost

First, we determine two extreme values for Cost, i.e., the minimum and maximum values. The maximum value, "TCostMax", is found by using Dijkstra's shortest path algorithm with respect to delay. The minimum value, "TCostMin", is only found by using optimal minimum Steiner tree algorithm using Branch and Bound techniques, which are not suitable to apply due to the exhibition of very high time complexity. So, approximate minimum value, "TCostmin", can be found by taking 75% of the cost found by using Dijkstra's shortest path algorithm with respect to cost which is the initial solution of our proposed tabu search algorithm. The cost is normalized with respect to "TCostMax". The shape of the membership function is shown in Figure 4.9. The membership value for the normalized cost is then compared

using Equation 4.2.

$$\mu_1(x) = \begin{cases} 1 & \text{if } TCost \leq TCostMin \\ \frac{TCostMax - TCost}{TCostMax - TCostMin} & \text{if } TCostMin \leq TCost \leq TCostMax \\ 0 & \text{if } TCostMax \leq TCost \end{cases} \quad (4.2)$$

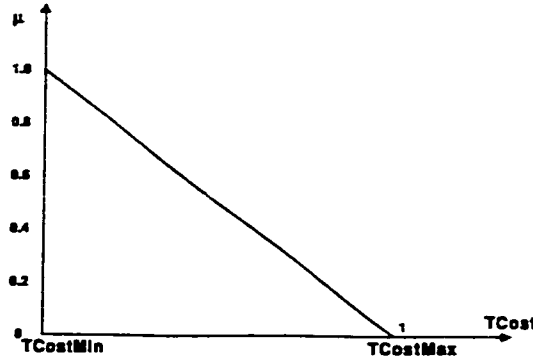


Figure 4.9: Membership function for cost.

#### 4.4.2 Membership Function for End-to-End Delay

First, we determine two extreme values for maximum end-to-end delay, i.e., the minimum and maximum values. The maximum value, “TDelayMax”, is found by using Dijkstra’s shortest path algorithm with respect to cost. The minimum value, “TDelayMin”, is found by using Dijkstra’s shortest path algorithm with respect to delay. The shape of the membership function is shown in Figure 4.10. The

membership value for the normalized delay is then compared using Equation 4.3.

$$\mu_2(x) = \begin{cases} 1 & \text{if } TDelay \leq TDelayMin \\ \frac{TDelayMax - TDelay}{TDelayMax - TDelayMin} & \text{if } TDelayMin \leq TDelay \leq TDelayMax \\ 0 & \text{if } TDelayMax \leq TDelay \end{cases} \quad (4.3)$$

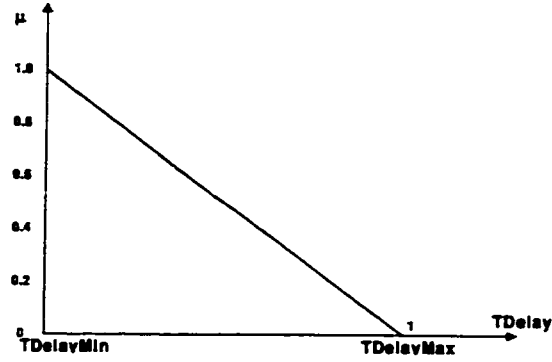


Figure 4.10: Membership function for end-to-end delay.

#### 4.4.3 Membership Function for Number of Steiner nodes

First, we determine two extreme values for number of Steiner nodes., i.e., the minimum and maximum values. The minimum value for the number of hop is 0, and the maximum value is taken to be maximum number of Steiner nodes found by either Dijkstra w.r.t. cost or Dijkstra w.r.t. delay. The shape of the membership function is shown in Figure 4.11. The membership value for the normalized Steiner nodes is

then compared using Equation 4.4.

$$\mu_3(x) = \begin{cases} 1 & \text{if } TS\_nodes \leq TS\_nodesMin \\ \frac{TS\_nodesMax - TS\_nodes}{TS\_nodesMax - TS\_nodesMin} & \text{if } TS\_nodesMin \leq TS\_nodes \leq TS\_nodesMax \\ 0 & \text{if } TS\_nodesMax \leq TS\_nodes \end{cases} \quad (4.4)$$

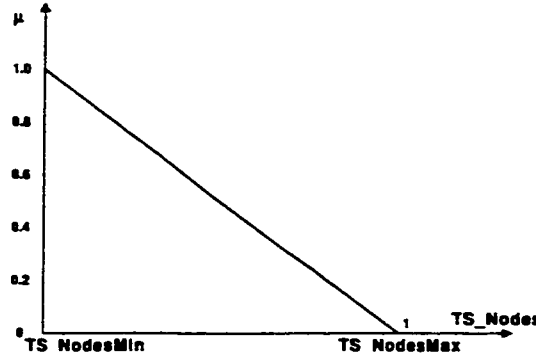


Figure 4.11: Membership function for Steiner nodes.

#### 4.4.4 Membership Function for Delay Variance

First, we determine two extreme values for delay variance, i.e., the minimum and maximum values. The maximum value, “TDVarMax”, is found by using Dijkstra’s shortest path algorithm with respect to cost. The minimum value, “TDVarMin”, is found by using Dijkstra’s shortest path algorithm with respect to delay. The shape of the membership function is shown in Figure 4.12. The membership value for the



normalized delay variance is then compared using Equation 4.5.

$$\mu_4(x) = \begin{cases} 1 & \text{if } TDvar \leq TDvarMin \\ \frac{TDvarMax - TDvar}{TDvarMax - TDvarMin} & \text{if } TDvarMin \leq TDvar \leq TDvarMax \\ 0 & \text{if } TDvarMax \leq TDvar \end{cases} \quad (4.5)$$

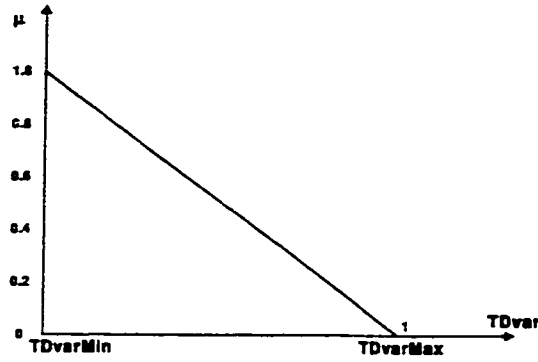


Figure 4.12: Membership function for delay variance.

#### 4.4.5 An Example

The algorithm for multiobjective optimization is similar to the one discussed in Section 4.2, except that new cost function, also called as goodness, is based on Equation 4.1. Consider again the network of Figure 4.2. Figure 4.13 shows the tree obtained by applying Dijkstra's shortest path algorithm to it with respect to delay.

The following are the extreme values used in membership functions that are obtained from Figure 4.13.

- Maximum Value of Cost i.e., "TCostMax" = 33

- Minimum Value of Delay i.e., “TDelayMin” = 3
- Minimum Value of Delay Variance i.e., “TDVarMin” =  $\text{Variance}(2,2,1,3) = 0.5$

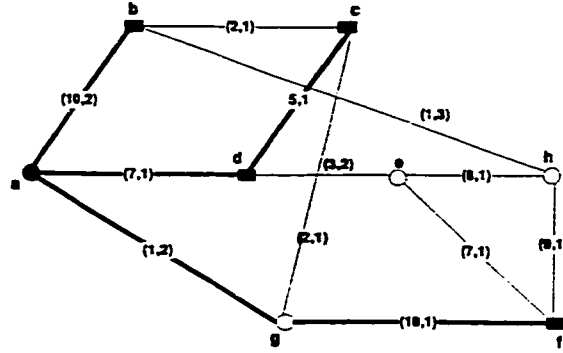


Figure 4.13: Minimum delay tree.

Figure 4.14 shows the tree obtained by applying Dijkstra's shortest path algorithm with respect to Delay. This is used as the initial solution for fuzzy tabu search based algorithm.

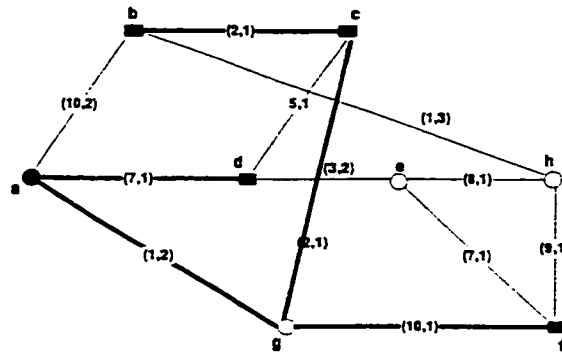


Figure 4.14: Minimum cost tree.

The following are the extreme values used in the membership functions that are obtained from Figure 4.14.

- Minimum value of cost i.e., “TCostMin” =  $0.75 \times 22 = 16.5$
- Maximum value of end-to-end delay i.e., “TDelayMax” = 4
- Maximum value of delay Variance i.e., “TDVarMax” =  $\text{variance}(4,3,1,3) = 1.1875$

The only two extreme values are remaining “TS\_nodesMin” and “TS\_nodesMax”.

- Minimum value of number of Steiner nodes i.e., “TS\_nodesMin” = 0
- Maximum Value of number of Steiner nodes i.e., “TS\_nodesMax” =  $\max(1,1) = 1$

Now, the initial solution and neighborhood solutions are generated as the same procedure as described in Section 4.2. The cost function for the initial solution is calculated from Equation 4.1. The following are the numeric values for the membership functions of the initial solution.

From Equation 4.2  $\mu_1(x) = \frac{33-22}{33-16.5} = 0.67$

From Equation 4.3  $\mu_2(x) = \frac{4-4}{4-3} = 0.0$

From Equation 4.4  $\mu_3(x) = \frac{1-1}{1-0} = 0.0$

From Equation 4.5  $\mu_4(x) = \frac{1.1875-1.1875}{1.1875-0.5} = 0.0$ .

Thus goodness or the cost function is calculated using Equation 4.1 which gives  $\mu(x) = 0.08334$ .

Now, as mentioned in Section 4.2, Each iteration begins by generating a set  $V^*$  of neighboring solutions. In our algorithm, the set  $V^*$  is dynamic, i.e., it varies from one iteration to another. At each iteration, we randomly delete one superpath from the encoding of current solution and then generate different feasible solutions by adding superpaths from one of the destinations sink trees as shown in Figure 4.4. The two possible neighbors are shown in Figure 4.6. The goodness for Figure 4.6(a) is calculated as follows,

$$\mu_1(x) = \frac{33-20}{33-16.5} = 0.789$$

$$\mu_2(x) = \frac{4-4}{4-3} = 0.0$$

$$\mu_3(x) = \frac{1-1}{1-0} = 0.0$$

$$\mu_4(x) = 1 \text{ since } \mu_4(x) \leq TDVarMin.$$

Using Equation 4.1, we get  $\mu(x) = 0.2235$

Similarly, the goodness for Figure 4.6(b) is calculated as follows,

$$\mu_1(x) = \frac{33-25}{33-16.5} = 0.485$$

$$\mu_2(x) = 0.0 \text{ since } \mu_2(x) \geq TDDelayMax$$

$$\mu_3(x) = \frac{1-1}{1-0} = 0.0$$

$$\mu_4(x) = \frac{1.1875-0.6875}{1.1875-0.5} = 0.723$$

Again using Equation 4.1,  $\mu(x) = 0.177$

Among the neighbors, the one with the high goodness is selected, and considered as a new current solution for the next iteration. Thus, the solution represented by the tree of Figure 4.6(a) is selected as a new solution for the next iteration because

of high goodness. The tabu list size is set to be 15 and the aspiration criteria and termination rule are same as described in Section 4.2.

## 4.5 Time Complexity Analysis of Tabu Search

The most expensive step of our heuristic is the initial step where Dijkstra's shortest-path algorithm is used for generating sink trees for the source and the destinations. The worst time complexity of this step is  $O(M|V|^2)$  where  $M$  is the number of multicast members including the source and  $|V|$  is the number of nodes in the network. In tabu search, one iteration costs  $O(M)$ . Thus, for  $k$  iterations, the cost becomes  $O(Mk)$ . Thus the expected time complexity of proposed tabu search algorithm is  $O(Mk + M|V|^2)$ . The term  $Mk$  is usually much smaller than  $M|V|^2$ . Therefore, complexity of the algorithm is  $O(M|V|^2)$ . In comparison with other algorithms with respect to time complexity, our tabu search algorithm compares favorably with KPP and BSMA which have time complexities of  $O(U|V|^3)$  and  $O(K|V|^3 \log |V|)$  respectively. The time complexity of CAO is not known.

## 4.6 Conclusion

The multicast routing problem is an NP-hard problem. The number of possible multicast trees in a computer network even of medium size is extremely large. A tabu search algorithm technique is being proposed and described in this chapter

that is suitable for both unconstrained and constrained multicast routing problem.

For multiobjective multicast routing problem, a fuzzy logic is used in the cost function. Time complexity of the proposed algorithm is also explained in this chapter.

# Chapter 5

## Experiments and Simulation

### Results

In this chapter, algorithms discussed in Chapter 3 and Chapter 4 have been incorporated into our experimentation. The results of these experiments are also discussed. Moreover, we provide a comparison between different constrained and unconstrained multicast routing algorithms with tabu search is done. This chapter is organized as follows. Section 5.1 is the experimental setup. Section 5.2 gives simulation results for unconstrained steiner tree problem followed by simulation results for delay constrained steiner tree problem in Section 5.3. Section 5.4 gives the simulation results for multiobjective minimum Steiner tree problem. A conclusion is given in Section 5.5.

## 5.1 The Experimental Setup

Full duplex ATM networks with homogeneous link capacities of 155 Mbps were used in the experiments. The positions of the nodes are fixed in a rectangle of size  $4000 \times 2400 \text{ Km}^2$ , roughly the dimensions of the continental United States. ATM networks permit the applications to specify their own QoS requirements, and they allow cell multicasting in the physical layer. Thus, it was appropriate to comply with the ATM standards. A random generator [9] (based on Waxman's generator [4] with some modifications) was used to create links interconnecting the nodes. A multicast routing simulator MCRSIM [41] developed at North Carolina State University is used to generate random graphs as described in [9]. In this model  $n$  nodes are randomly distributed over a rectangular coordinate grid. Each node is placed at a location with integer coordinates. The Euclidean metric is then used to determine the distance between each pair of nodes. In this model, edges are introduced between pairs of nodes  $(u, v)$  with a probability that depends on the distance between them. The edge probability is given by

$$P(u, v) = \beta \exp \frac{-d(u, v)}{L\alpha} \quad (5.1)$$

where  $d(u, v)$  is the distance from node  $u$  to  $v$ ,  $L$  is the maximum distance between two nodes, and  $\alpha$  and  $\beta$  are parameters in the range  $(0, 1)$ . Larger values of  $\beta$  result in graphs with higher edge densities, while small values of  $\alpha$  increases the density



of short edges relative to longer ones.

Each node represented a non-blocking ATM switch, and each link had a small output buffer. The propagation speed through the links was taken to be two thirds the speed of light. The link propagation delay was dominant under these assumptions, and the queuing component of the link delay was not taken into account. The link delays were thus symmetric,  $delay(u, v) = delay(v, u)$ , because the link lengths, and hence the propagation delays, were symmetric.

For the multicast sources we used variable bit rate (VBR) video sources. These represented realistic, bursty, multimedia traffic sources. Any session traversing a link  $e$ , reserved a fraction of  $e$ 's bandwidth equal to the equivalent bandwidth of the traffic it generated. The link cost,  $cost(e)$ , was set equal to the reserved bandwidth on that link, because it is a suitable measure of the utilization of both the link's bandwidth and its buffer space. Therefore, the cost of a heavily utilized link was larger than the cost of a lightly utilized link. The link costs were dynamic, and varied as new sessions were established or existing sessions were torn down.

A link could accept sessions and reserve bandwidth for them until its cost, i.e., the sum of the equivalent bandwidth of the sessions traversing that link, exceeded 85% of the link's capacity; then it got saturated. This admission control policy allowed statistical multiplexing and efficient utilization of the available resources. More sophisticated admission control policies for real-time traffic exist, but the simple policy just described was sufficient for the purposes of our study of multicast

routing algorithms. A detailed study of admission control algorithms for real-time traffic can be found in [12].

In our experiments we used random networks with an average node degree of 4, which is close to the average node degree of the current Internet. The values for the parameters  $\alpha$  and  $\beta$  are 0.15 and 2.2 respectively. Figure 5.1 shows an example of a randomly generated 20-node network.

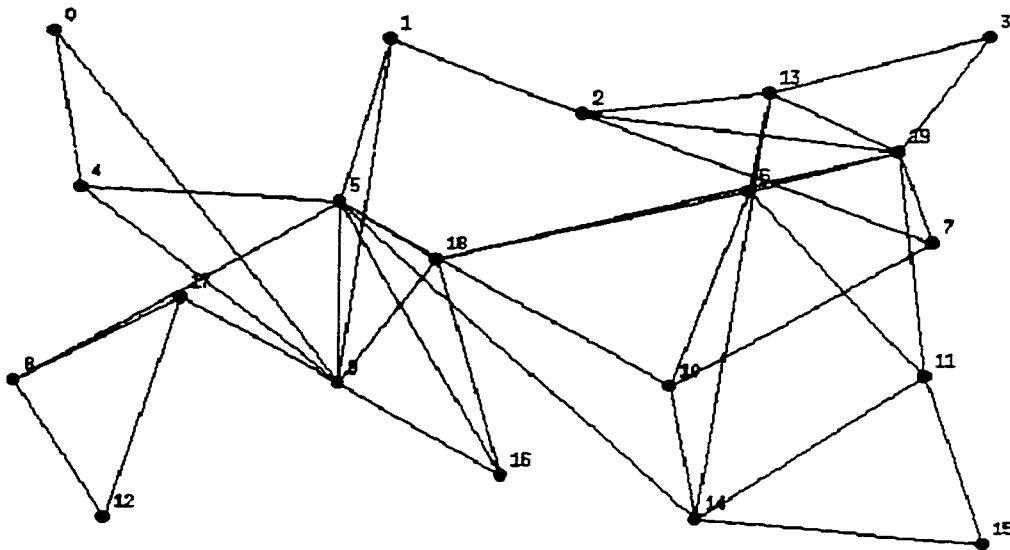


Figure 5.1: A randomly generated network, 20 nodes, average degree 4.

## 5.2 Simulation Results for Unconstrained Multicast Tree Design Problem

Many existing unconstrained Steiner tree heuristics including KMB [14], RPM [31, 32] have been implemented in MCRSIM. For each run of the experiment, a random set of links to interconnect the fixed nodes is generated. Random background traffic for each link is also generated. A random source node and a multicast group of randomly chosen destination nodes are selected. The equivalent bandwidth of each link's background traffic was a random variable uniformly distributed between minimum background traffic  $B_{min}$  and maximum background traffic  $B_{max}$ . As the range of the link loads, i.e., the difference between  $B_{min}$  and  $B_{max}$ , increased, the asymmetry of the link loads also increased, because the load on the forward link  $e = (u, v)$  is independent of the load on the backward link  $e' = (v, u)$ . The tabu search based algorithm was run on 20, 30, 40, 50, 60, 70, 80, 90, and 100 node random graph with  $\alpha = 0.15$  and  $\beta = 2.2$ . The  $B_{min}$  and  $B_{max}$  are assumed to be  $10Mbps$  and  $150Mbps$  respectively. The default link capacity is  $155Mbps$ . Following sets of experiments are carried out.

- Comparing tabu search with KMB [14].
- Comparing tabu search with RPM [31, 32].

In order to analyze the performance of the proposed tabu search algorithm, we compare it with KMB as well as with RPM. There are three sets of experiments. In the first set of experiments, tabu search is compared with KMB and RPM for symmetric networks. In the second set of experiments, tabu search is compared for asymmetric networks, and in the third set of experiment, a completely unloaded network is taken and kept adding multicast (MC) sessions and constructing the corresponding MC trees until the cumulative tree failure rate exceeded 15%. A MC session consisted of a random source node generating VBR video traffic with an equivalent bandwidth of 0.5 Mbps, and a MC group of randomly chosen destination nodes. The experiment was repeated with different MC groups.

### 5.2.1 Comparison of Tabu Search and KMB

In section 3.1, we have discussed KMB algorithm, which is the most widely used algorithm for unconstrained minimum steiner tree problem. Figures 5.2 and 5.4 show the cost comparison between KMB and Tabu under fixed group size of 10 and different networks for Symmetric Networks and Asymmetric Networks respectively. Figures 5.3 and 5.5 show the cost comparison between KMB and Tabu under different group size for symmetric networks and asymmetric networks respectively. Our proposed tabu search based heuristic was able to identify better trees for all Test Networks. Figure 5.6 shows the number of multicast sessions versus cost for a network of 20 nodes and group size of 5. The number of multicast sessions is the

same for both heuristics, and both algorithms are able to identify alternate paths for the saturated links. Tabu was even able to identify a better tree in terms of cost in almost all cases for different multicast sessions.

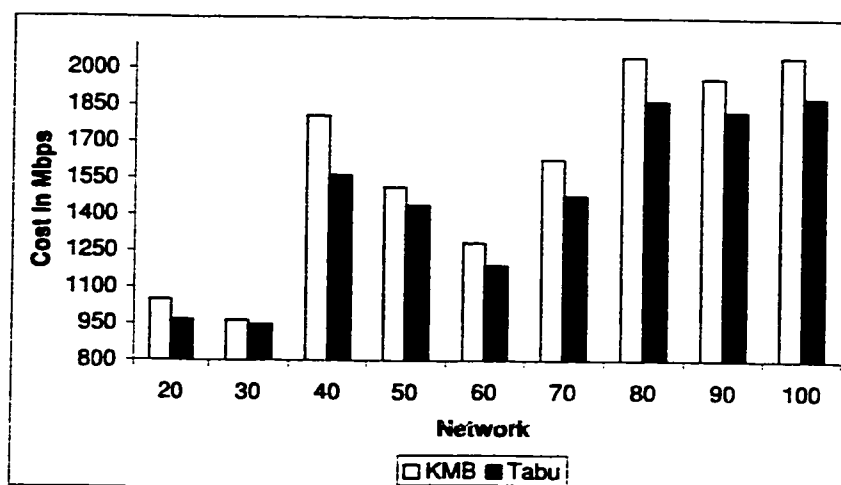


Figure 5.2: Symmetric load. Cost comparison of KMB and Tabu. Group size =10.

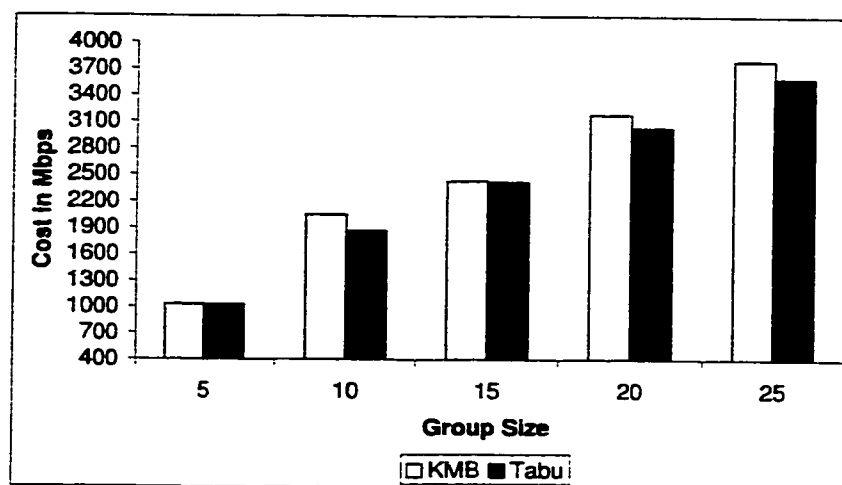


Figure 5.3: Symmetric load. Cost comparison of KMB and Tabu. Network size =80.

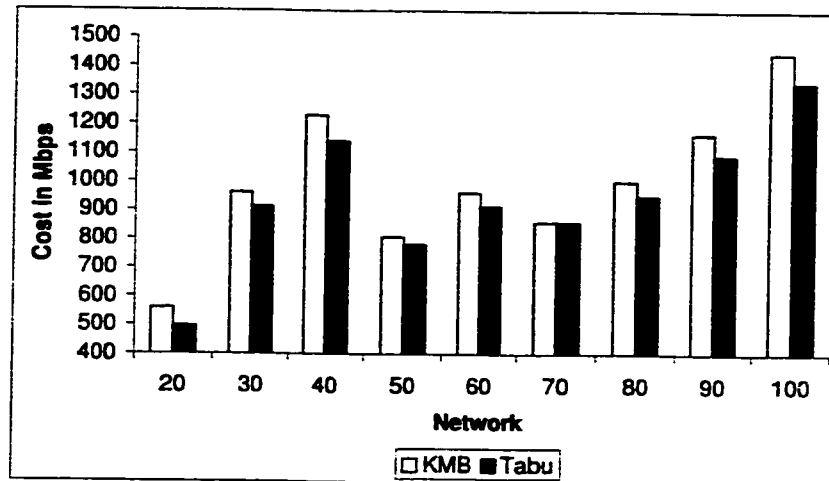


Figure 5.4: Asymmetric load. Cost comparison of KMB and Tabu. Group size =10.

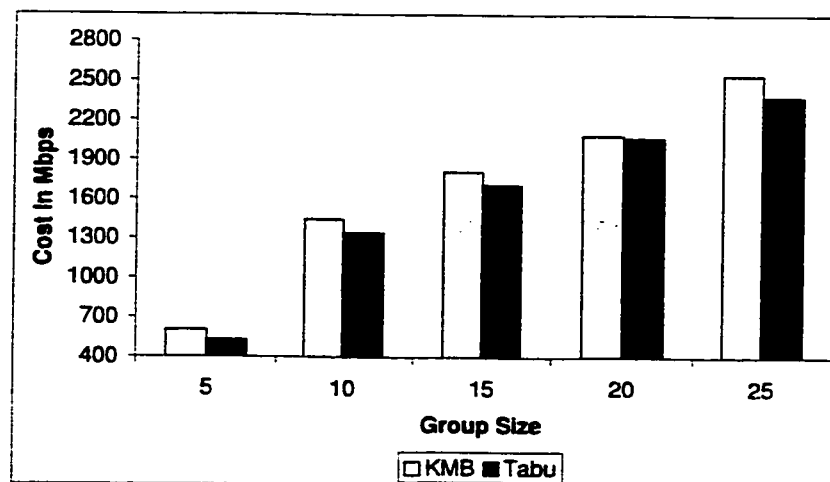


Figure 5.5: Asymmetric load. Cost comparison of KMB and Tabu. Network size =100.

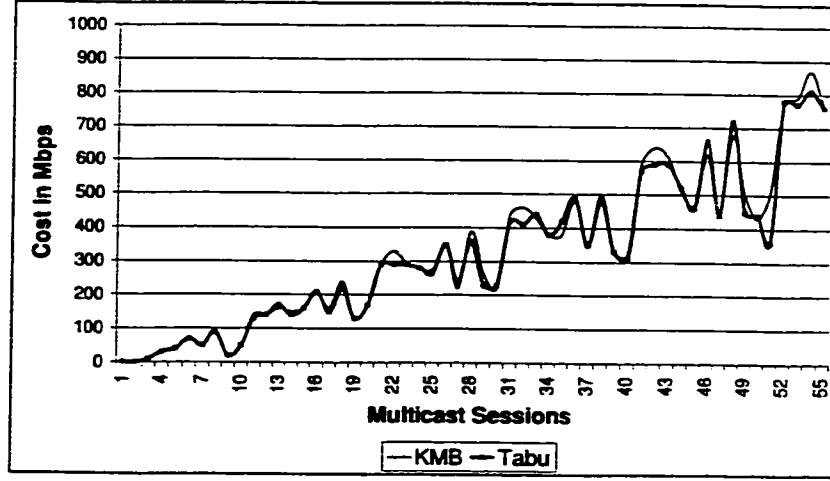


Figure 5.6: Number of multicast session versus cost between KMB and Tabu. Network Size =20, Group size=5.

### 5.2.2 Comparison of Tabu Search and RPM

RPM is used in practice, because it requires only limited information to be available at each node in order to construct a reverse shortest path MC tree [42]. Figures 5.7 and 5.9 show the cost comparison between RPM and Tabu under fixed group size of 10 and different networks for symmetric networks and asymmetric networks respectively. Figures 5.8 and 5.10 show the cost comparison between RPM and Tabu under different group size for symmetric networks and asymmetric networks respectively. Our proposed tabu search based heuristic was able to identify better trees for all test networks. Figure 5.11 shows the number of multicast sessions versus cost for a network of 20 nodes and group size of 5. The sharp line in the graph at multicast session 52 for RPM indicates the failure of RPM to identify the alternate paths when the links got saturated.

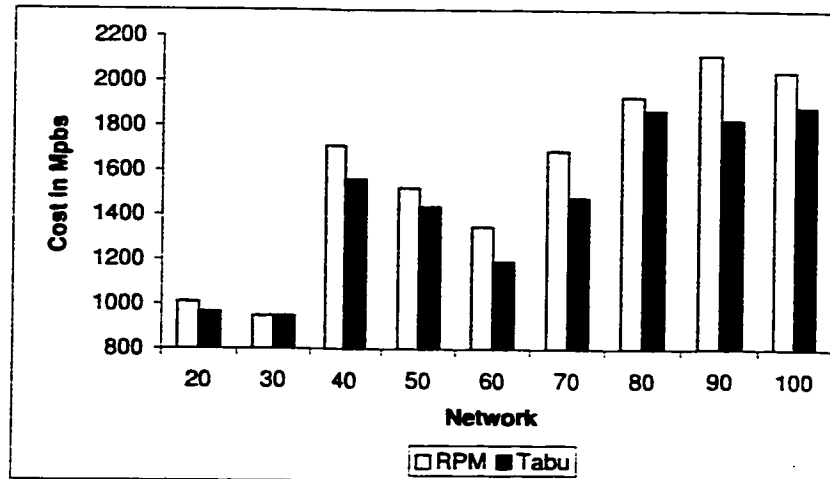


Figure 5.7: Symmetric load. Cost comparison of RPM and Tabu. Group size =10.

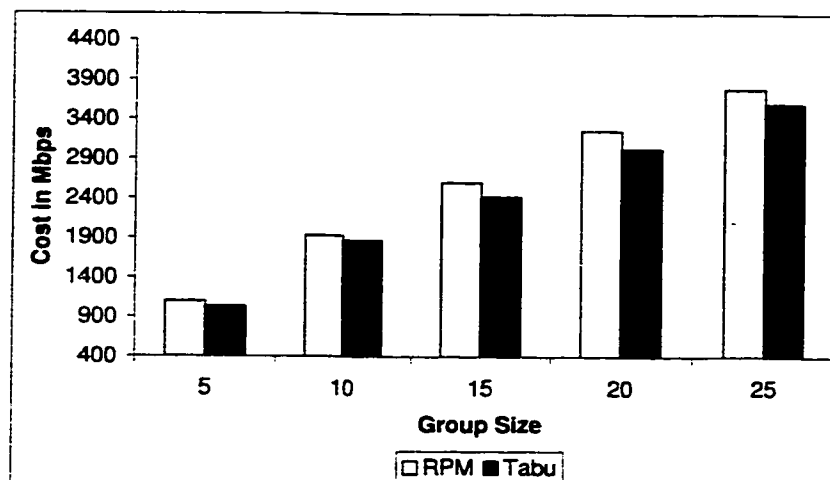


Figure 5.8: Symmetric load. Cost comparison of RPM and Tabu. Network size =80.



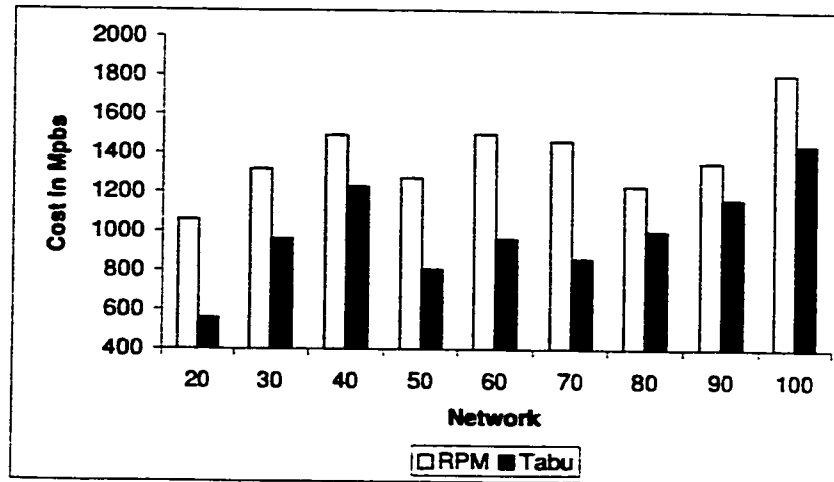


Figure 5.9: Asymmetric load. Cost comparison of RPM and Tabu. Group size =10.

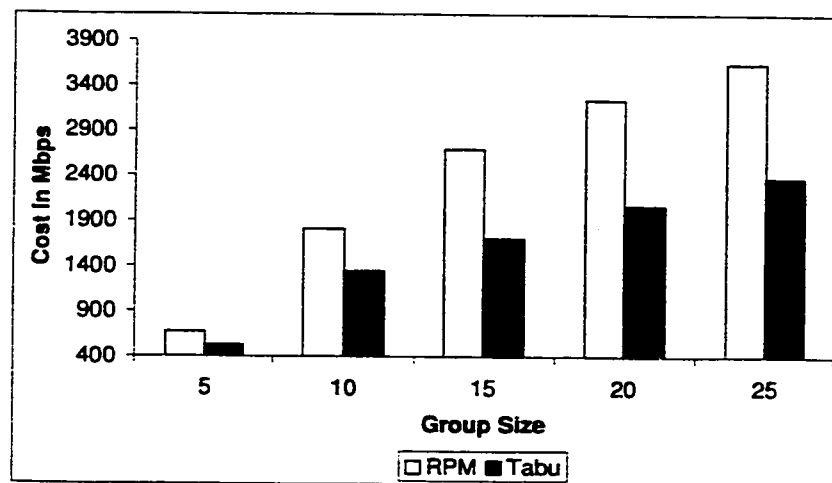


Figure 5.10: Asymmetric load. Cost comparison of RPM and Tabu. Network size=100.

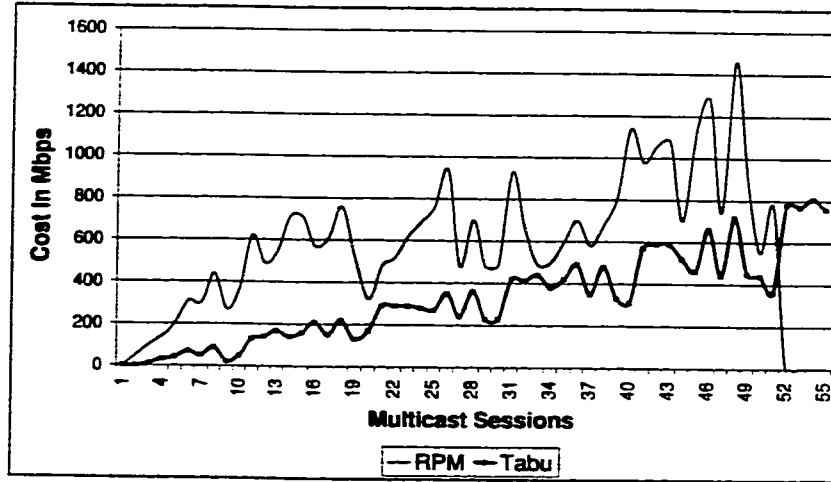


Figure 5.11: Number of multicast session versus cost between RPM and Tabu. Network size =20, Group size=5.

### 5.3 Simulation Results for Delay Constrained Multicast Tree Design Problem

Many existing delay constrained Steiner tree heuristics including KPP1 [11], KPP2 [11], CAO [24], BSMA [23], SC [26], MSC [27] are already implemented in MCRSIM. The same experiment setup is used as mentioned in Section 5.2. The tabu search based algorithm was run on 20, 30, 40, 50, 60, 70, 80, 90, and 100 nodes random graph with  $\alpha = 0.15$  and  $\beta = 2.2$ . For each node, 5 different random graphs have been generated and the results are the average of these 5 random graphs. The  $B_{min}$  and  $B_{max}$  are assumed to be 10Mbps and 150Mbps respectively. The default link capacity is 155Mbps. The following experiments have been carried out:

- Comparison of tabu search with SC [26] and MSC [27].

- Comparison of tabu search Tabu with KPP1 [11] and KPP2 [11].
- Comparison of tabu search Tabu with CAO [24].
- Comparison of tabu search Tabu with BSMA [23].

### 5.3.1 Comparison of Tabu Search with SC and MSC

In Section 3.2, we have discussed SC and MSC algorithms. Figures 5.12 and 5.13 show the cost comparison between SC, MSC and Tabu for Symmetric and Asymmetric load respectively under fixed group size, fixed delay  $U$  and different networks. Figure 5.14 shows the cost comparison under fixed group size and fixed network with increasing value of  $U$ . Both SC and MSC showed no change in cost with increasing value of  $U$  because of their Semi-Constrained Nature. Figure 5.15 shows the cost comparison under fixed network delay and fixed network with different number of Group Members. SC is not shown in Figure 5.15 because it was unable to find a multicast tree on group size more than 15 node network. Our proposed tabu search based heuristic was able to identify better trees for all Test Networks. Figure 5.16 shows the number of multicast sessions versus cost for a network of 20 nodes and group size of 5. The tabu was able to find more multicast sessions and of better quality as compared to SC and MSC.

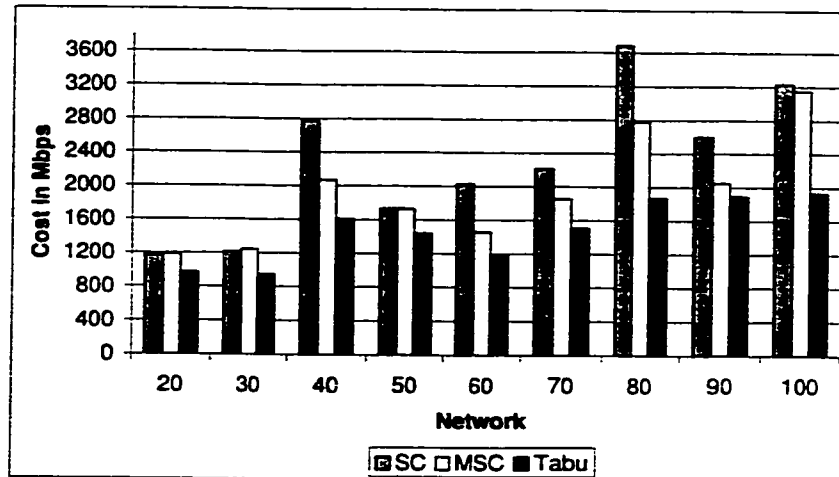


Figure 5.12: Symmetric load. Cost comparison of SC, MSC and Tabu. Group size =10 and U=0.05 Sec.

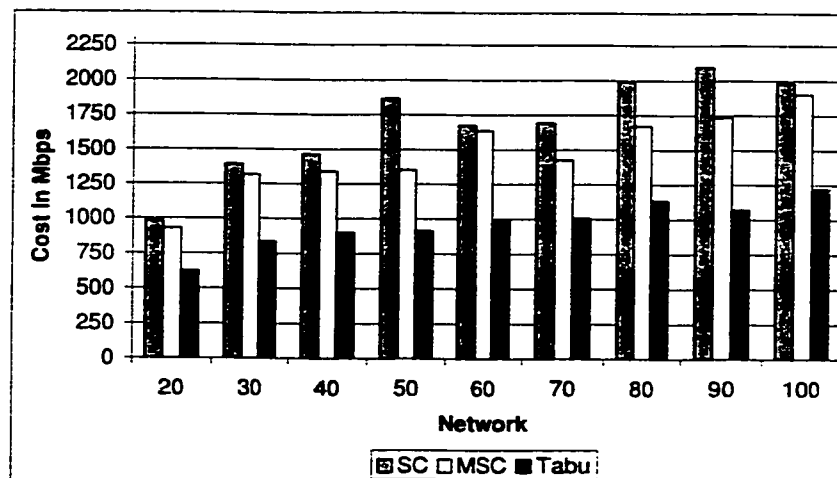


Figure 5.13: Asymmetric load. Cost comparison of SC, MSC and Tabu. Group size =10 and U=0.04 Sec.

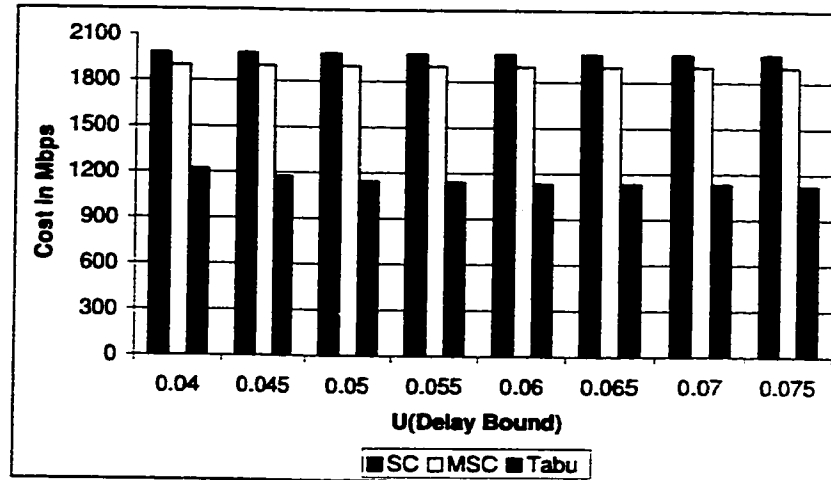


Figure 5.14: Asymmetric load. Cost comparison of SC, MSC and Tabu. Group size =10, Network size=100 nodes.

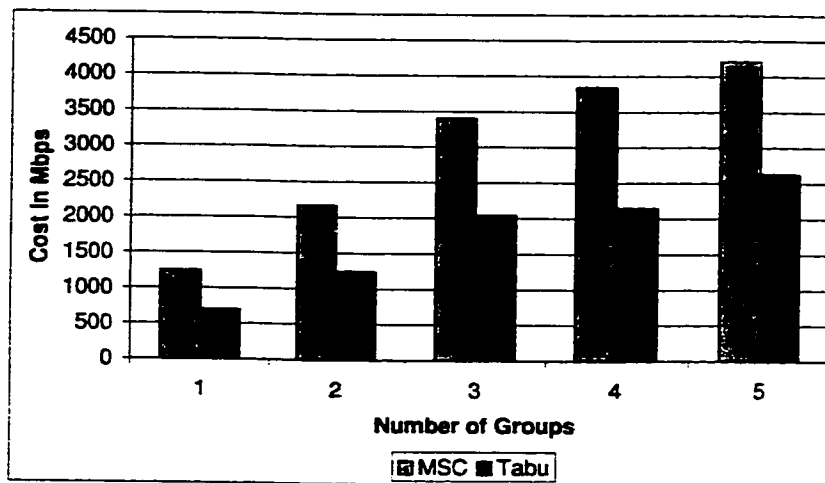


Figure 5.15: Asymmetric load. Cost comparison of MSC and Tabu.  $U = 0.05$  sec, Network size=100 nodes.

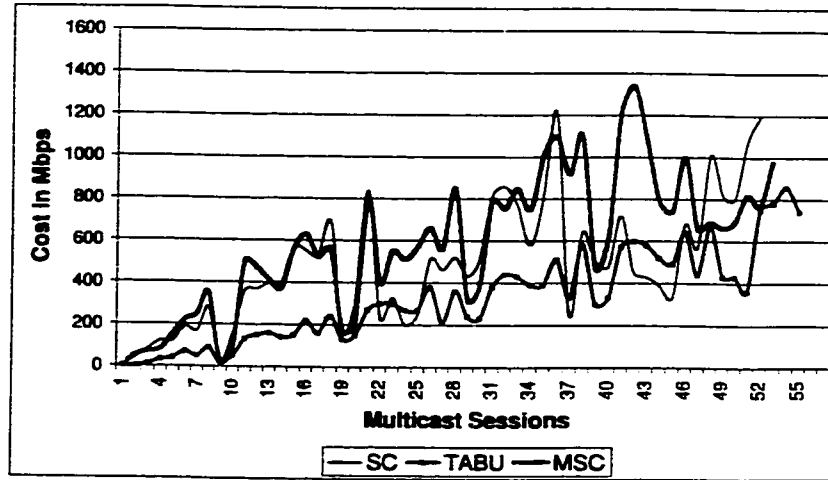


Figure 5.16: Number of multicast session versus cost between SC, MSC, and Tabu. Network size =20, Group size=5.

### 5.3.2 Comparison of Tabu Search with KPP1 and KPP2

In Section 3.2, we discussed KPP1 and KPP2 algorithms. Figures 5.17 and 5.18 show the cost comparison between KPP1, KPP2 and Tabu for asymmetric networks under fixed group size of 10,  $U = 0.04$  sec and different networks. Figure 5.19 shows the cost comparison under fixed group size and fixed network and with increasing value of  $U$ . Figure 5.20 shows the cost comparison under fixed network delay and fixed network of 100 nodes with different number of Group Members. Our proposed tabu search based heuristic was able to identify better trees for all Test Networks. Figures 5.21 and 5.22 show the number of multicast sessions versus cost for a network of 20 nodes and group size of 5. The tabu able to find multicast session of better quality as compared to KPP1 and KPP2. The number of multicast sessions are the same for both KPP and tabu.

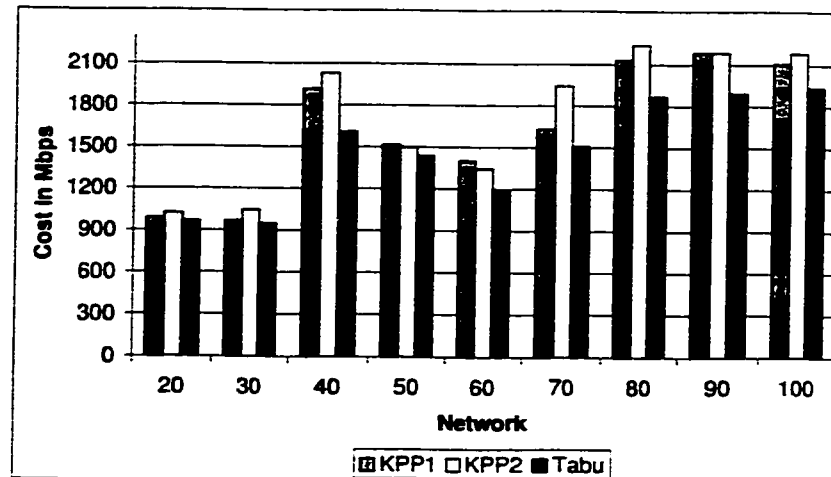


Figure 5.17: Symmetric load. Cost comparison of KPP2, KPP1 and Tabu. Group size =10 and  $U=0.05$  Sec.

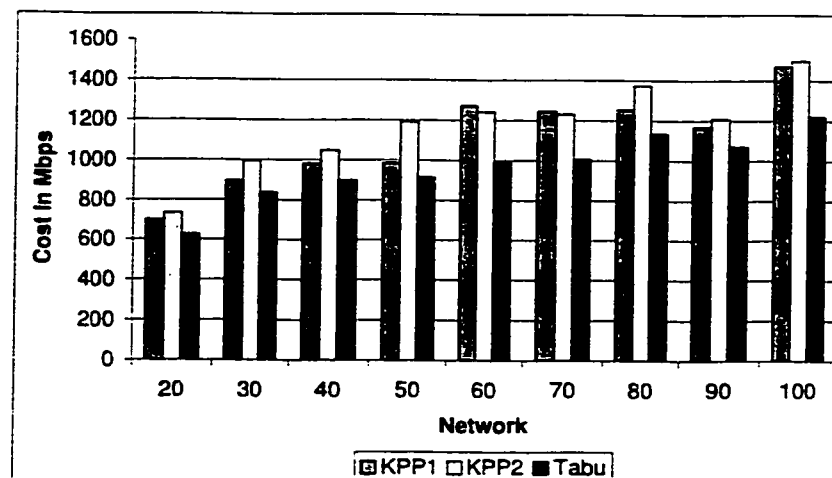


Figure 5.18: Asymmetric load. Cost comparison of KPP2, KPP1 and Tabu. Group size =10 and  $U=0.04$  Sec.

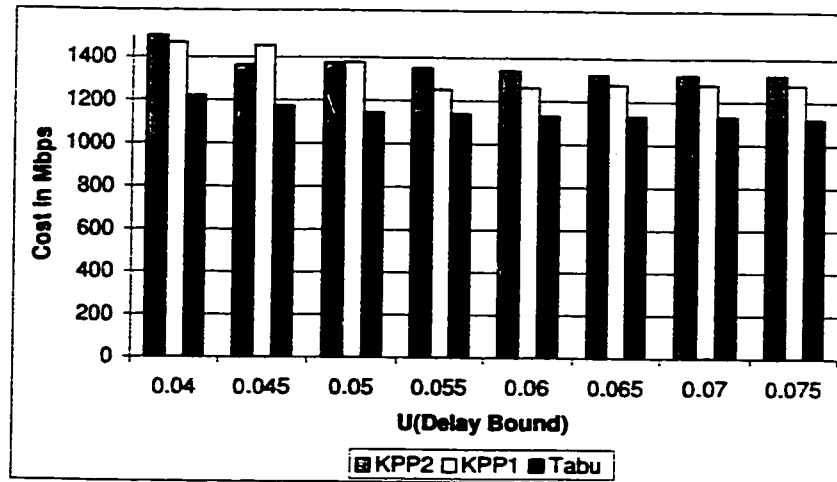


Figure 5.19: Asymmetric load. Cost comparison of KPP2, KPP1 and Tabu. Group size =10, Network size=100 nodes.

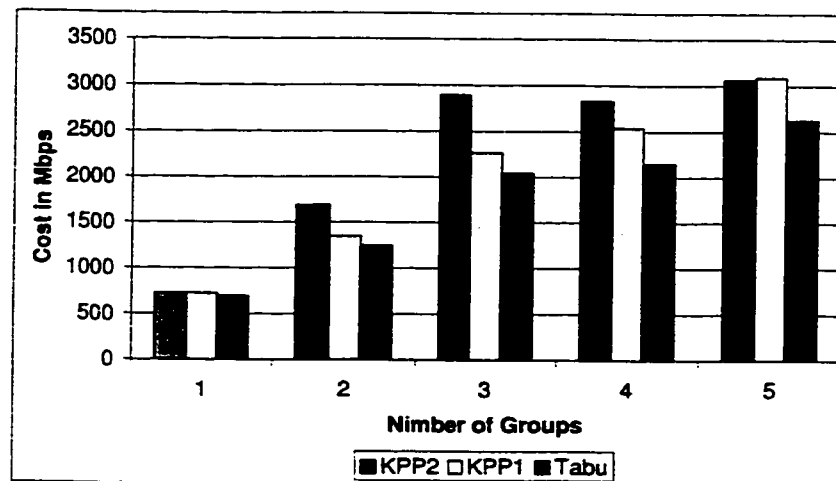


Figure 5.20: Asymmetric load. Cost comparison of KPP2, KPP1 and Tabu.  $U = 0.05$  sec, Network size=100 nodes.



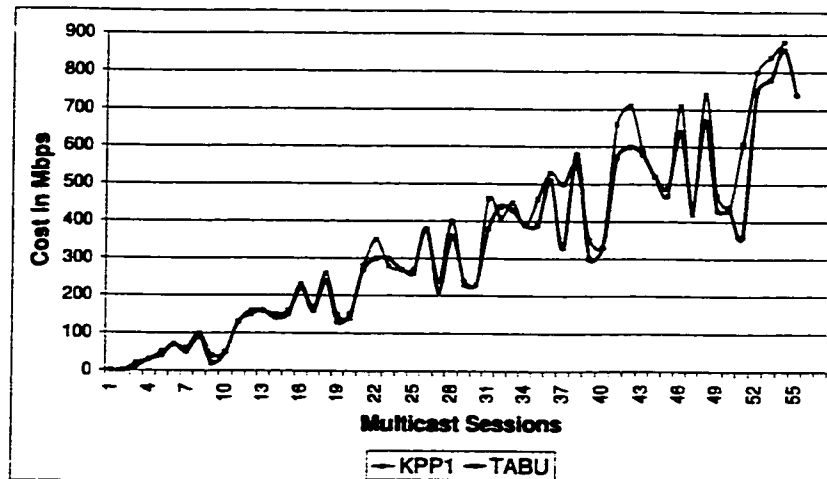


Figure 5.21: Number of multicast session versus cost between KPP1 and Tabu. Network size =20, Group size=5.

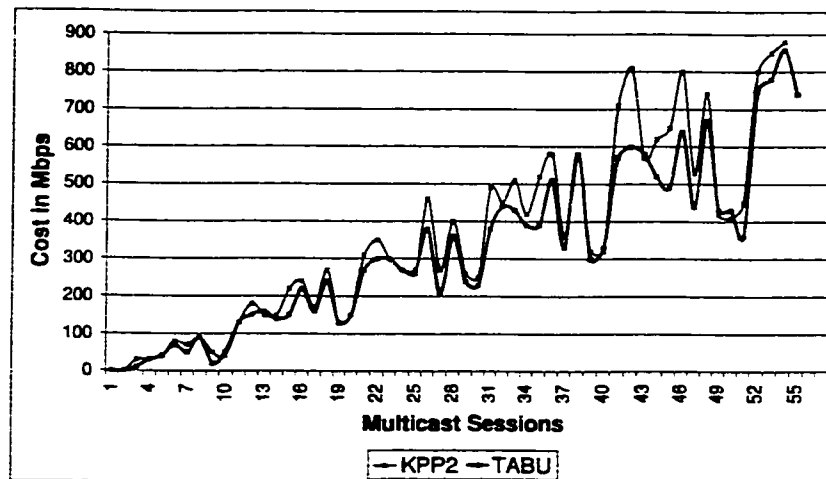


Figure 5.22: Number of multicast session versus cost between KPP2 and Tabu. Network size =20, Group size=5.

### 5.3.3 Comparison of Tabu Search and CAO

Figures 5.23 and 5.24 show the cost comparison between CAO and Tabu for symmetric and asymmetric load under fixed group size, fixed delay and different networks. Figure 5.25 shows the cost comparison under fixed group size and fixed network and with increasing value of  $U$ . Figure 5.26 shows the cost comparison under fixed network delay and fixed network and with different number of group members. Our proposed tabu search based heuristic was able to identify better trees for all test networks. Figure 5.27 shows the number of multicast sessions versus cost for a network of 20 nodes and group size of 5. The tabu able to find multicast session of better quality in almost all cases as compared to CAO. The number of multicast sessions are the same for CAO and tabu.

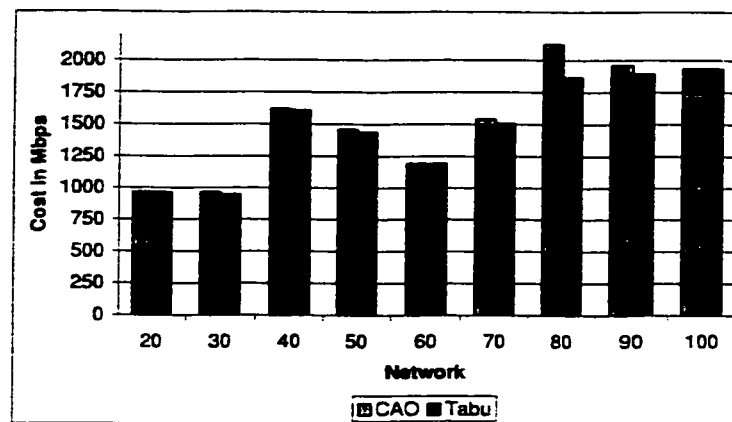


Figure 5.23: Symmetric load. Cost comparison of CAO and Tabu. Group size =10 and  $U=0.05$  Sec.

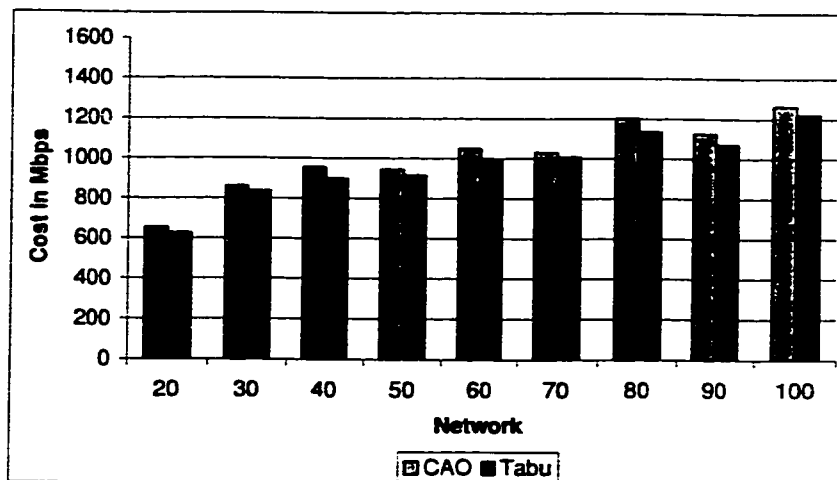


Figure 5.24: Asymmetric load. Cost comparison of CAO and Tabu. Group size =10 and  $U=0.04$  Sec.

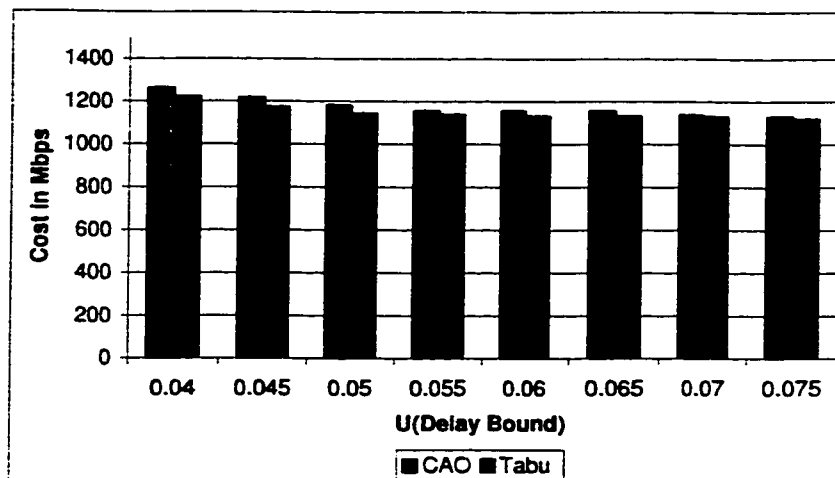


Figure 5.25: Cost comparison of CAO and Tabu. Group size =10, Network size=100 nodes.

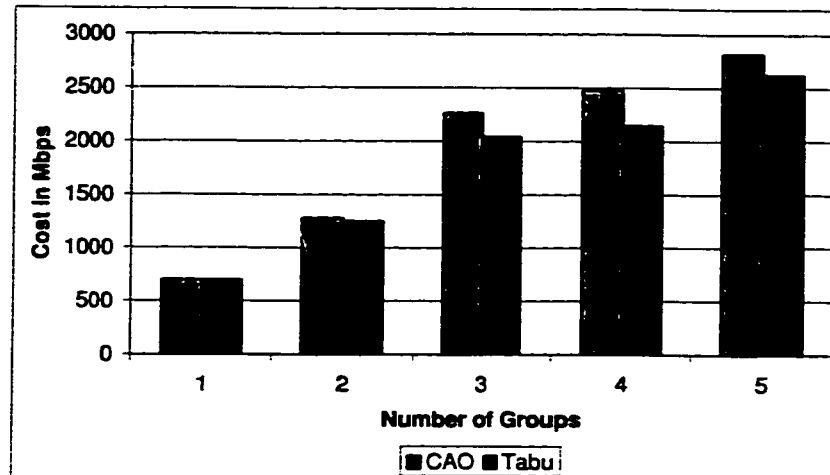


Figure 5.26: Cost comparison of CAO and Tabu.  $U = 0.05$  Sec, Network size=100 nodes.

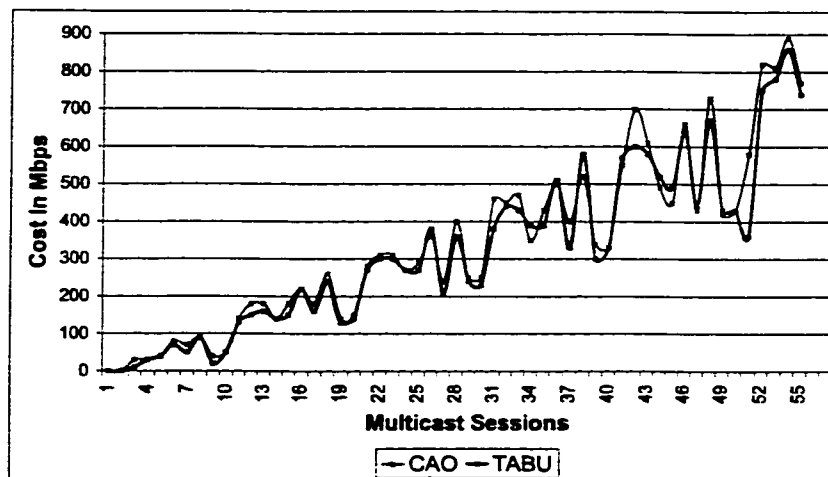


Figure 5.27: Number of Multicast Session versus cost between CAO and Tabu. Network size =20, Group size=5.

### 5.3.4 Comparison of Tabu Search and BSMA

Figures 5.28 and 5.29 show the cost comparison between BSMA and Tabu for symmetric and asymmetric networks under fixed group size, fixed delay and different networks. Figure 5.30 shows the cost comparison for asymmetric networks under fixed group size and fixed network of 100 nodes with increasing value of  $U$ . Figure 5.31 shows the cost comparison for asymmetric networks under fixed network delay and fixed network of 100 nodes with different number of group members. Our proposed tabu search based heuristic was able to identify better trees for all test networks. Figure 5.32 shows the number of multicast sessions versus cost for a network of 20 nodes and group size of 5. The tabu able to find multicast session of better quality in almost all cases as compared to BSMA. The number of multicast sessions are the same for BSMA and Tabu.

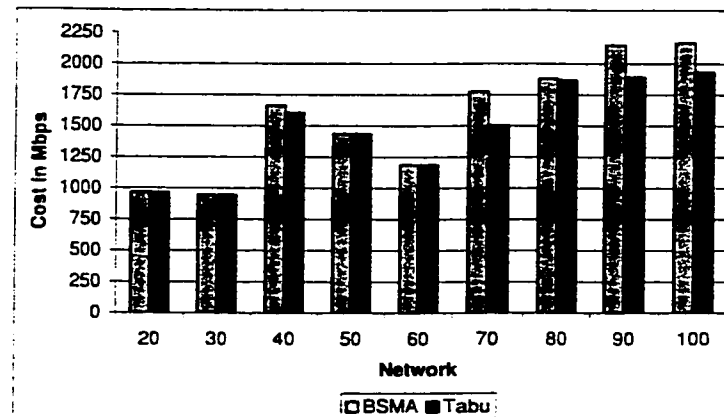


Figure 5.28: Symmetric load. Cost comparison of BSMA and Tabu. Group size =10 and  $U=0.05$  Sec.

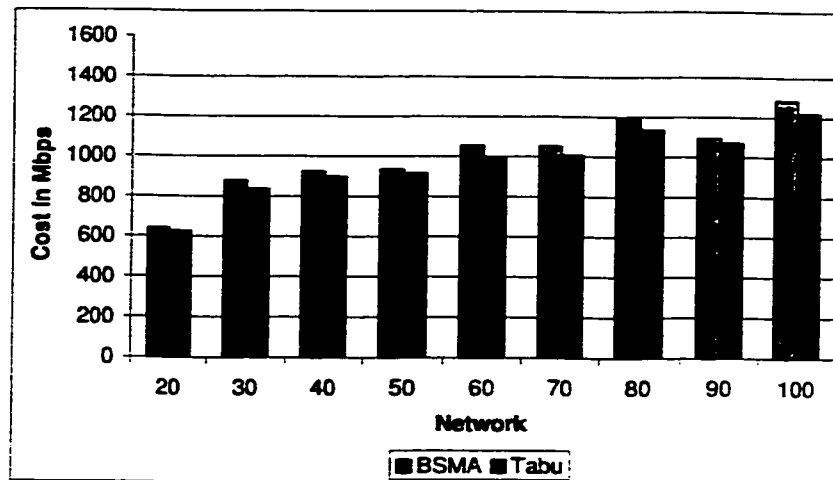


Figure 5.29: Asymmetric load. Cost comparison of BSMA and Tabu. Group size =10 and  $U=0.04$  Sec.

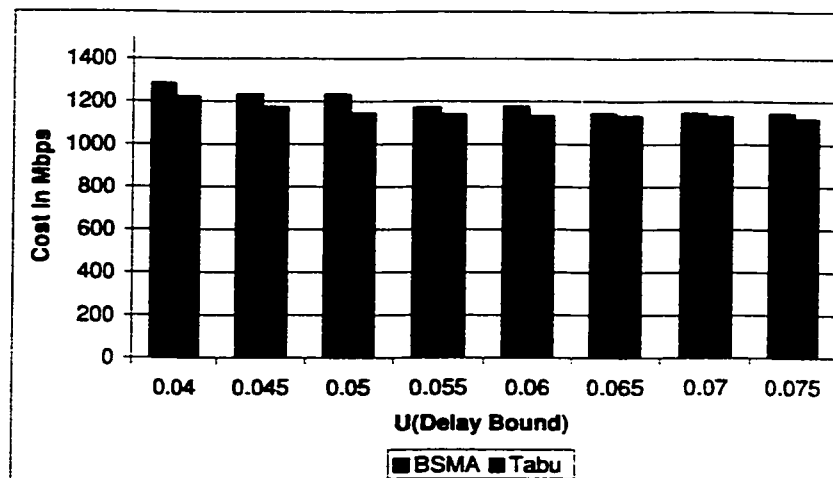


Figure 5.30: Cost comparison of BSMA and Tabu. Group size =10, Network size=100 nodes.

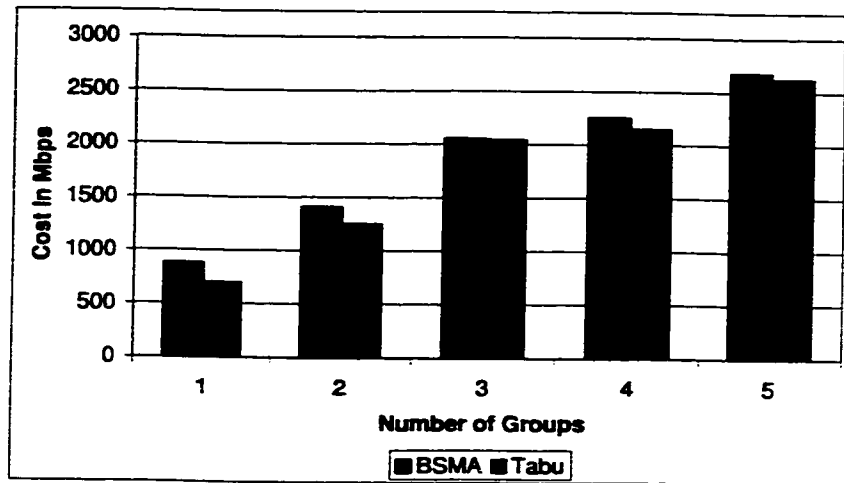


Figure 5.31: Cost comparison of BSMA and Tabu.  $U = 0.05$  Sec, Network size=100 nodes.

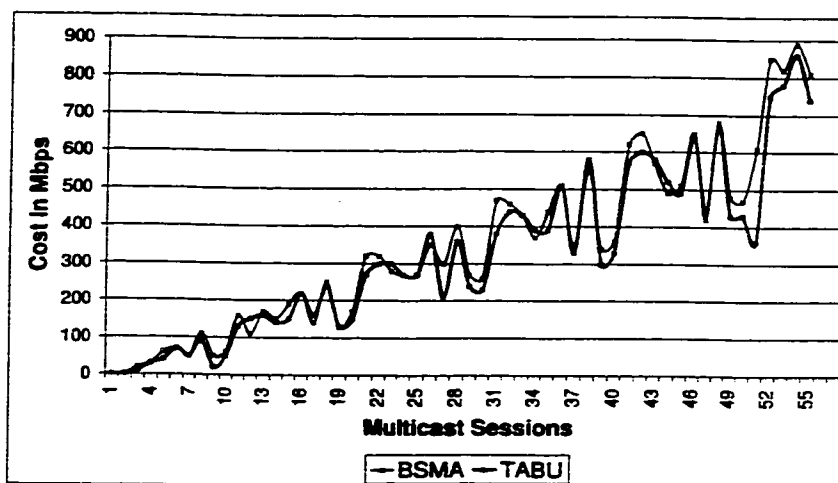


Figure 5.32: Number of multicast session versus cost between BSMA and Tabu. Network size =20, Group size=5.

### 5.3.5 Quality of Solution by Tabu Search

Figure 5.33 shows how well focused is tabu search on the good solution subspace. It is clear from the figure that more than 50% of the multicast trees found and evaluated by TS were in the good solution subspace (barchart highly skewed towards the left), i.e., in the cost interval [1590 – 1700). Figure 5.34 tracks with time the total number of solutions found by the proposed TS algorithm for various cost intervals. The plot clearly indicates that as more iterations are evaluated, TS keeps converging to better solution subspaces. For example, very few solutions( $< 20$ ) are found beyond 300 iterations in cost interval [1700 – 2100). In contrast, nearly all solutions found and evaluated after the first 300 iterations are in the cost interval [1590 – 1700). Figures 5.33 and 5.34 clearly indicate that TS has been well tuned to the problem of delay constrained minimum steiner tree design problem. Figure 5.35 tracks the cost of the best solution over time. As is clear, for small networks size( $< 60$ ), TS converges within a maximum of 100 iterations.

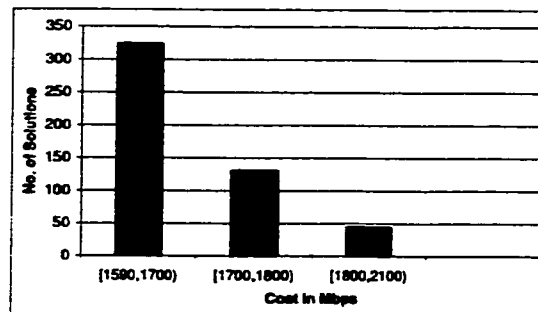


Figure 5.33: Total number of solutions evaluated by tabu search during 500 iterations for different cost ranges. Network size = 100 nodes, Group size = 10 and  $U=0.05$  Sec.



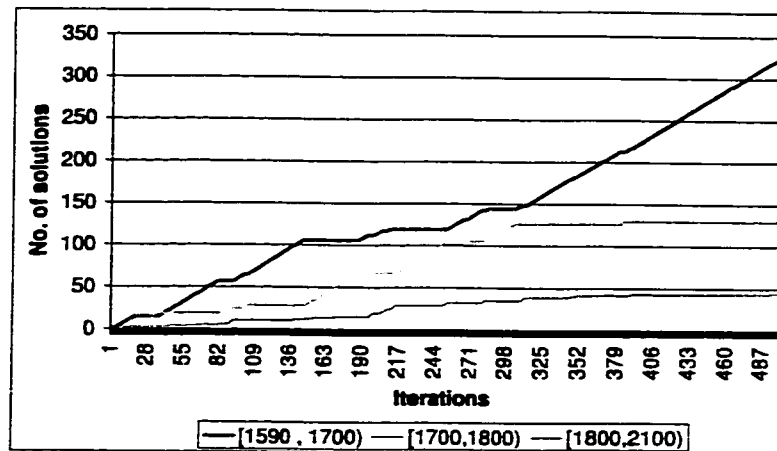


Figure 5.34: Total number of solutions evaluated by tabu search versus iteration number for different cost ranges. Network size = 100 nodes, Group size = 10 and  $U=0.05$  Sec.

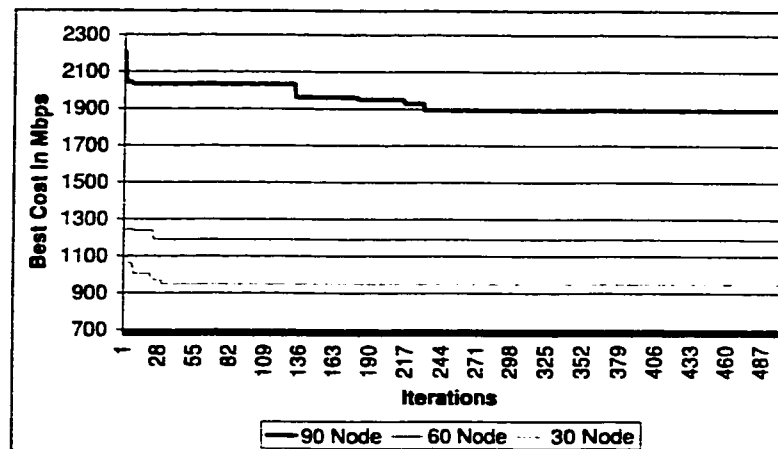


Figure 5.35: Cost of best solution found by tabu search versus iteration number for 3 networks. Group size = 10 and  $U=0.05$  Sec

## 5.4 Simulation Results for Multiobjective Minimum Steiner Tree Problem

As discussed before, the multiobjective minimum Steiner tree problem is to determine a multicast tree connecting the source node  $s$  to every destination nodes  $d \in D$  satisfying the source bandwidth requirement and such that the total cost of the tree, maximum end-to-end delay, the delay variance, and the number of steiner nodes are minimized.

We re-ran the same experiment of Section 5.2. Fuzzy based tabu search algorithm has been implemented to re-run the experiments. Comparison has been performed in such a way that the minimum end-end delay achieved from Tabu Algorithm is considered as delay bound  $U$  for other algorithms KPP, CAO and BSMA.

### 5.4.1 Comparison of KPP and Tabu

Table 5.1 shows the comparison of tabu and KPP under a fixed group members and different networks. It is clear from the table that tabu performs better than KPP as far as cost and number of Steiner nodes is concerned in all test cases. For example, a gain of 25% is achieved for the cost in a network of 60 nodes. Similarly, a gain of 80% is achieved for the number of Steiner nodes in a network of 60 nodes. For Delay variance, in some cases, KPP performs better, but the overall performance is better if we combined all three parameters in tabu. For example, although a gain

of 34% is achieved by KPP in the network of 70 nodes but for the same network, a gain of 5% and 38% in cost and number of Steiner nodes is achieved respectively by tabu. Table 5.2 shows the comparison of tabu and KPP under a fixed network of 100 nodes and different group members. It is clear from the table that overall performance of tabu is better than KPP in all test cases.

Table 5.1: Comparison of Tabu and KPP. N= Network size, D=Maximum end to end delay in *msec*, C=Cost in *Mbps*, V=Delay variance in  $\mu sec$ , and S = Number of Steiner nodes. Group size = 10.

N	D	Tabu			KPP			%Gain		
		C	V	S	C	V	S	C	V	S
100	29	1503.0	38.7	10	1569	60.7	14	4.4	56.7	45
90	29	1248.0	23.5	9	1331.75	24.9	12	6.7	6.06	26.3
80	21	1239.0	21.2	7	1406.25	26.1	11	13.5	22.8	7.7
70	32	1435.5	60.2	9	1512.0	39.49	12	5.33	-34.4	38.0
60	32	1125.7	42.8	5	1560.0	51.7	9	25.52	20.73	80
50	31	1172.2	33.7	6	1209.75	25.42	10	3.2	-24.5	66.67
40	23	1089.7	20.0	6	1236.0	24.5	8	14.2	7.135	25
30	22	1027.5	23.8	4	1066.5	29.14	5	3.80	22.56	22.22

Table 5.2: Comparison of Tabu and KPP. G= Group size, D=Maximum end to end delay in *msec*, C=Cost in *Mbps*, V=Delay variance in  $\mu sec$ , and S = Number of Steiner nodes. Network size = 50.

G	D	Tabu			KPP			%Gain		
		C	V	S	C	V	S	C	V	S
25	33	2190.0	47.44	5	2344.5	56.60	7	7.06	19.31	40
20	37	1762.5	61.41	4	1864.5	75.73	9	5.8	23.93	75.0
15	37	1251.0	71.6	5	1324.5	88.69	8	5.875	22.94	60.0
10	37	1027.5	43.71	4	1161.0	23.72	9	13.0	-45.73	125.0
5	36	817.5	62.7	6	933	10.0	11	14.13	-84.05	83.33

### 5.4.2 Comparison of CAO and Tabu

Table 5.3 shows the comparison of tabu and CAO under a fixed group members and different networks. It is clear from the table that tabu performs better than CAO as far as cost and number of Steiner nodes is concerned in all test cases. For example, a gain of 16% is achieved for the cost in a network of 60 nodes. Similarly, a gain of 80% is achieved for the number of Steiner nodes in a network of 60 nodes. For Delay variance, in some cases, CAO performs better, but the overall performance is better if we combined all three parameters in Tabu. For example, although a gain of 27% is achieved by CAO in the network of 70 nodes but for the same network, a gain of 6% and 39% in cost and number of Steiner nodes is achieved respectively by tabu. Table 5.4 shows the comparison of tabu and CAO under a fixed network of 100 nodes and different group members. It is clear from the table that overall performance of tabu is better than CAO in all test cases.

### 5.4.3 Comparison of BSMA and Tabu

A comparison of tabu and BSMA under a fixed group members and different networks is shown in Table 5.5. It is clear from the table that tabu performs better than BSMA as far as cost and number of Steiner nodes is concerned in all test cases. For example, a gain of 10% is achieved for the cost in a network of 50 nodes. Similarly, a gain of 40% is achieved for the number of Steiner nodes in a network of 50

Table 5.3: Comparison of Tabu and CAO. N= Network size, D=Maximum end to end delay in *msec*, C=Cost in *Mbps*, V=Delay variance in  $\mu sec$ , and S = Number of Steiner nodes. Group size = 10.

N	D	Tabu			CAO			%Gain		
		C	V	S	C	V	S	C	V	S
100	29	1503.0	38.7	10	1593.0	39.76	11	6.0	2.64	15.0
90	29	1248.0	23.5	9	1377.75	17.8	10	10.4	-6.94	10
80	21	1239.0	21.2	7	1406.25	26.1	11	16.0	3.02	42.86
70	32	1435.5	60.2	9	1522.5	44.15	12	6.1	-26.64	38.9
60	32	1125.7	42.8	5	1305.0	43.8	9	16.0	2.32	80.0
50	31	1172.2	33.7	6	1218.75	25.42	10	3.97	-7.9	41.67
40	23	1089.7	20.0	6	1236.0	24.5	8	13.42	22.1	25
30	22	1027.5	23.8	4	1064.2	26.4	5	3.6	10.9	22.2

Table 5.4: Comparison of Tabu and CAO. G= Group size, D=Maximum end to end delay in *msec*, C=Cost in *Mbps*, V=Delay variance in  $\mu sec$ , and S = Number of Steiner nodes. Network size = 50.

G	D	Tabu			CAO			%Gain		
		C	V	S	C	V	S	C	V	S
25	33	2190.0	47.44	5	2541.0	59.5	10	16.03	25.34	100.0
20	37	1762.5	61.41	4	2001.0	67.28	9	13.53	9.56	125.0
15	37	1251.0	71.56	5	1326.0	81.25	8	6.0	13.54	60.0
10	37	1027.5	43.71	4	1047.0	51.2	5	1.9	17.16	25.0
5	36	817.5	62.7	6	1087.5	25.9	12	33.03	-58.7	100.0

nodes. BSMA is assumed is to be within 7% of the optimal for small networks. For Delay variance, in some cases, BSMA performs better, but the overall performance is better if we combined all three parameters in tabu. For example, although a gain of 42% is achieved by BSMA in the network of 70 nodes but for the same network, a gain of 1% and 10% in cost and number of Steiner nodes is achieved respectively by Tabu. Table 5.6 shows the comparison of Tabu and BSMA under a fixed network of 100 nodes and different group members. It is clear from the table that overall performance of tabu is better than BSMA in all test cases.

Table 5.5: Comparison of tabu and BSMA. N= Network Size, D=Maximum end to end delay in *msec*, C=Cost in *Mbps*, V=Delay variance in  $\mu sec$ , and S = Number of Steiner nodes. Group size = 10.

N	D	Tabu			BSMA			%Gain		
		C	V	S	C	V	S	C	V	S
100	29	1503.0	38.7	10	1597.5	41.1	13	5.9	5.7	23.1
90	29	1248.0	23.5	9	1322.2	28.4	11	5.6	17.5	17.4
80	21	1239.0	21.2	7	1401.7	21.9	10	11.6	2.93	30
70	32	1435.5	60.2	9	1450.5	42.5	10	1.03	-41.7	10.0
60	32	1125.7	42.8	5	1139.2	32.4	8	1.2	-32.0	41.2
50	31	1172.2	33.7	6	1308.7	31.5	10	10.43	-6.7	40
40	23	1089.7	20.0	6	1148.2	21.2	8	5.1	5.64	25
30	22	1027.5	23.8	4	1056.7	27.2	6	2.77	12.53	25

#### 5.4.4 Quality of Solution by Tabu Search

Figure 5.36 and 5.37 show the cost, maximum end-to-end delay, delay variance and steiner nodes versus iterations during the solution search space by Tabu. Figure 5.38

Table 5.6: Comparison of Tabu and BSMA. G= Group size, D=Maximum end to end delay in *msec*, C=Cost in *Mbps*, V=Delay variance in  $\mu sec$ , and S = Number of Steiner nodes. Network size = 50.

G	D	Tabu			BSMA			%Gain		
		C	V	S	C	V	S	C	V	S
25	33	2190.0	47.44	5	2139.0	54.21	6	-2.32	14.3	20.0
20	37	1762.5	61.41	4	1834.5	75.35	7	4.1	22.70	75.0
15	37	1251.0	71.56	5	1402.5	76.44	7	12.11	6.82	40.0
10	37	1027.5	43.71	4	1302.0	40.4	9	26.7	-7.57	125.0
5	36	817.5	62.7	6	1014.0	48.24	10	24.04	-23.06	66.67

shows how well focused tabu search is on the good solution subspace. As it is clear from the figure, more than 50% of the multicast trees found and evaluated by tabu search were in the good solution subspace (barchart highly skewed towards the right), i.e., in the membership interval  $[0.4 - 0.72]$ . Figure 5.39 tracks with time the total number of solutions found by the proposed tabu search algorithm for various cost intervals. The plot clearly indicates that as more iterations are evaluated, tabu search keeps converging to better solution subspaces. For example, very few solutions ( $< 50$ ) are found beyond 700 iterations in membership range  $[0.0 - 0.4]$ . In contrast, nearly all solutions found and evaluated after 700 iterations are in the cost interval  $[0.4 - 0.72]$ .

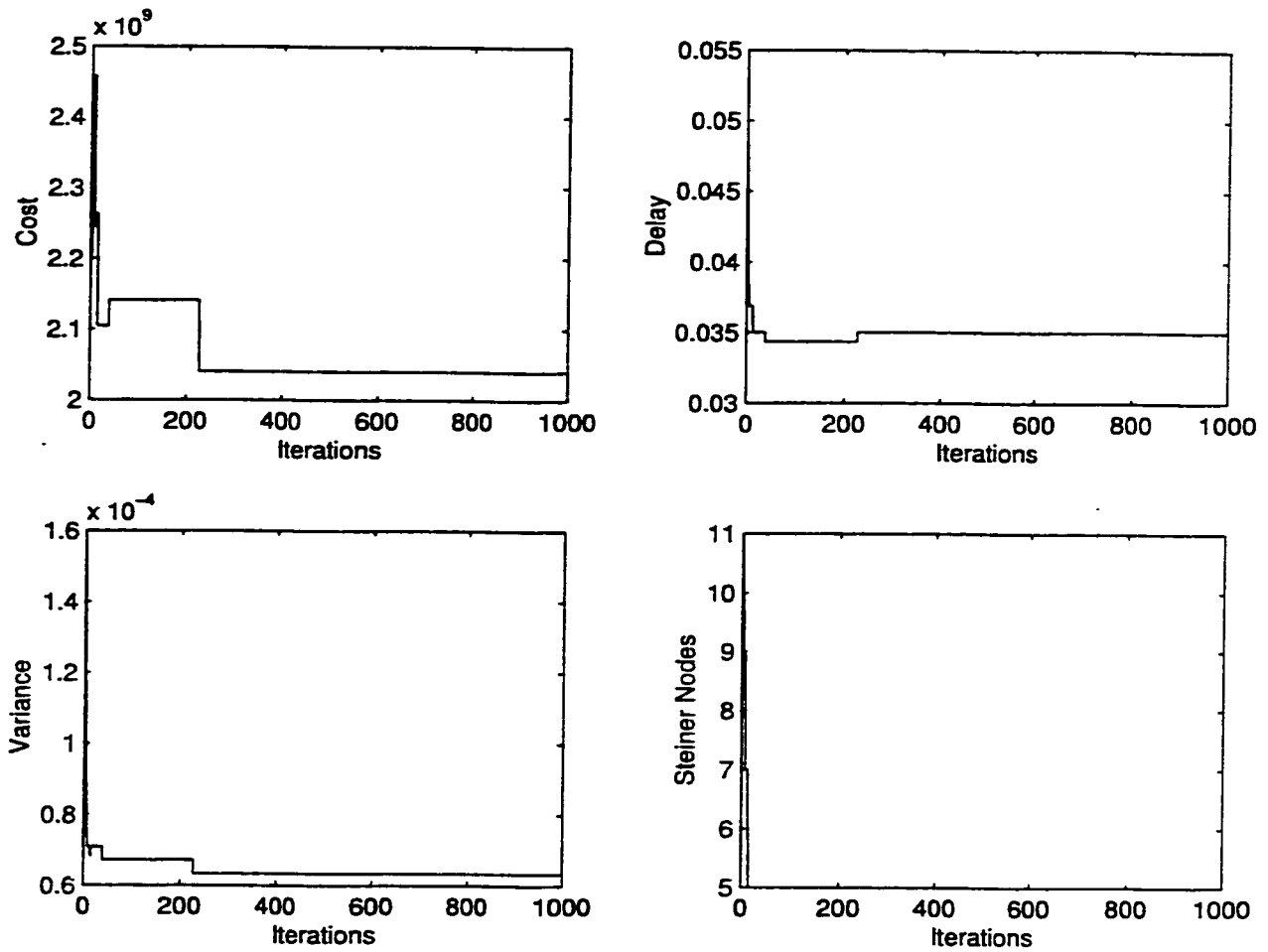


Figure 5.36: Four different objective functions versus iterations. Network size = 50 Nodes. Group size = 20.



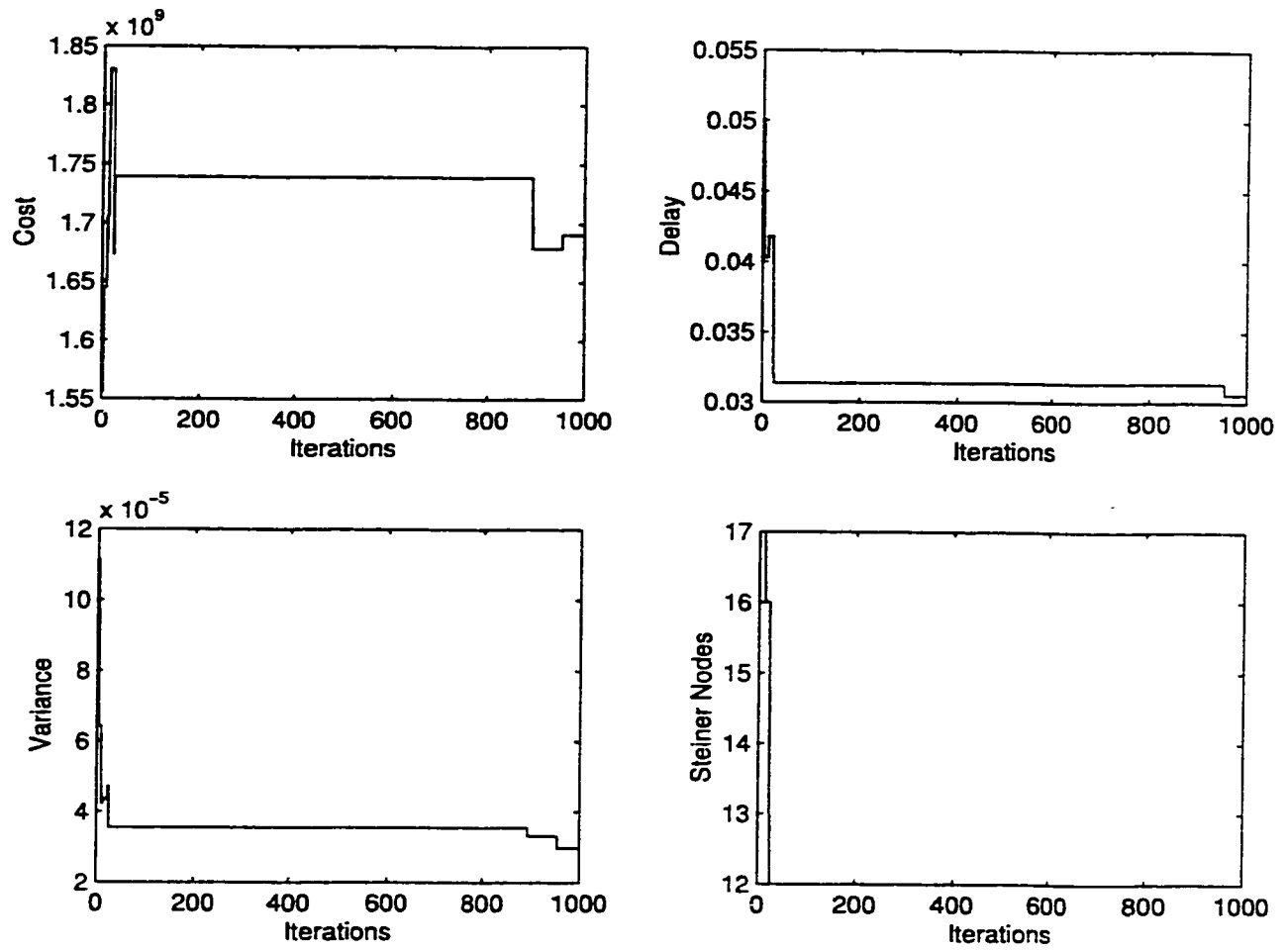


Figure 5.37: Four different objective functions versus iterations. Network size = 100 Nodes. Group size = 10.

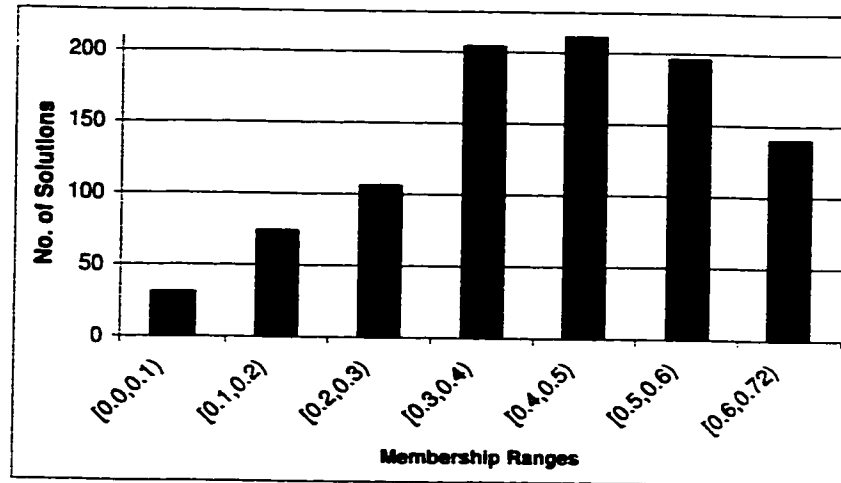


Figure 5.38: Total number of solutions evaluated by tabu search during 1000 iterations for different membership ranges. Network size = 100 nodes, Group size = 10.

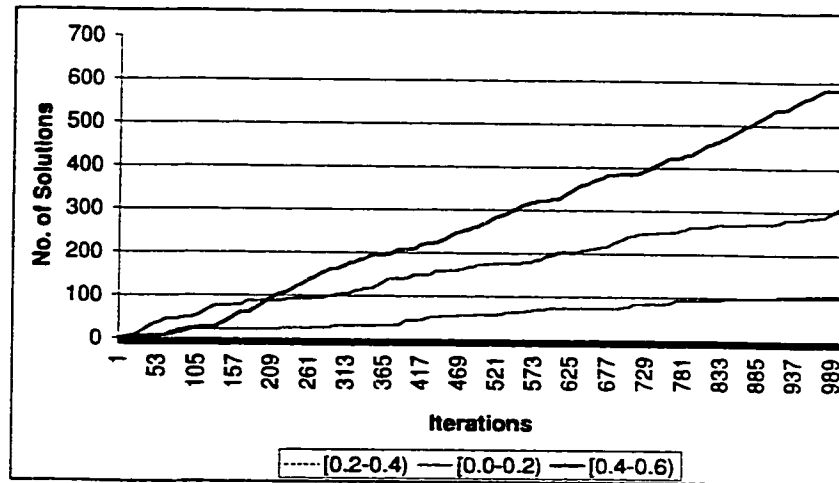


Figure 5.39: Total number of solutions evaluated by tabu search as a function of time (Iteration number) for different membership intervals. Network size = 100 nodes and Group size = 10.

## 5.5 Conclusion

In this chapter, we presented and discussed various experiments that were performed using the approaches based on tabu search algorithm. Our proposed idea has performed well for all three static multicast routing problems. Fuzzy logic is used for the cost function in multiobjective minimum Steiner tree problem. The proposed tabu search algorithm was always able to find a multicast tree if one exists. Tabu search is better in terms of tree cost as compared to BSMA, CAO, KPP1 and KPP2. Results suggest that the search performed by tabu search is of intelligent and superior quality. Further, as time elapsed, tabu search progressively zoomed towards a better solution subspace, a desirable characteristics of approximation iterative heuristics.

## Chapter 6

# Dynamic Multicast Routing

If nodes are allowed to join or leave the multicast group at any time during the lifetime of the multicast connection, then the problem is called Dynamic Multicast Routing Problem. In the dynamic multicast routing problem, the members of the multicast group are dynamically changing. Let us take a snapshot of the multicast group and denote it by  $\hat{S}$ . Suppose that a static multicast routing algorithm is applied to find a multicast tree for  $\hat{S}$  in  $G$ . A good static multicast routing algorithm can generally produce better results than a dynamic one [4]. This is because a dynamic multicast routing algorithm usually adds or removes a node without disrupting the connections to existing members. It does not take the multicast group as a whole into consideration and reconstructs the multicast tree as a static algorithm does. Therefore, a static routing algorithm which produces near optimal results can serve as a reference for comparing dynamic multicast routing algorithms. In this

problem, KPP is used as a reference algorithm. KPP is static delay constrained algorithm. The minimum delay found by our proposed technique is considered as delay bound for this algorithm.

This chapter is organized as follows, Section 6.1 describes the proposed approach for Multiobjective Steiner Tree Optimization with dynamic membership Problem. Section 6.2 explains experiment setup and simulation results. Section 6.3 concludes the chapter.

## 6.1 Proposed Approach

In a dynamic multicast connection, nodes may join or leave the multicast group dynamically during the lifetime of the multicast connection. First, a static multicast tree is generated by using Fuzzy based tabu search algorithm when a multicast session is established. Then, the multicast tree is modified for each connection request (either addition or deletion) by the scheme as mentioned below.

### 6.1.1 Nodes Leaving

Let us assume that node  $n$  in the tree issues a leave request to end its participation in the multicast session. If node  $n$  is not an end destination node in the existing tree  $T_i$ , no action will be taken. The new tree  $T_{i+1}$  will be same as  $T_i$ , i.e.,  $T_{i+1} = T_i$ , with the only difference that node  $n$  will stop forwarding the multicast packets to

its users. If  $n$  is an end destination node of  $T_m'$ , then in order to avoid wasting bandwidth, tree  $T_m$  has to be pruned to exclude node  $n$  and related steiner nodes and links used in the tree  $T_m$  to avoid forwarding packets to  $n$ . This approach used here is similar to those reported in earlier works [4, 5].

### 6.1.2 Nodes Joining

There are two situations when a new node  $n$  intends to join tree  $T_i$ .

- the node  $n$  that wants to join the tree is not in tree  $T_i$  and
- the node  $n$  that wants to join the tree is a steiner node of tree  $T_i$ .

If the node  $n$  is a steiner node of the existing multicast tree  $T_m$ , then node  $n$  can be used without any change to forward multicast packets to its user, in addition to forwarding them to downstream nodes.

If the node  $n$  is not in the tree  $T_m$ , a shortest path is first found to all nodes (steiner nodes, destination nodes and source node) in the multicast tree from the new member node  $n$ . Now, we will treat every shortest path as a new added edge in the multicast tree. The result is in the number of multicast trees. Every multicast tree is evaluated by using Fuzzy Membership function. And the tree with the best membership function is considered as new multicast tree  $T_{i+1}$ . Thus,  $T_i$  to new tree  $T_{i+1}$  involves only establishment of a new path and does not affect any of the paths from source to destination nodes already in the multicast tree  $T_i$ .

## 6.2 Experiment Setup and Simulation Results

In our simulations, sequence of random requests are generated for adding and removing nodes. A simple probability model is used to determine whether the request is adding a node to the multicast group or removing the node from the multicast group. The probability for adding a node to the multicast group is determined by the probability function [4, 5].

$$Prob(add) = \frac{\gamma(N-M)}{\gamma(N-M) + (1-\gamma)M}$$

where  $M$  is the current number of nodes in the multicast group,  $N$  is the number of nodes in the network, and  $\gamma$  is a parameter of real number in the range  $(0, 1]$ . The parameter  $\gamma$  determines the size of the multicast group in equilibrium. Note that

- $Prob(add) = \frac{1}{2}$  if  $M = \gamma N$ .
- $Prob(add) > \frac{1}{2}$  if  $M < \gamma N$ , and
- $Prob(add) < \frac{1}{2}$  if  $M > \gamma N$ .

The probability that a request will be a delete-request is given by

$$Prob(del) = 1 - Prob(add)$$

If the request is a node addition, a node is randomly chosen from the nodes which are not in the multicast group. It is then added to the multicast group. If the

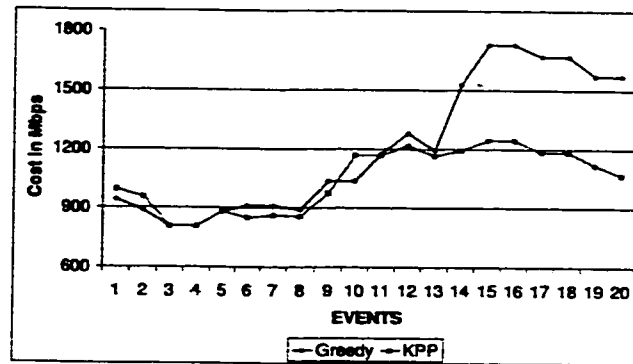
request is a node removal, a node excluding the source node is randomly chosen from the multicast group. It is then removed from the multicast group.

The above proposed technique is then applied to different networks with an initial group size of 10 members. A sequence of 20 events are generated using a fixed value for  $\gamma = M/N$ , for which  $Prob(add) = 1/2$  for each network. We are comparing our approach with other static multicast routing algorithm.

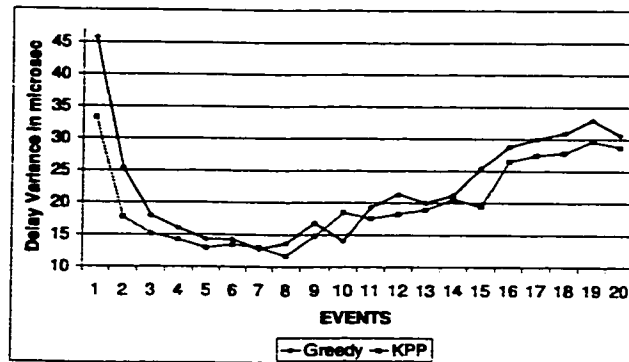
Figure 6.1 shows the cost, delay variance and number of steiner nodes comparison vs event number between greedy and KPP. An event is either leaving or joining with probability 1/2. The initial solution for Greedy is assumed to be found by Fuzzy based multiobjective tabu search algorithm. For smaller number of events, the results are comparable to that of static multicast routing algorithms. But as the number of events increases, the performance of Dynamic Greedy is decreasing as expected because it is not considering the tree as a whole in finding the multicast tree.

Table 6.1 depicts the same picture as described in Figure 6.1 but with different network of size 40 nodes and also showing whether the event is a Join or Leave. For small number of events, the result are comparable to KPP. Due to members leaving or joining, there is a change of almost 100% in the tree topology. That is why Greedy Algorithm tends to find multicast tree of high cost as comparable to KPP.

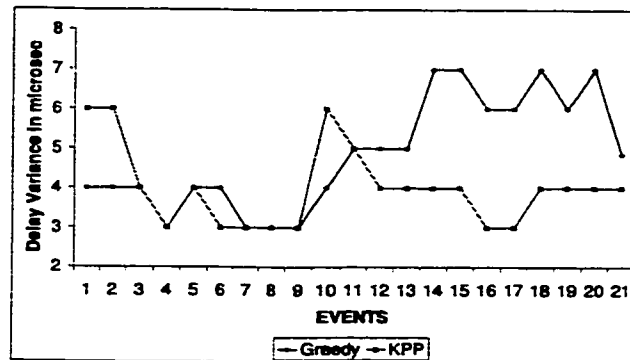




(a)



(b)



(c)

Figure 6.1: Network size = 30 nodes, Initial group size = 10. (a) Cost versus Event number between KPP and Greedy. (b) Delay variance versus Event number between KPP and Greedy. (c) Steiner nodes versus Event number between KPP and Greedy.

Table 6.1: Comparison of Dynamic Greedy and KPP. E= Event, J=Join, L=Leave, D=Maximum end to end delay in *msec*, C=Cost in *Mbps*, V=Delay variance in  $\mu sec$ , and S = Number of Steiner nodes. Initial group size = 10.

E	G	D	Dynamic Greedy			KPP		
			C	V	S	C	V	S
1	J	24	1318.5	35.14	6	1431.0	20.58	9
2	L	24	1318.5	30.54	7	1431.0	19.96	10
3	J	29	1560.0	46.9	8	1491.0	29.65	12
4	J	29	1560.0	43.1	7	1680.0	29.19	13
5	L	29	1318.5	32.6	6	1431.0	20.58	9
6	J	24	1318.5	29.05	5	1507.5	21.81	9
7	J	48	1612.5	105.0	7	1458.0	102.55	13
8	J	48	1743	99.43	7	1444.5	115.7	12
9	L	48	1612.5	105.0	7	1458.0	102.55	13
10	L	24	1318.5	29.05	5	1507.5	21.81	9
11	L	22	1197.0	24.45	5	1386.0	18.98	9
12	J	22	1342.5	21.94	5	1386.0	17.65	8
13	L	22	1152.0	21.34	4	1212.0	15.84	8
14	L	22	1008.0	19.16	3	1023.0	13.80	7
15	J	38	1173.0	68.36	4	886.5	89.3	7
16	L	38	1173.0	75.41	5	886.5	94.28	8
17	L	38	1173.0	81.56	6	858.0	41.84	4
18	J	38	1446.0	87.6	8	919.5	37.93	5
19	J	38	1891.5	117.8	10	1111.5	47.1	6
20	J	46	2158.5	165.3	11	1167.0	46.01	6

## 6.3 Conclusion

A greedy approach for dynamic multicast routing has been proposed in this chapter. Our proposed algorithm involves only in the establishment of a new path and is able to find multicast tree of good cost for small number of events. As the number of events increases, the performance of dynamic greedy is decreasing as expected because it is not considering the tree as a whole in finding the multicast tree.

# Chapter 7

## Conclusions and Future Work

### 7.1 Summary

In most general terms, the multicast routing problem is to determine a tree spanning a source node to all the members of the multicast session. Real-time applications have QoS service requirements that must be guaranteed by the underlying network. One QoS requirement, the end-to-end delay constraint, can be guaranteed in wide-area networks by using the appropriate routing algorithms. In addition to QoS requirements, many real-time applications have high bandwidth requirements, and hence it is important to use routing algorithms which manage the network bandwidth efficiently. In this thesis, we defined two functions for each link in a network: a link cost which is a function of the utilized fraction of the link's capacity, and a link delay which is a function of the delay a packet experiences when it traverses that

link. One of the objectives of our thesis is to find minimum cost multicast tree and delay constrained minimum cost multicast tree. Tabu search technique has been used to achieve both these objectives. There are also certain classes of applications in which minimizing delay variation and minimizing number of steiner nodes are important parameters along with minimizing cost and minimizing delay. Due to the non-deterministic nature of the network state, QoS measures and other objectives are imprecise. Fuzzy logic provides a suitable mathematical framework to address such a problem. In this thesis, a fuzzy logic based tabu search algorithm is also proposed to find a low cost multicast tree with desirable QoS parameters. If nodes are allowed to join or leave the multicast group any time during the lifetime of the multicast connection, it becomes another problem called as Dynamic Multicast Routing Problem. A greedy algorithm is also proposed in this thesis for Dynamic Multicast Routing Problem with desirable QoS parameters.

In Chapter 2, multicast routing problems have been described. All problems studied in this thesis are NP-complete. The optimal solutions for these problems are therefore too complex, and can only be used to benchmark heuristic solutions. In chapter 3, we surveyed previous work on multicast routing. In Chapter 4, solutions for multicast routing problems have been proposed. In Chapter 5, experimental results have been presented. Our proposed algorithm is then compared with some previous algorithms namely: KMB, RPM, KPP, CAO and BSMA. In Chapter 6, a greedy approach for Dynamic Multicasting Routing Problem has been discussed

along with simulation results.

## 7.2 Conclusions

Following are the conclusions of our research.

- A new tabu search algorithm has been proposed for both unconstrained and delay constrained multicast routing problem. Comparison with other reported algorithms suggest that search performed by our proposed algorithm is more intelligent and of superior quality than that of others.
- QoS parameters targeted are guaranteed throughput, end-to-end delay, and delay variation. Tree cost is measured by the utilization of tree links. Also tree cost is highly correlated with the number of Steiner tree nodes, i.e., nodes not members of the multicast group. A fuzzy logic based tabu search algorithm to find a low cost multicast tree with desirable QoS parameters is proposed.
- In all three problems namely: MST, CMST, and MOST, tabu search progressively zoomed towards a better solution subspace, a desirable characteristics of approximation iterative heuristics.
- A new greedy algorithm for dynamic join/leave of multicast group members is proposed to find a low cost multicast tree with desirable QoS parameters.

## 7.3 Future Work

Following are the suggestions for the future work.

- Performance of tabu search and other previous heuristics have been evaluated using simulation at randomly generated wide-area networks. Due to the rapid growth of wide-area networks, researchers are currently developing mechanisms to organize these networks into hierarchies of sub-networks. We therefore recommend that future work should use hierarchical network models.
- A distributed multicast routing algorithm can be proposed instead of centralized approach.
- A multicast routing algorithm can be proposed that can also consider sudden failures of links.

# Bibliography

- [1] Sadiq M. Sait and H. Youssef. "General Iterative Algorithms for Combinatorial Optimization". *IEEE Computer Society*, 1999.
- [2] D. Kosiur. *"IP Multicasting: The Complete Guide to Interactive Corporate Networks"*. Willey Computer Publishing, 1998.
- [3] M. Imase and B. M. Waxman. "Dynamic Steiner Problem". *SIAM J. Disc. Math*, 4(3):369–384, August 1991.
- [4] B. M. Waxman. "Routing of Multipoint Connections". *IEEE Journal on Selected Areas in Comm.*, 6(9):1617–1622, December 1988.
- [5] H. Lin and S. Lai. "VTDM - A Dynamic multicast routing problem". *Proc. IEEE INFOCOM*, pages 1426–1432, 1998.
- [6] S. Hong, H. Lee, and B. H. Park. "An efficient multicast routing algorithm for delay-sensitive applications with dynamic membership". *Proc. IEEE INFOCOM*, pages 1433–1440, 1998.
- [7] G. Manimaran and S. Raghavan. "A rearrangeable algorithm for the construction of delay-constrained dynamic multicast trees". *IEEE/ACM Transactions on Networking*, 7(4):514–529, Aug 1999.
- [8] B. M. Waxman. "Performance Evaluation of Multipoint Routing Algorithms". *IEEE INFOCOM*, pages 980–986, 1993.
- [9] H.F. Salama, D.S. Reeves, and Y. Viniotis. "Evaluation of Multicast Routing Algorithms for Real-Time Communication on High Speed Networks". *IEEE Journal on Selected Areas in Communication*, 15:332–346, April 1997.
- [10] S. Rampal and D. Reeves. "An Evaluation of Routing and Admission Control Algorithms for Multimedia Traffic". *Computer Communications*, 18(10):755–768, October 1995.



- [11] P. Kompella, C. Joseph, and C. George. "Multicast Routing For Multimedia Communication". *IEEE/ACM Transactions on Networking*, 1(3):286–292, June 1993.
- [12] Q. Zhang and Y. W. Leung. "An Orthogonal Genetic Algorithm for Multimedia Multicast Routing". *IEEE Transactions on Evolutionary Computation*, 3(1):53–62, April 1999.
- [13] F. Hwang and D. Richards. "Steiner Tree Problems". *Networks*, 22(1):55–89, January 1992.
- [14] B. M. Waxman. "Routing of Multipoint Connections". *IEEE Journal on Selected Areas in Comm.*, 6(9):1617–1622, Dec 1988.
- [15] R. Prim. "Shortest Connection Networks and Some Generalizations". *The Bell Systems Technical Journal*, 36(6):1389–1401, November 1957.
- [16] M. Doar and I. Leslie. "How Bad is Naive Multicast Routing". In *Proceedings of IEEE INFOCOM 93*, pages 82–89, 1993.
- [17] C. S. Ricudis, M. Saltouros, and M. Angelopoulos. "An efficient evolutionary algorithm for (near-) optimal Steiner trees calculation: An approach to routing of multipoint connections". *Third International Conference on Computational Intelligence and Multimedia Applications*, pages 448–453, Sept 1999.
- [18] A. Hac and K. Zhou. "A New Heuristic Algorithm For Finding Minimum-cost Multicast Trees with Bounded Path Delay". *International Journal of Network Management*, 9:265–278, 1999.
- [19] L. Guo and I. Matta. "QDMR: An Efficient QoS Dependent Multicast Routing Algorithm". *Proceeding of the Fifth IEEE Real-Time Technology and Application Symposium*, pages 213–222, June 1999.
- [20] A. Shaikh and K. Shin. "Destination-Driven Routing for Low- Cost Multicast". *IEEE Journal on Selected Areas in Communications*, 15(3):373–381, April 1997.
- [21] A. S. Tanenbaum. *Computer Networks*. Prentice-Hall Publishing Company, 1996.
- [22] C. S. Sung and J. M. Hong. "Branch-and Price Algorithm for a multicast routing problem". *Journal of the Operational Research Society*, 50(11):1168–1175, 1999.
- [23] Q. Zhu, M. Parsa, and J. Garcia. "A Source-Based Algorithm for Delay-Constrained Minimum-Cost Multicasting". In *In proceedings of IEEE INFOCOM 95*, pages 353–360, 1995.

- [24] R. Widyono. "The Design and Evaluation of Routing Algorithms for Real-Time Channels". Tech report icsi tr-94-024, University of California at Berkeley, International Computer Science Institute, June 1994.
- [25] B. K. Haberman and G. N. Rouskas. "Cost, Delay, and Delay Variation Conscious Multicast Routing". Technical report, Department of Computer Science, North Carolina State University, march 1997.
- [26] A. Waters. "A new Heuristic for ATM Multicast Routing". *In proceedings of the Second IFIP Workshop on Performance Modeling and Evaluation of ATM Networks*, pages 8.1–8.9, July 1994.
- [27] H. Salama, D.Reeves, Y. Viniotis, and T.L. Sheu. "Comparison of Multicast Routing Algorithms for High-Speed Networks". Technical report, TR 29.1930. IBM, September 1994.
- [28] D. Chakraborty, C. Ponnavaiah, G.Chakraborty, and N.Shiratori. "An Efficient Routing to Minimize the Cost for Dynamic Multicasting". *IEEE Asia Pacific Conference on Circuits and Systems, Bangkok, Thailand*, pages 463–466, Nov 1998.
- [29] M. Kang. "An Optimal Dynamic Multicast Routing Algorithm for Multimedia Applications". *Proceedings of the 1997 International Conference on Multimedia Computing and Systems*, 1997.
- [30] H. Fujinoki and K.J. Christensen. "The New Shortest Path Tree (SBPT) Algorithm for Dynamic Multicast Trees". *Proceedings of the 24th Conference on Local Computer Networks*, 1998.
- [31] Y. Dalal and R. Metcalfe. "Reverse Path Forwarding". *Communications of the ACM*, 21(12):1040–1048, December 1978.
- [32] S. Deering and D. Cheriton. "Multicast Routing in Datagram Internetworks and Extended LANs". *ACM Transactions on Computer Systems*, 8(2):85–110, May 1990.
- [33] F. Glover. "Tabu Search; A tutorial". Technical report, University of Colorado, Boulder, February 1990.
- [34] F. Glover. "Tabu Search and adaptive memory programming- advances, applications and challenges". Technical report, College of Business, University of Colorado at Boulder, 1996.

- [35] Y. M. Sharaiha, M. Gendreau, G. Laporte, and I. H. Osman. "A Tabu Search Algorithm for the Capacited Shortest Spanning Tree Problem". *Networks*, 29:161–171, 1997.
- [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 application"*. Kluwer Academic Publishers, third edition, 1996.
- [38] R. Y. Yager. "On ordered Weighted averaging aggregation operators in Multicriteria Decision Making". *IEEE Transactions on Systems, Man, and Cybernetics*, pages 345–377.
- [39] L. A. Zadeh. "Fuzzy Sets". *Information Contr.*, pages 338–353, 1965.
- [40] E. Shragowitz, H. Youssef, Sadiq M. Sait, and H. Adiche. "Fuzzy Genetic Algorithm for VLSI Floorplan Design". *International Conference on Applications of Soft Computing, SPIE'97*, 1997.
- [41] H. F. Salama et al. *"MCRSIM simulator source code and Users Manual"*. Center for Advanced Computing and Communication, North Carolina State University, Raleigh, 1995. Available by anonymous ftp at <ftp.csc.ncsu.edu/pub/rtcomm>.
- [42] D. Waitzman, C. Partidge, and S. Deering. "Distance Vector Multicast Routing Protocol", November 1988. Internet RFC 1075, <http://ds.internic.net/rfc/rfc1075.txt>.
- [43] L. A. Zadeh. "The concept of a linguistic variable and its application to approximate reasoning". *Information Sciences*, 8:199–249, 1975.

## Vitae

- Muhammed Atif Tahir.
- Born in Karachi, Pakistan.
- Received Bachelor of Engineering (B.E.) degree in Computer Engineering from NED University, Karachi, Pakistan in December 1997.
- Worked as a Computer Programmer in Karachi, Pakistan from January 1998 to August 1999.
- Joined the Department of Computer Engineering at KFUPM as a Research Assistant in September 1999.
- Completed Master of Science (M.S.) in Computer Engineering at KFUPM in July 2001.
- Email: [matiftahir@yahoo.com](mailto:matiftahir@yahoo.com)