GENETIC ALGORITHM APPROACH FOR CONSTRAINED MULTICAST ROUTING IN COMPUTER NETWORKS

By

Moussa Ali Hamdan

Submitted
In partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

Major Subject: Electrical and Computer Engineering Department

At

DALHOUSIE UNIVERSITY

Halifax, Nova Scotia   April, 2004

Contents

List of Figures vii

Acknowledgements xi

Abstract xii

1 Introduction 1

1.1 Multicast . . . 3

1.2 Routing in Computer Networks . . . 4

1.3 Quality of Service Requirements(QoS) . . . 6

1.4 Dissertation Objectives and Outline . . . 7

2 Overview of Multicast Routing Algorithms 9

2.1 Multicast Routing Approaches . . . 10

2.2 Characteristics of Multicast Routing Algorithms . . . 12

2.3 Network Model and Problem Definition . . . 13

2.4 Survey of Multicast Routing Algorithms . . . 15

2.4.1 Shortest Path Algorithms . . . 15

2.4.2 Minimum Steiner Tree Algorithms . . . 19

2.4.3 Dynamic Multicast Routing Algorithms . . . 24

2.4.4 Other Multicast Routing Algorithms . . . 26

2.4.5 Evaluation of Multicast Routing Algorithms 27
2.5 Multicast Routing Protocols 29
2.6 Summary 32

3 Genetic Algorithm For Multicast Routing 33
3.1 Overview of Genetic Algorithms 34
3.1.1 Operation of the GA 35
3.2 The Proposed GA for Constrained Multicast Routing 36
3.2.1 Problem Formulation and Network Model 36
3.3 QoS Metrics 37
3.4 Algorithm description 38
3.4.1 Construction of Routing table 38
3.4.2 Generating The Initial Population 39
3.4.3 Generating the new population 39
3.4.4 Crossover Operation 40
3.4.5 Mutation Operation 40
3.4.6 Steps of the Algorithm 41
3.5 GADVM Parameters Setup 45

4 Performance Evaluation of the Proposed GADVM 49
4.1 Random network generator 49
4.2 Simulated Algorithms 50
4.3 Performance Metrics 51
4.4 Performance with respect to the Delay Variation 52
4.4.1 Failure Rate 52
4.4.2 Average Cost per Path 56
4.5 The Effect of Changing Multicast Group Size on GADVM Algorithm 60
4.5.1 Failure Rate 60
4.5.2 Average Cost per Path 64
4.6 The Effect of Changing Network Size on GADVM Algorithm 68

Your file Votre référence
ISBN: 0-612-93283-4
Our file Notre référence
ISBN: 0-612-93283-4

List of Figures

1.1 Multicast vs.unicast communications. 2

1.2 Multicast group (shown in black circles) in a network 4

3.1 Simple genetic algorithm flowchart 35

3.2 Generating a new population 39

3.3 Crossover and mutation operations 40

3.4 Crossover and Mutation operation 46

3.5 Crossover and Mutation operation 47

3.6 Crossover and Mutation operation 48

4.1 The effect of delay variation constraints on failure rate with network size=50, Multicast group=5, Average node degree = 4 53

4.2 The effect of delay variation constraints on failure rate with network size=50, Multicast group=25, Average node degree = 4 54

4.3 The effect of delay variation constraints on failure rate with network size=50, Multicast group=5, Average node degree = 10 55

4.4 The effect of delay variation constraint on average cost/path rate with network size=50, delay constraint= 0.05, Multicast group=5, Average node degree = 4 57

4.5 The effect of delay variation constraint on average cost/path rate with network size=50, delay constraint= 0.05, Multicast group=5, Average node degree = 4 58

4.6 The effect of delay variation constraint on average cost/path rate with network size=50, delay constraint= 0.05, Multicast group=5, Average node degree = 4 59
4.7 The effect of multicast group constraint on failure rate with network size=50, delay variation constraint= 0.05, delay constraint=0.2, Average node degree = 4 61
4.8 The effect of multicast group constraint on failure rate with network size=50, delay variation constraint= 0.05, delay constraint=0.2, Average node degree = 10 62
4.9 The effect of multicast group constraint on failure rate with network size=50, delay variation constraint= 0.15, delay constraint=0.2, Average node degree = 4 63
4.10 The effect of multicast group constraint on average cost/path rate with network size=50, delay variation constraint= 0.05, delay constraint=0.20, Average node degree = 4 65
4.11 The effect of multicast group constraint on average cost/path rate with network size=50, delay variation constraint= 0.05, delay constraint=0.20, Average node degree =10 66
4.12 The effect of multicast group constraint on average cost/path rate with network size=50, delay variation constraint= 0.05, delay constraint=0.20, Average node degree = 4 67
4.13 The effect of Network size on failure rate with delay variation constraint= 0.02, delay constraint=0.05, Average node degree = 8 69
4.14 The effect of Network size on failure rate with delay variation constraint= 0.04, delay constraint=0.05, Average node degree = 8 70
4.15 The effect of the past solution on the performance of GADVM algorithm with same source and set of destinations 72
4.16 The effect of the past solution on the performance of GADVM algorithm when the destination set is different from the previous one . . . 73

List of Abbreviations

QoS	Quality of Services
GADVM	Genetic Algorithm With Delay And Delay Variations
OSI	Open Systems Interconnection
LC	Least Cost
RPF	Reverse Path Forwarding
TRPB	Truncated Reverse Path Broadcasting
RPM	Reverse Path Multicasting
NP	Non-Deterministic Polynomial
CBF	Bellman-Ford Algorithm
CDKS	Constrained Dijkstra Heuristic
KPP	Kompella, Pasquale, and Polyzos
BSMA	Bounded Shortest Multicast Algorithm
GSDM	Geographic Spread Dynamic Multicast
IGMP	Internet Group Management Protocol
DVMRP	Distance Vector Multicast Routing Protocol
MBone	Multicast Backbone
RIP	Routing Information Protocol
MOSPF	Multicast Open Shortest Path First
OSPF	Open Shortest Path First
PIM-DIM	Protocol Independent Multicast-Dense Mode
CBT	Core Based Tree
QOSPF	Quality-of-Service Open Shortest Path First
GA	Genetic Algorithm

CPT	Shortest Path Tree
CAO	Constrained Adaptive Ordering
LP	Linear Programming
LD	Least Delay
ATM	Asynchronous Transfer Mode
KMB	Kou, Maskowski, and Berman (heuristic)
MSC	Modified Semi-Constrained (heuristic)
PIM-SM	Protocol Independent Multicast-Sparse Mode
RP	Rendezvous Point
RS	Rayward and Smith (heuristic)
SPT	Shortest Path Delay Tree
TM	Takahashi and Matsuyama (heuristic)
VP	Virtual Path

Acknowledgements

I wish to express my deepest admiration and thanks to my thesis advisor, Professor M. El-Hawary for his guidance and help in all aspects of my work. I was fortunate to have him as my advisor. Dr. El-Hawary was never too busy to listen to me and offer his advice. I appreciate the confidence he showed in my abilities and the constant encouragement that he expressed to me. I would also like to thank other members of my advisory committee, Dr. M.Rahman, and Dr, T. Little, for going through this thesis in detail. I would like to thank Carolyn A. Theriault, the graduate studies administrator, for her kindness, help, and patience.

Abstract

Multicast communication has become a key requirement for many applications where one source transmits the same information simultenously to many destinations. The problem of finding a route from the source to other group members is referred to as multicast routing. The main objective of multicast routing is to find a route shaped tree that either has the least *total* cost, which is known as the Steiner tree, or has the least cost for every *path* from source to each destination, which is known as the shortest path tree. Due to the fast evolution of real time and multimedia applications such as audio/video conferencing, interactive distributed games and real time remote control systems, some quality of services, QoS, need to be guaranteed in the underlying network. Multicast routing algorithms should support the required QoS. This thesis presents a constrained multicast routing scheme based on genetic algorithm (GA) by constructing a multicast tree that satisfies two quality of service (QoS) requirements: (i) End-to-End delay and (ii) Delay variation among path delays. The proposed scheme constructs a multicast tree by using a simple mapping technique that does not require a complex transformation in order to obtain a feasible multicast tree, and also scales well for large size networks. A new multicast routing algorithm with constraints based on genetic algorithm, called *Genetic Algorithm With Delay and Delay Variations* GADVM, is proposed and discussed in depth with the flowcharts and pseudo codes of its main subroutines. The proposed algorithm is applied to the problem of multicast routing with delay and delay-variation constraints. The delay variation constraint is a bound on the delay difference between any two destinations. The problem is formulated as one of the shortest path routing under delay and delay variation constraints which is know to be NP-complete. A large number of simulation experiments have been done to analyze the performance of the proposed GADVM algorithm and compare it to some other known multicast algorithms.

Chapter 1

Introduction

With the advances in networking and switching technology and rapid growth of the Internet, many new communication services, high-bandwidth real time and multimedia applications have become reality. Some of these applications require the transformation of multiple copies from a source node to a set of destination nodes in a network. This kind of communication must be supported by a network and is called *multicasting*. *Multicasting* is defined as the ability of a communication network to accept a single message from an application and to deliver copies of the message to many recipients at different locations.

One of the challenges is to minimize the amount of network resources employed by the multicast. To illustrate this point, assume that a video server wants to transmit a movie to 1000 recipients. If the server were to employ 1000 separate point-to-point connections, 1000 copies of the movie may have to be sent over a single link, thus making poor use of the available bandwidth (Fig. 1.1a). An efficient implementation of multicasting results in a much better use of the available bandwidth by sending one copy of the movie on each link in the network (Fig. 1.1b).

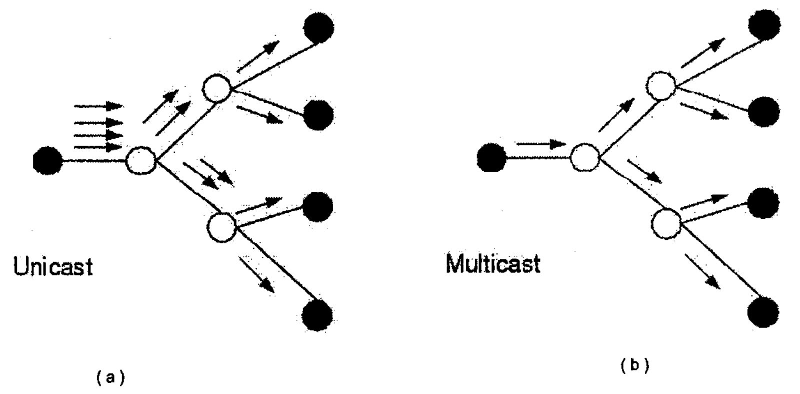

Figure 1.1: Multicast vs.unicast communications.

Computer networks available today were mainly designed to support point-to-point connection(unicast) and therefore many multicast applications are not efficiently implemented, and as more and more multicast applications become popular and bandwidth-intense, there is a growing need to provide an efficient multicasting support.
There are a wide variety of multicast applications , for example:

- Video/telebroadcasts, where one source sends messages to potentially large and geographically spaced set of destinations;
- Video/telelectures, where a small group of sources transmit to potentially large and geographically dispersed set of recipients;
- Video/teleconferencing, where a group of participants where each one concurrently being both a source and recipients of information while participating in a group visual display, discussions, and distributed work environment; and
- Applications that require efficient and regular distribution and update of information (such as databases, files, or digitized images) to group of sites.

1.1 Multicast

Computer networks are constructed from two classes of hardware blocks, *nodes* and *links.* A node can serve as a host that users run applications on, it might be used inside the network as a switch that forwards messages from one link to another, or it can be used as a router that forwards internet packets from one network to another. Network links are implemented on a variety of different physical media, such as twisted pair, coaxial cable, optical fiber, microwaves, and radio waves. Information packets transmitted from a source to a destination are routed through these interconnected nodes. A network is represented by*directed graph* which consists of a set of nodes V and a set of links E. A link connecting a node u to node v is represented by a tuple(u,v). Communication links in a network may have different properties. For example, the fiber optical link may have a very large bandwidth compared to copper wire. A property of a communication link is represented by a *weight* of the corresponding link in a graph. For example, if the propagation delay of the communication link is 1ms this information can be represented by assigning a weight equal to 1 in the link.

The communication links can be of two types: *symmetric* and *asymmetric*. Symmetric links have the same weight in both directions, while asymmetric links have different weights that depend on the direction of transmission.

In one-to-one communications (unicast), this is when the source wants to send a message to single destination node. The routing problem in this case is treated as a *shortest path problem in graphs* and when two nodes wish to communicate, a *minimum weighted path (shortest path)* connecting both nodes is selected.

In multicast communications , when the source node wants to send a message to a subnet of other nodes, but not all of them, and in this case a *minimum weighted tree* which spans over all nodes in the multicast group is constructed. Figure 1.2 shows a network with multicast group connections.

Depending on the application, the tree constructed in multicast communication can be either *Source-specific*, where only one node acts as a source and sends data while all other nodes receive data, or *Group-shared*, where each node can send/receive data to/from other members in the group. For example, applications that involve reliable transfer of data such as software distribution and update, has different requirements from real time multimedia such as videoconferencing.

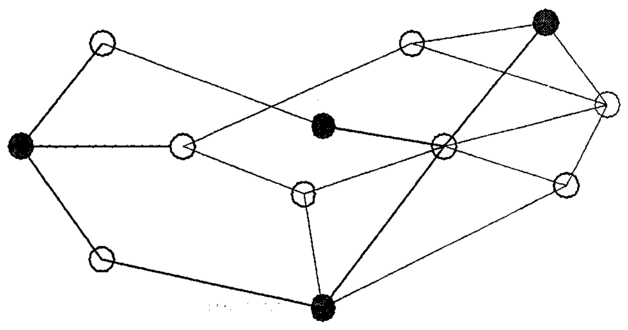
Figure 1.2: Multicast group (shown in black circles) in a network

1.2 Routing in Computer Networks

Routing is the process of determining systematically how to forward messages to their destination nodes based on their addresses. The routing function in computer networks is executed at the network layer and consists of two phases. The first phase is to select a route of connected links during the connection establishment period. The second phase is to make sure that each packet of that session is forwarded along the route selected in phase one [5].
A *routing algorithm* is a way to select routes connecting a set of sources belonging to a given session to the set of destinations of the same session. This dissertation deals only with the route selection mechanisms.

There are three types of routing algorithms based on the type of the communication session :

- A unicast session involves only one source and one receiver. A unicast routing algorithm constructs a path from the source to the receiver.

- Routing algorithms for a one-to-many multicast session: construct a multicast tree rooted at the source and spanning all receivers.
  There are two approaches to the many-to-many multicast routing problem.

  1. Source-specific multicast trees: Construct a one-to-many multicast tree for each source.
  2. Shared multicast trees: Construct only one multicast tree to carry the traffic from any source to any destination.

- Broadcast session, that is when the set of receivers of a given session includes all nodes in the network, then the routing algorithm constructs a *broadcast* tree that spans the entire network. The broadcast and unicast routing problems are special cases of the multicast routing problem. They are usually of lesser complexity than the general problem. In this thesis, only multicast routing algorithms are discussed.

A *routing protocol* describes how to implement a theoretical routing algorithm in practical networks. Protocols must be robust and fault tolerant. For example a protocol is expected to react fast and safely to link or (node) failures, in order to minimize the resulting instability in the network. Similarly, a protocol should be designed such that, if given incorrect or outdated input information, the results will not be disastrous for the applications. The work in this thesis is a study of routing algorithms and not routing protocols. The objective is to study the performance of the proposed genetic algorithm multicast routing with constraints and determine its suitability for real-time communication over high-speed networks. Robustness and fault tolerance are examples of protocol implementation issues that are beyond the scope of this dissertation. Routing protocols usually reside in the network layer of the open systems interconnection (OSI) protocol stack.

1.3 Quality of Service Requirements(QoS)

Real-time applications impose strong requirements on the underlying network. The application's demands are expressed by their quality of service (QoS) parameters such as acceptable end-to-end delay and delay jitter, needed bandwidth, and acceptable loss rate. The QoS parameters usually depend on the traffic streams. For example, video and audio streams can tolerate certain loss rates, but they have stringent end-to-end delay and delay jitter requirements. High bandwidth must be guaranteed in order to accommodate the high transmission rates of real-time video. Data streams require very low loss rates, but their end-to-end delay and delay jitter requirements are not as demanding. The upper bound on end-to-end delay from any source to any receiver and the delay variation among path delays are the two QoS parameter considered in the investigation of various routing problems.

In high-speed wide-area networks, the transmission delay is required to be small and the queuing delay is also required to be small, because small buffer sizes are used [1]. Therefore, the propagation delay is the dominant component of the link delay. The propagation delay is proportional to the distance traversed by the link. It is fixed, irrespective of the link utilization. Therefore, a route selection algorithm can guarantee an upper bound on the end-to-end delay by choosing the appropriate links for the session being initiated, such that the delay from any source to any receiver does not exceed the overall delay bound.

The routing problem studied in this dissertation is formulated as delay-constrained optimization problem, i.e., the upper bound on the end-to-end delay is used as a constraint. The other QoS parameter considered is the delay variation among the paths from source to any two destination nodes.

1.4 Dissertation Objectives and Outline

With the advent of real-time interactive applications, attention has not only been given to minimizing the total tree cost, but also to providing guarantees in terms of end-to-end delay along individual paths from the source to each of the destination nodes. The problem of minimizing the tree cost under the end-to-end delay constraint is known as the delay constrained Steiner tree problem, and it is known as NP-complete. Many heurestic soultions have been proposed for this problem . However, there are certain classes of applications that require, in addition to end-to-end delay bound on each indivitual path, a bound on a *variation* of delay among paths to different destinations of the multicast group. For example, in interactive applications such as video-conferencing, a guarantee on delay variations is needed in order to maintain the feeling of a face-to-face discussion, and all participatns should hear and see the speaker at almost the same time. Another example is an online software update and distribution. It is neccessary that all the remot hosts get a new copy of the software at the same time so that sites never lose integrity while working in the virtual world environment. Finally, minimizing the delay variation may be needed in order to reduce the competitive advantage a destination gains by being able to process a message sooner than the other destinations, as in the case of distributed game applications where the server multicasts information to competing users.

The work in this thesis is motivated by the need for new algorithms for multicast routing that can guarantee certain *Quality of Services* (QoS). Applications like those mentioned above require bounds on both delay and delay variation to be guaranteed. A new multicast routing scheme based on genetic algorithm that takes into account both delay and delay variation when constructing multicast tree is proposed.

Chapter 2 starts with a classification of multicast routing algorithms. Then we survey previous work on multicast routing for communication networks. The chapter concludes with a brief discussion of multicast routing protocols.

In chapter 3, The proposed genetic algorithm for multicast routing is introduced. First, a brief overview of genetic algorithms is given, and then the proposed algorithm is explained in detail. Parameters setting and genetic operations, such as crossover, mutation and selection are described.

In chapter 4, simulations are used to evaluate the performance of the proposed Genetic Algorithm for Delay and Delay Variation Multicast (GADVM) which is applied to the multicast routing problem with delay and delay-variation constraints, and compareed the performance with the other four known multicast routing algorithms.

In chapter 5, conclusions along with directions for future work are given. Finally, we summarize the contributions of the research presented in this dissertation.

Chapter 2

Overview of Multicast Routing Algorithms

The multicast routing problem in communication networks has been the subject of intense research for many years. However, only in recent years, the concept has been used over wide-area networks . For example, the early significant experiment over the Internet was the *audiocast* of the Internet Engineering Task Force(IETF) meeting to 20 sites on three continents in March of 1992 [6]. Many efforts have been under way since then, aimed at both developing the algorithms and protocol levels, and developing multicast routing mechanisms which incorporate the following characteristics:

- Satisfying the QoS requirements of real-time applications,
- Efficient management of the network resources, and
- Application to large network sizes.

This chapter offers a survey of previous work on multicast routing algorithms. Sections 2.1 and 2.2, present various classifications of multicast routing algorithms and discuss some important criteria for surveying previous work. section 2.3 gives a problem definition and a mathematical model of a communication network. Then section 2.4 surveys previous work on multicast routing algorithms. Section 2.5, provides a brief overview of multicast routing protocols. Finally, summary and concluding remarks are given in section 2.6.

2.1 Multicast Routing Approaches

There are two main approaches for multicast routing:

1. The *shortest path* algorithms where the objective is to find a tree that minimizes the length of each route from the source node to each node in the multicast group.

2. The *minimum Steiner tree* algorithms. In this case the objective is to minimize the overall cost of the multicast tree. This problem is known to be NP-hard [7].

A special case of minumum steiner tree is a *minimum spanning tree*, where all nodes in the network become receivers/members in the multicast tree as is the case for example in broadcast applications. The minimum spanning tree is solvable in polynomial time [8].

Real-time applications require certain QoS guarantees. Network algorithms and protocols must be designed to support such QoS. For example, distributed multimedia applications require a guaranteed upper bound on the end-to-end delay, and it is sufficient for the network to satisfy this bound and there is no need to minimize the end-to-end delay.

Routing algorithms designed specifically for high-speed networks construct an efficient multicast tree without violating the constraint implied by the upper bound on delay. These are called delay-constrained algorithms, to distinguish them from other algorithms which are unconstrained. In many cases, the network nodes have a limited copying capability, i.e., there is an upper limit on the number of copies of an incoming packet which the node can forward simultaneously to the next hop nodes on the multicast tree. This is known as the degree constraint, and algorithms that consider this problem are called degree-constrained multicast routing algorithms.

Dynamic multicast routing algorithms permit sources and receivers to join and leave a multicast session and the corresponding multicast trees at any time. In static multicast routing algorithms, however, the multicast group is fixed, and paths from the sources to all receivers are computed at the same time when initiating a multicast session.

In *distributed* multicast routing algorithms, the computations required to construct a multicast tree are shared among multiple nodes. This reduces the computational overhead at each node but requires messages to be exchanged between the nodes. The complexity of these algorithms is measured by the number of messages exchanged. The amount of information about the state of the network that must be stored at each node is another factor affecting the practicality of a distributed algorithm.

Centralized multicast routing algorithms are usually more stable than the distributed algorithms. However, complete network topology information must be available at any node running the centralized algorithm. As has been mentioned already in section 1.2, multicast trees can be classified into source-specific trees and shared trees. The construction of a shared multicast tree consists of two parts: the *center* selection part and the *route* selection part. On the other hand, the construction of a source-specific multicast tree requires only the selection of proper routes. The focus in this chapter is on route selection only.

2.2 Characteristics of Multicast Routing Algorithms

The following criteria is considered when summarizing the features and performance of the different multicast routing algorithms.

- End-to-end delay performance. A measure of the suitability of the algorithm for real-time applications imposing delay constraints.
- Network management efficiency. The ability of an algorithm to manage the network bandwidth and buffer space efficiently. The cost of a link is frequently defined as a function of the utilized link bandwidth. We therefore refer to the efficiency of an algorithm in managing the network bandwidth as the cost performance of the algorithm.
- Complexity of the algorithm. The algorithm's complexity together with the possibility of distributed implementation are major factors in determining the algorithm's scalability to large network sizes.
- Symmetric/asymmetric networks. The multicast routing problems in symmetric networks are less complex than equivalent problems in asymmetric networks. Furthermore, an algorithm designed with the assumption that the network is symmetric is not guaranteed to perform well when applied to asymmetric networks, even if it performs well with symmetric networks.

The performance of multicast routing algorithms is usually evaluated using simulations. Researchers used different assumptions and created different scenarios when evaluating multicast routing algorithms. Some of the major differences between these scenarios are listed below.

- The network topology used, e.g., real networks, randomly generated networks, or mesh networks.
- The size of the networks used.

- The link cost and link delay functions used. The link cost may be some monetary cost or a function of the link's utilization, or, in some cases, unity cost is assigned to all links. Some researchers even defined a link's cost to be equal to its delay. The link delay may represent only the queuing component of the link's delay or only the propagation component of the link's delay or both.

If researchers evaluation of an algorithm's performance is based on simulation of special case scenarios, such as mesh networks, then it is not appropriate to generalize the results to all networks. It is therefore important to take the evaluation scenarios into account when surveying and evaluating previous work on multicast routing algorithms.
The literature survey, given in section 2.4, uses the classification of section 2.1 and the criteria listed earlier to differentiate between the various algorithms. Before proceeding to the survey itself, we present some definitions.

2.3 Network Model and Problem Definition

A point-to-point communication network is represented as a directed, connected, simple network $N = (V, E)$, where V is a set of nodes and E is a set of directed links. The existence of a link $e = (u, v)$ from node u to node v implies the existence of a link $e^{\text{`}} = (v, u)$ for any u ; $v \in V$, i.e., full duplex in networking terms. A link $(u; v) \in E$ is an outgoing link for node $u \in V$ and an incoming link for $v \in V$. Any link $e = (u, v) \in E$ has a cost $C(e)$ (the same as $C(u, v)$) and a delay $D(e)$ (the same as $D(u, v)$) associated with it. $C(e)$ and $D(e)$ may take any nonnegative real values. The link cost $C(e)$ may be either a monetary cost or some measure of the link's utilization.
The link delay D(e) is a measure of the delay a packet experiences when traversing the link e. Thus it may consist of queuing, transmission, and propagation components. Because of the asymmetric nature of computer networks, one often finds that

$C(e) \neq C(e^{‘})$ and $D(e) \neq D(e^{‘})$. If the network is symmetric, it can be represented as an undirected network in which $C(e) = C(e^{‘})$ and $D(e) = D(e^{‘})$ for all $e \in E$. A path is defined as an alternating sequence of nodes and links $P(v_0, v_k) = v_0, e_1, v_1, e_2, v_2, ...v_{k-1}, e_k, v_k$, such that every $e_i = (v_{i-1}, v_i) \in E, 1 \leq i \leq k$. A path contains loops if some of its nodes are not distinct. If all nodes are distinct, then the path is loop-free.

In the remainder of this dissertation, it will be explicitly mentioned if a path contains loops. Otherwise a path always denotes a loop-free path. We will use the following notation to represent a path: $P(v_0, v_k) = v_{\rightarrow}v_1 \rightarrow .... \rightarrow v_{k-1} \rightarrow v_k$. The cost of a path $P(v_0, v_k)$ is defined as the sum of the costs of the links constituting $P(v_0, v_k)$

$$Cost(P(v_0, v_k)) = \sum_{e \in P(v_0, v_k)} C(e) \tag{2.1}$$

Similarly, the end-to-end delay along the path $P(v_0, v_k)$ is defined as the sum of the delays on the links constituting $P(v_0, v_k)$

$$Delay(P(v_0, v_k)) = \sum_{e \in P(v_0, v_k)} D(e) \tag{2.2}$$

The definitions given below apply to a multicast session with a single source. A multicast group $G = g_1, ...g_n \subseteq V$, where $n = |G| \leq |V|$, is a set of nodes participating in the same network activity, and is identified by a unique group address i, and $|G|$ denots the cardinality of the set G. A node $s \in V$ is a multicast source for the multicast group G. A multicast source s may or may not be itself a member of the group G. A source-specific multicast tree $T(s, G) \subseteq E$, is a tree rooted at the source s and spanning all members of the group G. The total cost of a tree $T(s, G)$ is simply the sum of the costs of all links in that tree.

$$Cost(T(s, G)) = \sum_{e \in T(s,G)} C(e) \tag{2.3}$$

In general, an algorithm that minimizes the total cost of a multicast tree will encourage the sharing of links. The maximum end-to-end delay of a multicast tree is the maximum delay from the source to any multicast group member.

$$Max - Delay(T(s,G)) = \max(\sum_{e \in P_T} (P_T)_(s,g) D(e)) \tag{2.4}$$

where $P_T(s,g)$ is the path from s to g along the tree $T(s,g)$.

2.4 Survey of Multicast Routing Algorithms

This section provides a survey of different multicast algorithms and group them according to the classification provided in the previous section. Only distinguishing features of each algorithm are presented without listing the pseudo code. For simplicity, we consider only source-specific trees when describing the features of an algorithm. However, many of the algorithms surveyed are also applicable for the construction of shared trees.
The focus of this dissertation is on the ability of multicast routing algorithms to satisfy the delay constraints of real-time applications. Due to the large amount of work reported in the literature on multicast routing problems, we cannot survey all of them in detail. So we focus primarily on static multicast delay constrained routing algorithms. Such algorithms will be covered in more detail in the survey than the other classes of multicast routing algorithms.

2.4.1 Shortest Path Algorithms

As the name indicates, a shortest path algorithm minimizes the sum of the lengths of the individual links on each individual path from the source node to a multicast group member.

The properties of a shortest path tree depend on the metric the link length represents. If unit link lengths are used, the resulting shortest path tree is called a minimum-hop tree. If the link length is set equal to the link cost, then a shortest path algorithm, denoted as the least-cost (LC) algorithm in this case, computes the LC tree. The objective of an LC algorithm can be expressed mathematically as follows

$$\min_{e \in T(s,G)} Cost(P_T(s,g)),\ \forall g \in G \tag{2.5}$$

The total cost of an LC tree is not necessarily optimal. If the length of a link is a measure of the delay on that link, then a shortest path algorithm, denoted as least-delay (LD) algorithm in this case, computes the LD tree. The objective function of an LD algorithm is to

$$\min_{e \in T(s,G)} Delay(P_T(s,g)),\ \forall g \in G \tag{2.6}$$

An LD tree is optimal with respect to end-to-end delay. For real-time applications, if the LD tree cannot satisfy the imposed delay constraint, no other multicast tree can.

The Bellman-Ford algorithm [11] and the Dijkstra algorithm [12] are two well known shortest path algorithms. Both algorithms are exact and run in polynomial time. The worst case time complexity of the Bellman-Ford algorithm is $O(|V|^3)$ where $|V|$ is the number of nodes in the network. An exact, distributed version of the Bellman-Ford algorithm is given in [5]. It requires only limited information about the network topology to be kept at each node. Awerbuch et al. [13] show that the worst case message complexity of the exact, distributed Bellman-Ford algorithm may grow exponentially with the number of nodes. To avoid this excessive complexity, they propose two approximate distributed versions of the algorithm. For Dijkstra's shortest path algorithm, only centralized versions exist. Its execution time is $O(|V|^2)$

time in the worst case. Efficient, nondistributed versions of both Bellman-Ford and Dijkstra's algorithms have comparable average running times [14]. Both algorithms remain exact for asymmetric networks.

The reverse path forwarding (RPF) algorithm proposed by Dalal and Metcalfe [15] is an algorithm for broadcasting in datagram networks. Each packet is forwarded from the source to the receivers over the reverse shortest paths, i.e., the shortest paths from the receivers back to the source. Thus RPF creates an optimal shortest path broadcast tree only if the network is symmetric.
Deering [4, 16] generalized the RPF algorithm to the multicast case by presenting the truncated reverse path broadcasting (TRPB) algorithm and the reverse path multicasting (RPM) algorithm. The objective function of TRPB and RPM can be stated as

$$\min_{e \in T(s,G)} \sum_{\in P_T(s,g)} C(\acute{e}),\ \forall g \in G \tag{2.7}$$

where $e = (u : v)$ and $\acute{e} = (v : u)$. TRPB and RPM do not suffer from some of the limitations which RPF suffers from with respect to its applicability to multi-access networks. RPF, TRPB, and RPM are distributed algorithms that rely on limited information at each node in the network. They scale well with the size of the network, and dynamic implementations of these algorithms exist.

The shoretst path Algorithms presented above are unconstrained algorithms. ***Delay-Constrained Shortest Path Algorithms*** minimize the cost of each path, i.e., the sum of the link costs, from the source node to a multicast group member subject to an end-to-end delay constraint. Thus the tree is a delay-constrained LC tree. An algorithm for solving the delay-constrained LC problem has the same objective function as that of the unconstrained LC problem, stated in equation 2.5, with the added constraint that

$$Max - Delay(T(s, G)) \leq \Delta \tag{2.8}$$

where Δ is the value of the imposed delay constraint. The delay-constrained shortest path problem is NP-hard [17]. Algorithms for solving that problem were proposed recently, motivated by the increasing importance of end-to-end delay as a QoS constraint for real-time applications. The distinguishing characteristics of each algorithm are summarize below. Note that the delay-constrained multicast routing algorithms surveyed in this section and in the next section are only applicable for the construction of source-specific trees.

Widyono [18] presented the constrained Bellman-Ford (CBF) algorithm. CBF performs a breadth-first search to find the delay-constrained shortest path tree. CBF is optimal and therefore its running times grow exponentially with the size of the network. Widyono used CBF as a basis for several delay-constrained minimum Steiner tree heuristics which will be surveyed in the next section.

Sun and Langendoerfer [19] proposed a delay-constrained shortest path heuristic. It is called the constrained Dijkstra heuristic (CDKS) because it is based on Dijkstra shortest path algorithm. This heuristic computes an unconstrained LC tree. If the end-to-end delay to any group member violates the delay constraint, the path from the source to that group member is replaced with the LD path. Thus if the LC tree violates the delay constraint, an LD tree must be computed, and the two trees are merged. This algorithm always finds a constrained multicast tree if one exists. CDKS runs in $O(|V|^2)$ time, the same as Dijkstra's algorithm. The authors compared the cost performance of their heuristic to LD and KPP (a delay-constrained minimum Steiner tree heuristic which will be presented in the next section) using simulation over random networks. They used unit link costs and integer link delays ranging in value from 1 to 5.

Wi and Choi [20] presented a distributed LD algorithm to use for solving the delay-constrained shortest path problem. They used simulations to evaluate its performance and execution times relative to KPP for 20-node symmetric networks.

2.4.2 Minimum Steiner Tree Algorithms

The objective of the minimum Steiner tree problem is to minimize the total cost of the multicast tree, i.e.

$$\min_{e \in T(s,G)} Cost(T(s,G)) \tag{2.9}$$

This problem is known to be NP-complete [7].

Hwang [21] provided an extensive survey of both exact and heuristic minimum Steiner tree algorithms. Another survey was given by Winter [22]. Few algorithms have been proposed for the minimum Steiner tree problem in asymmetric networks, and all of them operate under special assumptions, e.g. acyclic networks. If the multicast group includes all nodes in the network, the minimum Steiner tree problem reduces to the minimum spanning tree problem. The minimum spanning tree problem in symmetric networks can be solved in $O(|V|^2)$ time in the worst case using Prim's algorithm [8]. Unconstrained minimum Steiner tree algorithms do not attempt to optimize the end-to-end delay at all. Therefore they may not be suitable for real-time applications.

The best known minimum Steiner tree heuristics were proposed by Kou, Markowski, and Berman (KMB heuristic) [23], Takahashi and Matsuyama (TM heuristic) [24], and Rayward- Smith (RS heuristic) [25].

The KMB heuristic [23] uses Prim's minimum spanning tree algorithm [8] during its computation. Prim's algorithm is optimal only for symmetric networks. Thus the cost performance of the KMB heuristic may be affected if it is applied to asymmetric networks. The worst case time complexity of the KMB heuristic is $O(|G||V|^2)$, where $|G|$ is the size of the multicast group.

Wall [26, 27] proposed a distributed version of the KMB heuristic. The total cost of trees generated using KMB heuristic in symmetric networks is on the average only 5% worse than the cost of the optimal minimum Steiner tree [28, 29]. The TM heuristic [24] starts with a tree that contains the source node only. Then it adds the multicast group members, one at a time, to the existing tree via the cheapest LC path to any node already in the tree. TM heuristic runs in $O(|G||V|^2)$ time in the worst case. The RS heuristic [25] starts with a forest of trees, with each multicast group member constituting a tree included. Then the heuristic unites trees that are closest to each other (in terms of cost) by adding the appropriate links until it ends up with a single tree.

Using a limited number of simulations, Rayward-Smith and Clare [30] showed that RS heuristic yields tree costs that are closer to optimal than KMB and TM heuristics. Unfortunately, however, the RS heuristic was designed for symmetric networks, and we cannot envision an efficient method for implementing it in case of asymmetric networks. Jiang [31] presented modified versions of KMB heuristic and RS heuristics that construct multicast trees with lower costs than the original heuristics. Jiang simulated a symetric network model with random link connections and heterogeneous link capacities. He defined the link cost as function of the utilized link bandwidth. The same author also proposed a distributed minimum Steiner tree heuristic in [32].

Ramanathan [33] proposed a heuristic for constructing minimum Steiner trees in asymmetric networks. This heuristic permits trading off low tree cost for fast execution time by proper selection of a parameter k. The author showed that Dijkstra shortest path algorithm, KMB minimum Steiner tree heuristic, and TM minimum Steiner tree heuristic are particular cases of his proposed heuristics when k is set to 1, $(|G| + 1)$, and $|V|$ respectively. Many other heuristics for constructing minimum

Steiner trees in communication networks were proposed. See for example Chow [33], Leung and Yum [35], and Bauer and Varma [36].

As with the shortest path tree algorithms, we now survey a Delay-Constrained Minimum Steiner Tree Algorithms.
The ***delay-constrained source-specific minimum Steiner tree*** problem was first formulated by Kompella, Pasquale, and Polyzos [37, 38]. The authors proved the NP-completeness of the problem. The objective of the problem is to minimize the total cost of the tree, equation 2.8, without violating the imposed delay constraint, equation 2.9. Optimal algorithms for this problem exist. For example, Noronha and Tobagi [39] proposed an algorithm, based on integer programming, which constructs the optimal source-specific delay-constrained minimum Steiner trees for multiple multicast sessions simultaneously. However, this algorithm is rather complex and is useful only as a reference to evaluate heuristic solutions for the same problem.

The first heuristic for the delay-constrained minimum Steiner tree problem was given by Kompella, Pasquale, and Polyzos [37, 38]. this is labeled as the KPP heuristic. KPP assumes that the link delays and the delay constraint Δ are integers, while the link costs may take any positive real value. The heuristic is dominated by computing a constrained closure graph which takes time $O(\Delta|V|^3)$. Thus KPP takes polynomial time only if Δ has a fixed value. When the link delays and Δ take noninteger values, Kompella et al. propose to multiply out fractional values to get integers. Following this approach, KPP is guaranteed to construct a constrained tree if one exists. However, in some cases the granularity of the delay constraint becomes very small, and hence the number of bits required to represent it increases considerably. As a result the order of complexity, $O(\delta|V|^3)$, may become too high. To avoid prohibitively large computation times, a fixed granularity may be used. However, fixing the granularity has side effects. When the granularity is comparable to the average link delays, KPP's accuracy is compromised and in many cases it fails to construct a constrained multicast tree when one exists. The authors proposed two alternative objective functions for KPP to use during tree construction. The first is a function of the link cost only. The second objective function is a function of both the link cost and the residual delay if this link is added to the tree. The authors used simulation of random, symmetric networks with up to 100 nodes to evaluate their heuristic procedure.

Similar to KMB, KPP uses Prim's algorithm [8] to obtain a minimum spanning tree of a closure graph. Prim's algorithm is only optimal for symmetric networks. This might affect the performance of KPP when applied to asymmetric networks. Kompella, Pasquale, and Polyzos also proposed a distributed heuristic solution for the delay-constrained minimum Steiner tree problem [40]. The heuristic is based on Prim's algorithm [8], but it involves the making and breaking of cycles during the construction of the multicast tree. It runs in $O(|V|^3)$ time, and is guaranteed to find a multicast tree, if one exists.

Widyono [34] proposed four delay-constrained minimum Steiner tree heuristics. The four delay-constrained heuristics are based on the CBF algorithm described in the previous section. Therefore all of them have worst case scenarios with exponentially growing execution times. Widyono's constrained adaptive ordering (CAO) heuristic yields better performance than the other three constrained heuristics he proposed. In CAO, the CBF algorithm is used to connect one group member at a time to the source. After each run of CBF, the unconnected member with the cheapest constrained LC path to the source is chosen and is added to the existing subtree. The costs of links in the already existing subtree are 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 CBF conducts. Widyono defined the link cost as a function of the available bandwidth, the residual buffer space, and the link's delay. The link delay was defined as the sum of the queuing, transmission, and propagation delays along the link. The author evaluated his heuristics using simulation of eight by eight mesh networks.

The bounded shortest multicast algorithm (BSMA) was proposed by Zhu, Parsa, and Garcia-Luna-Aceves [41]. BSMA starts by computing an LD tree for a given source and multicast group. Then it iteratively replaces superedges in the tree with cheaper superedges not in the tree, without violating the delay constraint, until the total cost of the tree cannot be reduced any further. BSMA uses a kth-shortest path algorithm to find cheaper superedges. It runs in $O(k|V|^3 log|V|)$ time. For large, densely connected networks, k may be very large, and it may be difficult to achieve acceptable running times. It is possible to trade off multicast tree cost for fast execution speed when using BSMA by either limiting the value of k in the kth-shortest path algorithm or by limiting the number of superedge replacements. BSMA always finds a constrained multicast tree, if one exists, because it starts with an LD tree. The authors defined the link cost as a function of the link utilization and defined the link delay as the sum of the queuing delay, transmission delay, and propagation delay over the link. They evaluated the performance of BSMA and compared it to KMB and LD. Random networks with up to 100 nodes generated using Waxman's random network generator [42] were used.

The minimum Steiner tree heuristic proposed by Waters [43] is considered to be semi-constrained, because it uses the maximum end-to-end delay from the source to any node in the network (not to any group member) as the delay constraint. Note that 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 tree is pruned beyond the multicast nodes. this is a semi-constrained (SC) heuristic. In [44], they implemented original version of the algorithm proposed in [43] which resembles a semi-constrained minimum spanning tree, and also implemented a modified version which is closer to a semi-constrained shortest paths broadcast tree. Simulation results given in [44] showed that the modified version, denoted as the modified semi-constrained (MSC) heuristic always performs better than the original heuristic with respect to tree costs, end-to-end delays, and network balancing. SC and MSC are dominated by the computation of the internal delay bound. This computation uses an extension to Dijkstra's algorithm, and therefore it takes $O(V^2)$ time in the worst case.

In addition to the algorithms surveyed above, many other variations of the multicast routing problem have been studied over the years. Research reports on the dynamic multicast routing problem, in particular, appeared frequently in the literature. We dedicate the next section to previous work on this problem. Then, in section 2.4.6, we survey previous work on other variations of the multicast routing problem.

2.4.3 Dynamic Multicast Routing Algorithms

Dynamic multicast routing algorithms were proposed to avoid rerouting an entire multicast tree whenever a node joins or leaves a multicast session. In dynamic multicast routing algorithms, when a node leaves a multicast session, the path connecting that node is simply pruned from the tree if it is not used to connect any other multicast group members. The situation is more difficult when a node joins an existing multicast session.

Waxman [42, 45] presented a greedy dynamic multicast routing algorithm. The algorithm has a weighting parameter w that varies from 0 to 0.5. When $w = 0$, a node joins an existing source-specific multicast tree via the shortest path to the tree. When $w = 0.5$, the node is added to the existing tree via the shortest path to the source. Waxman evaluated his algorithm using simulation over randomly generated 56-node and 60-node networks. He proposed an algorithm for generating random networks that resemble realistic networks. This random network generator has been adopted by many researchers since then.

Doar and Leslie [24] investigated a naive (simple) approach that always connects a joining node to the existing tree via the shortest path from the source. They simulated this mechanism using randomly generated networks, both flat and hierarchical. Their random network generator is a modified version of Waxman's generator. Simulations over 100-node networks showed that the naive approach constructs trees that are on the average 50% more expensive than costs of trees constructed using the static KMB heuristic.

Kadirire [46] defined the geographic spread as the shortest distance from any node in the network to the existing tree averaged over all nodes not in the tree. He proposed a geographic spread dynamic multicast (GSDM) algorithm that maximizes the geographic spread for the multicast tree it constructs. Kadirire also evaluated the performance of GSDM and compared it to Waxman's heuristic and Doar and Leslie's heuristic in [47] using simulation over random networks with up to 100 nodes. He showed that GSDM and Waxman's heuristics yield similar performance and are consistently better that Doar and Leslie's naive approach.

Biersack and Nonnenmacher [48] proposed a dynamic, distributed multicast routing algorithm named WAVE. WAVE uses a weighted function of the cost and the delay to attach a joining node to the existing tree. The authors evaluated their algorithm in comparison to static algorithms only.

Bauer and Varma [49] presented a dynamic multicast routing algorithm: ARIES. The operation of ARIES is similar to Waxman's dynamic algorithm. In addition, however, a subtree of the multicast tree is completely reconstructed each time a prespecified number of joins and leaves affects that subtree. The subtree reconstruction ensures that the cost of the multicast tree remains close to optimal. ARIES was evaluated using simulation over 200-node random networks. The authors used a modified version of Waxman's random network generator.

2.4.4 Other Multicast Routing Algorithms

This subsection briefly surveys a few more multicast routing algorithms that do not belong to any of the categories listed in the previous subsections.

Bharath-Kumar and Jaffe [50] presented a tradeoff algorithm between the mini mum Steiner tree and the LD tree. This algorithm constructs the minimum Steiner tree; then it locates the receiver with the largest difference between the delay along its path in the minimum Steiner tree and the delay along the LD path from the source to that receiver. The algorithm then replaces the minimum Steiner tree path with the corresponding LD path. The same authors also proposed two distributed multicast routing heuristics which are based on local information from nearby nodes only.

Rouskas and Baldine [51] studied the problem of constructing multicast trees subject to both an end-to-end delay constraint and a delay variation constraint. They defined the delay variation constraint as the maximum difference, that can be tolerated, between the end-to-end delays along the paths from the source to any two receivers. The authors proved that this problem is NP-complete; then they proposed a heuristic solution. Research on the degree-constrained multicast routing problem is motivated by the fact that current multicast capable high-speed switches have limited copy capability. In addition, limiting the maximum degree at any node in the multicast tree results in more evenly distributed load among all nodes in the network.

Tode et al. [52] proposed two algorithms for degree-constrained multicast routing. The first algorithm minimizes the average degree of the multicast tree it constructs, while the second algorithm attempts to construct a low-cost multicast tree subject to a given maximum degree constraint. The authors set the link costs equal to the link delays when evaluating the performance of their heuristics.

Bauer and Varma [53] investigated a variation of the degree-constrained multicast routing problem in which the degree-constraint may vary for different nodes in the network. Using simulation, they showed that many of the existing unconstrained minimum Steiner tree heuristics are capable of constructing degree-constraint multicast trees. The authors also proposed a simple degree-constrained heuristic which performs better than all other algorithms of the same or lesser complexity.

Ammar et al. [54] studied the problem of routing virtual paths (VP) for multicast communication in ATM networks. When constructing a multicast tree, they took into account the bandwidth cost, the switching cost, and the connection establishment cost. The authors studied different types of VPs. They formulated the problem as an integer programming problem and proposed heuristic solutions based on the transshipment simplex algorithm. The authors used a single 16-node network for evaluating their heuristics.

Kim [55] studied a problem similar to that in [54]. He proposed an optimal solution to the problem of routing multiple multicast connections simultaneously in ATM networks.

2.4.5 Evaluation of Multicast Routing Algorithms

It is evident from the previous sections that a large number of algorithms have been proposed for many variations of the multicast routing problem. Researchers used different networking environments and made different assumptions when evaluating the algorithms they proposed. An analytical study of the tradeoffs between shortest path trees and minimum Steiner trees was reported by Bharath-Kumar and Kadaba [50] in 1983. The authors did not distinguish between link cost and link delay, because, at the time, QoS issues and resource management issues were not well defined yet.

Tanaka and Huang [56] compared the performance of several static unconstrained minimum Steiner tree algorithms. In addition, they evaluated one dynamic algorithm, the weighted greedy algorithm [42, 45]. The authors ran simulations of a single 20-node symmetric network with the cost of a link being proportional to the distance spanned by that link.

Wei and Estrin [57] studied the LD algorithm and the KMB heuristic for constructing minimum Steiner trees. The authors simulated 50-node and 200-node random networks generated using Waxman's random network generator. The networks were asymmetric, and each link had a delay (equal to its length) and a cost assigned to it. Simulation of 50-node networks with an average node degree of 4 showed that minimum Steiner trees are lower in cost than least-delay trees by approximately 20%. However, the maximum end-to-end delays along minimum Steiner trees are up to 60% larger than those along least-delay trees.

Noronha and Tobagi [58] reported an evaluation of multicast routing algorithms for real-time applications . They evaluated three unconstrained algorithms, the LD algorithm, the LC algorithm, and the KMB heuristic, with respect to their suitability for real-time applications with delay constraints. The authors used the optimal delay-constrained algorithm which they presented in [39] to benchmark the algorithms. The link cost represented a monetary cost, while the link delay represented the actual delay along the link. The following admission control policy was enforced: the sum of the bandwidths of the multicast sessions utilizing a link cannot exceed the link's capacity. The authors defined the blocking probability as the probability that an algorithm fails to construct a multicast tree. There are two causes of failure: delay constraint violation and insufficient bandwidth to support the multicast session. Simulations were run on real networks as well as on random networks. Simulation results showed that, in the absence of a delay constraint, KMB heuristic is almost as good as optimal with respect to tree cost and blocking probability. When a delay constraint is enforced, however, the blocking probability of KMB is higher than those of the other algorithms studied. KMB has up to 20% blocking probability in scenarios in which no other algorithms fail. LD and LC have comparable blocking probabilities in all scenarios. The same authors also experimented with different randomly generated network topologies, and they concluded that two-connected random topologies yield simulation results that are closest to the results obtained from simulation of real networks. In real networks, short links are more likely to exist than long ones. However, the authors found that it is not necessary to bias the random network generator towards short links in order to get random topologies which yield similar performance to that of real networks.

2.5 Multicast Routing Protocols

Efforts to develop multicast routing protocols for wide-area networks have started in the late 1980s motivated by the rapid growth of the Internet and the emergence of new applications involving multiple users.

Semeria and Maufer [59] provided an introduction to IP multicast routing. IP Multicast routing protocols adopt the host group addressing model [3, 4]. The Internet Society designated Class D IP addresses for multicast group addressing.

The Internet Group Management Protocol (IGMP) [60,61] is used to exchange group membership information on a local subnetwork. We will focus only on multicast routing in wide-area networks. An in depth investigation of the complete IP multicast architecture can be found for example in [62].
Initially, there were two standard protocols for IP multicast routing. To avoid some of their shortcomings, two additional protocols were developed and standardized.
The Distance Vector Multicast Routing Protocol (DVMRP) [63] is the first standard

protocol for IP multicast routing. It is widely implemented in commercially available routing equipment, and it is the protocol used in most routers of the Internet's Multicast Backbone (MBone) [2, 64] which currently spans thousands of nodes on all continents.

DVMRP is based on the TRPB heuristic [4, 16]. It is a distributed protocol that uses the limited information available in the distance vectors of the Routing Information Protocol (RIP) [65, 66] to forward datagram packets from the source to all receivers over the reverse shortest paths. DVMRP uses source-specific multicast routing, and it allows receivers to dynamically join and leave a multicast session. In order to discover new members in a multicast session, DVMRP periodically sends the source's packets over a broadcast tree to all nodes in the network. Then leaf nodes which are not members of the multicast group send prune messages upstream towards the source to prune the links leading to these nodes from the tree. The occasional broadcasting behavior of DVMRP causes inefficient use of the network bandwidth, especially if the size of the multicast group is small. This limits the scalability of DVMRP to larger networks. A more efficient version of DVMRP is also developed [67].

The Multicast Extensions to OSPF (MOSPF) [68, 69, 70] is a standard multicast routing protocol for interior gateway routing, i.e., within a single domain. As the name indicates, MOSPF is based on the Open Shortest Path First (OSPF) [71] unicast routing protocol. MOSPF uses the centralized Dijkstra algorithm to construct the forward shortest path multicast tree. To achieve this, a node that runs MOSPF must maintain complete information about the network topology. Therefore complete topology information must be periodically broadcast to all MOSPF-capable nodes in the networks. The centralized nature of MOSPF as well as the necessary periodical broadcast operations severely limit the scalability of the protocol. In addition to the limitations mentioned above, both DVMRP and MOSPF rely on specific unicast routing protocols, RIP and OSPF respectively. Thus they cannot be deployed on routers not running these unicast protocols. Two new multicast routing protocols were then developed to avoid the shortcomings of DVMRP and MOSPF. The two new protocols are independent of the underlying unicast routing mechanisms.

The first protocol currently being developed is Protocol Independent Multicasting (PIM) [72]. PIM specifies a dense mode (PIM-DM) [73] and a sparse mode (PIM- SM) [74]. When a multicast group densely populates an internetwork, PIM-DM is used to create source-specific multicast trees. The basic operation of PIM-DM is very similar to that of DVMRP, but it is independent of the underlying unicast routing protocol. PIM-SM is specified for multicast groups where members are sparsely distributed over an internetwork. PIM-SM uses rendezvous points (RP) which are central nodes at which receivers can meet sources of the same multicast session. A shared tree is constructed around the RP. Receivers join the shared tree via the forward shortest paths towards the RP, and sources transmit to the shared tree via the forward shortest paths towards the RP. Thus packets are forwarded over the reverse shortest paths from the RP to the receivers. A receiver r discovers the existence of a source s when it receives that source's packets over the shared tree. Then the receiver r can elect to continue receiving packets from source s over the shared tree, or, alternatively, it can join that source's specific tree via the reverse shortest path from that s to r. Thus PIM-SM permits the use of both shared trees and source-specific trees. Routing for the same multicast session can use a mixture of shared trees and source-specific trees. Similar to DVMRP, PIM relies on a soft state refreshment mechanism, and thus it suffers from periodical message overhead to maintain the multicast trees. However, soft state enhances the robustness of PIM.

Core Based Trees (CBT) [75, 76, 77, 78] is the other protocol proposed for multicast routing over the Internet. In CBT, all sources sending to the same group use a single multicast tree to carry their traffic to all receivers belonging to that group.

The multicast tree has one or more cores. Similar to the shared mode of PIM-SM, a new receiver joins an existing core based tree via the forward shortest path towards a core, and therefore packets are forwarded on the tree along the reverse shortest paths from the cores to the receivers. The cores are interconnected via a core backbone. CBT relies on an explicit reliability mechanism to maintain the multicast tree. This mechanism introduces much less message overhead than that introduced by PIM's soft state mechanism.

Both PIM and CBT are expected to scale well to large networks. Interoperability with other multicast routing protocols is being considered in the specifications of both PIM and CBT. A quantitative comparison of the two protocols and suggestions to improve their performance can be found in [79]. Work on QoS routing has started only recently, motivated by the need for routing algorithms capable of providing QoS guarantees. Quality of Service extensions to OSPF (QOSPF) [80] is a protocol capable of routing both unicast and multicast connections and reserving bandwidth and other resources for those connections.

2.6 Summary

In this chapter, we classified the multicast routing algorithms into different categories based on the problems they address. Then we presented important criteria to be considered when summarizing the features of a multicast routing algorithm. The bulk of this chapter was dedicated to surveying previous work on multicast routing algorithms. Over the years, a lot of algorithms have been proposed for many variations of the multicast routing problem. However, different researchers have made different assumptions when evaluating the performance of the algorithms they proposed.

Chapter 3

Genetic Algorithm For Multicast Routing

Genetic Algorithms (GAs)[86] are computational models inspired by natural evolution. They have been used mainly as function optimizers[83,84]. NP hard combinatorial optimization problems have also been effectively treated [82],[85]. Genetic algorithm's strength is essentially due to the updating of the whole population of possible solutions in an adaptive way guided by operators such as selection, crossover, and mutation [82-86].
There are four basic components in GA:

1. representation of individuals,
2. determination of the fitness function,
3. design of genetic operators, and
4. determination of the probabilities controlling the genetic operators.

In this chapter, these issues are addressed within the framework of the genetic algorithm newly proposed for solving multicast routing problem. Our new algorithm is described in section 3.2. Comparison of the performance of the proposed algorithm with other known algorithms is given in chapter 4.

3.1 Overview of Genetic Algorithms

Holland [85] described a methodology for studying natural adaptive systems and designing artificial adaptive systems. It is now frequently used as an optimization method. The biological basis for this adaptation process is Darwinian natural selection and Mendelian genetics, that is elimination of weak elements by favoring retention of optimal and near-optimal individuals (survival of the fittest) and recombination of features of good individuals to perhaps make better individuals.

Multicast routing belongs to a class of problem known as NP-hard problems. The computation of a truly optimal solution to these class of problems is very hard and usually possible only for a very limited domain. Some heuristic methods must typically be applied to reduce the search space and generate sets of approximate (near-optimal) solutions. In the genetic algorithm approach, as shown in Fig. 3.1, each point in the search space is called a *chromosome* or string, and represents a possible solution to the problem [83].

A GA approach requires a population of chromosomes (strings) representing a combination of features from the set of features, and requires a cost function (called an evaluation or fitness function) $F(n)$. This function calculates the fitness of each chromosome. The algorithm manipulates a finite set (population) of chromosomes, based loosely on the mechanism of natural evolution. In each generation, chromosomes are subjected to certain operators, such as crossover, inversion, and mutation, analogous to processes which occur in natural reproduction. The *crossover* of two chromosomes produces a pair of offspring chromosomes which are syntheses or combinations of the traits of their parents. Inversion in a chromosome produces a mirror-image reflection of a contiguous subset of the features of the chromosome. A *mutation* on a chromosome produces a nearly identical chromosome with only local alterations of some regions of the chromosome. A great deal of information regarding these operators is available in the literature and will not be presented in detail here.

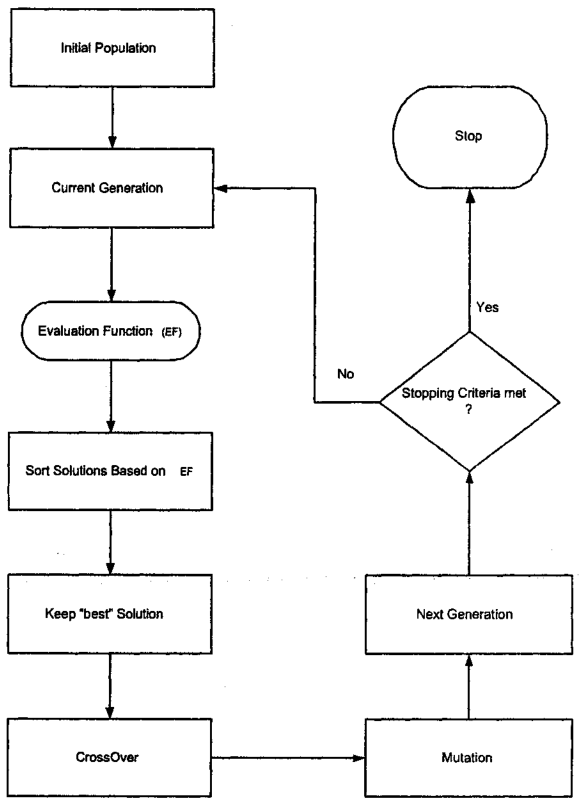

Figure 3.1: Simple genetic algorithm flowchart

3.1.1 Operation of the GA

The optimization process is performed in cycles called generations [90]. Figure 3.1 shows a flowchart of a typical GA algorithm. During each generation, a set of new chromosomes is created using the crossover, inversion, and mutation operators. Since the population size is finite, only the best chromosomes are allowed to survive to the next cycle of reproduction. The crossover rate often assumes quite high values (of the order of $80 - 85\%$), while the mutation rate is small (typically $1 - 15\%$) for efficient search. The cycle is repeated until the population converges; that is, the diversity of the feature values among the population is very low and further exploration seems pointless, or until the answer is good enough.

3.2 The Proposed GA for Constrained Multicast Routing

3.2.1 Problem Formulation and Network Model

The problem considered here is how communication paths are generated through a packet-switched network for multicast traffic. A network is presented as a directed graph $G = (V, E)$ consisting of a set of switches, V, and a set of directed links, E. Let the link from node i to node j be denoted by $e(i, j)$. Each link $e \in E$ is associated with a cost $C(e)$ and several QoS parameters, such as a delay, loss probability, and jitter. The cost function, $C(e)$ is a positive real function, i.e., $C : E \rightarrow R^+$. The cost function reflects the amount of resources required to support the quality of service provided by the link. The QoS supported on a link is described by QoS functions. Each QoS function, $Q_i(e)$, is a positive real function which gives the quality of the parameter that can be guaranteed on the link e.

For a multicast connection, packets originating at the source node $s \in V$, have to be delivered to a set of destination nodes $M \subseteq V - s$. We refer to M as the *destination group*, and $s \bigcup M$ the multicast group. Multicast packets are routed from the source to the destinations via the links of a multicast tree $T = (V_T, E_T)$. A multicast tree is a subgraph of G spanning S and the nodes in M. In addition, V may contain relay nodes, that is, nodes in the multicast tree but not members of the multicast group.

3.3 QoS Metrics

The QoS guarantee for a multicast connection is defined as follows. Let $q_1, .., q_n$ be the n QoS functions and $Q_1, ., Q_n$ be the corresponding QoS constraints that need to be satisfied. A multicast tree is said to be able to provide the required QoS guarantee if the end to end QoS of each source-destination pair of the multicast connection is satisfied.

In this thesis, we only consider QoS parameters that are additive, i.e., the end to end QoS of a path is the sum of individual QoS of each link on the path. QoS parameters such as a delay and jitter are additive in nature. The end to end loss probability of a path can be approximated by the sum of loss probabilities of all links of the path if the link loss probability is very small. Formally, for each $v \in M$, the end to end QoS is guaranteed by

$$\sum_{e \in P(s,v)} q_i(e) \leq Q_i, \forall v \in M, i = 1, 2, ..., n \qquad (3.1)$$

Where $P(s, v)$ is the path in T from s to v.

The multiple-constraint multicast routing problem is defined as follows

$$\min \sum_{e \in T} C(e) S.t \sum_{e \in P(s,v)} q_i(e) \leq Q_i, \forall v \in M, i = 1, 2, ..., n \qquad (3.2)$$

Two important QoS metrics are considered in this dissertation :

- ***Source-destination delay***: Δ: The parameter Δ represents an upper bound on the acceptable end-to-end delay along any path from the source to the destination nodes. This parameter reflects the fact that the information carried by a multicast packets becomes stale Δ time units after its transmission at the source. This parameter reflects the fact that a packet delivered Δ time units after its transmission at the source is of no value to the receivers.

- ***Interdestination delay variation*** : δ : is the parameter that represents the maximum difference between end-to-end delays along the paths from source to any two destination nodes that can be tolerated by the application. In essence, this parameter defines a synchronization window for the various receivers. By supplying values for parameters Δ, δ, the application in effect imposes a set of constraints on the paths of the multicast tree.

Given the delay, Δ and delay variation, δ tolerances, our objective then is to determine a multicast tree such that the delays along all source-destination paths are within the two tolerances. Or mathematically can be stated as:

Given a network $G = (v, A)$, a source node $s \in V$, a multicast group $M \subseteq V - s$, a link-delay function $D : A \longrightarrow R^+$, a delay Δ and delay variations, δ, is there a tree $T = (v_t, A_T)$ spanning s nodes in M, such that

$$\sum_{l \in P_T(s,v)} D(l) \leq \Delta, \forall v \in M \tag{3.3}$$

$$\left| \sum_{l \in P_T(s,v)} D(l) - \sum_{l \in P_T(s,u)} D(l) \right| \leq \delta, \quad \forall v \in M \forall (v,u) \in M \tag{3.4}$$

where Eq. 3.3 is the source-destination constraint, and Eq. 3.4 is the interdestination delay constraint. A tree T is feasible if and only if T satisfies both Eq. 3.3 and Eq. 3.4.

3.4 Algorithm description

3.4.1 Construction of Routing table

In the network graph , $G = (V, E)$, there are $|V|(|V| - 1)$ possible source-destination pairs. There are usually many possible routes between any source-destination pair. Our GADVM algorithm assumes that a routing table, consisting of R possible routes, has been constructed for each source-destination pair using the k-shortest path algorithm proposed in [82]. The size of the routing table, R, is the parameter of our algorithm.

3.4.2 Generating The Initial Population

for a given source node s and a destination set $M = \{m_1, m_2, ...m_k\}$, a *chromosome* can be represented by a string of integers with length k. A *gene*, $g_i, 1 \leq I, ... \leq k$, of the chromosome is an integer in $\{1, 2, , r\}$ which represents a possible route between s and m_i, where $m_i \in M$ [87]. Obviously, a chromosome represents a candidate solution for the multicast routing problem since it guarantees a path between the source node to any of the destination nodes. However a chromosome does not necessarily represent a tree. Therefore, we trim the extra edges using a minimum directed spanning tree algorithm, modified from the optimum branching algorithm proposed in [84].

3.4.3 Generating the new population

Select two chromosomes from the population of the current generation of which is the best chromosome (parent 1) and the other a randomly selected chromosome (parent 2). Make a crossover operation between the two chromosomes producing two new genomes (child 1 and child 2). Make a mutation operation to both of the children (Figure 3.2). Perform a selection to obtain the new generation.

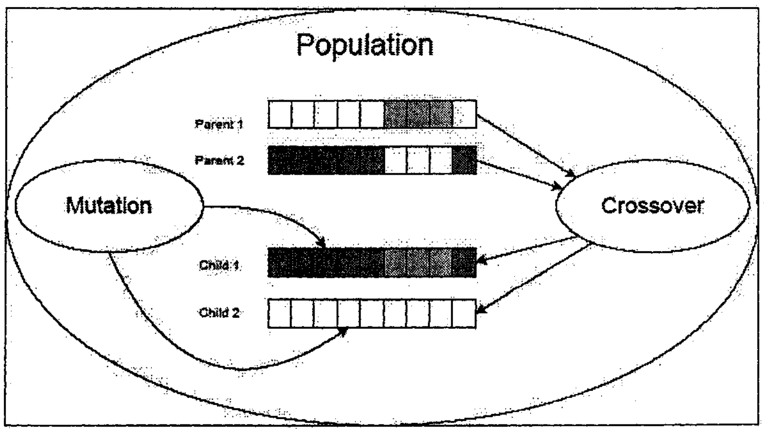

Figure 3.2: Generating a new population

3.4.4 Crossover Operation

Crossover operation generates two children from two parents. The children inherit genes randomly from the parents. Whether the crossover is made at all is determined by the parameter p_c (usually in the interval [0.5, 1.0]. If $p_c = 1.0$ crossover is always made. If $p_c \leq 1.0$ crossover is made with probability p_c. If the crossover operation is not made to the parents the genes are copied to the children unchanged. If crossover is made to the parents then in this algorithm the (method of two points) is used in which two randomly selected genes the starting point (6 in Figure 3.2) and the ending point (8 in Figure 3.2) are determined for parent 2. Then, the genes between these points are exchanged between the parents. In Figure 3.3 we present a more detailed example of crossover. In this example only one gene is changed between the parents (the starting and ending points are the same = 2, the corresponding values of the changing genes are 7 and 4).

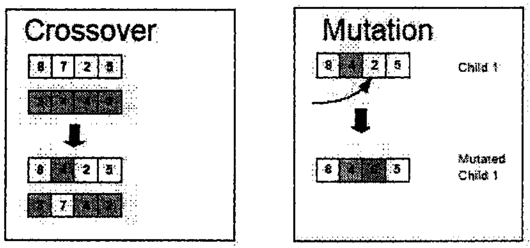

Figure 3.3: Crossover and mutation operations

3.4.5 Mutation Operation

When the children chromosomes have been created both of them undergo mutation operations. The number of the operations n_{op} is determined by the parameter p_m (usually in the interval [0.01, 0.2]) and is calculated as follows

$$n_{op} = (p)(p_m) \tag{3.5}$$

where n_{op} is the number of mutations, p is the length of chromosome, and p_m is a user-defined parameter, in the interval[0.01,0.2].

The mutation operations are performed by selecting n_{op} random genes of the chromosome and replacing the value of the selected gene by a random i integer from the feasible interval of the corresponding integer variable (Figure 3.3).

3.4.6 Steps of the Algorithm

The GADVM algorithm is outlined in the following steps [88]

- **Step 1**: Initialize a population of chromosomes. The GADVM algorithm first generates P different chromosomes at random which form the first generation. The set of chromosomes is called the chromosomes pool (or population), and P is the size of the gene pool.

- **Step 2**: Evaluate each chromosome in the chromosome pool. The fitness value of a chromosome is the value of the objective (fitness) function for the solution (e.g., a multicast tree) represented by the chromosome. Given a chromosome pool $H = \{h_1, h_2, .., h_p\}$, the fitness value of each chromosome is computed as follows. Let $C(h_i)$ be the overall link cost of the graph represented by the chromosome h_i. Let $C(E)$ be the sum of the costs of all links of the network. The fitness value of the chromosome $h_i, F(h_i)$, is given by

$$f_{h_i} = 1 - \frac{C(h_i)}{C(E)} \quad if \sum_{e \in P(s,v)} q_i(e) \leq Q_i \forall v \in M, i = 1,2,3,...n \quad (3.6)$$

$$= 0 \qquad Otherwise \quad (3.7)$$

  where $P(s,v)$ is the path from source s to destination v, derived from chromosome h_i. After evaluating the fitness values of all chromosomes, chromosomes are then sorted according to their fitness values such that $F(h_1) \geq F(h_2) \geq ... \geq F(h_p)$. That is the first chromosome in the pool is the best solution found so far.

- **Step 3**: If the number of generations is larger than the pre-defined maximum number of attrations, MaxGen, then stop and output the best chromosome (solution), otherwise, go to step 4.
- **Step 4**: Discard duplicated chromosomes. There might be duplicated chromosomes in the pool. Apply some of the genetic operations, e.g. crossover, on two duplicate chromosomes will yield the same offspring. Therefore, too many redundant chromosomes will reduce the ability of searching. Once this situation occurs, the redundant chromosomes must be discarded. New randomly generated chromosomes replace them.
- **Step 5**: Generate next generation of chromosomes by applying generic operations: reproduction, crossover, and mutation.
- **Step 6**: Stop when the number of generations reaches the maximum number of generation, MaxGen, or when no further improvement is observed on the fitness function.

<u>The Pseudo code for the GADVM algorithm:</u>

Given the input parameters as follows:
G=(v,E)
s= source node.
M= destination nodes.
$Msize$ = size of the destination set.
Q_i = QoS constraints (Δ, δ).
n = number of QoS parameters.
P = Size of the chromosome pool.
$h_1, h_2, ...h_p$= chromosome pool of previous generation.
$h'_1, h'_2, ...h'_p$= chromosome of the new generation.
$MuRate$ = mutation rate.
$F(h_i)$ = Fitness of the chromosome h_i

```
$MaxGen$= Maximum number of generations
$G_i$= current number of generations.
(initialization of the chromosome)
for $i = 1$ to $P$ Do
begin
for each $m_i \in M$ do
begin
randomly choose a route from s to $m_i$ to the jth gene of the chromosome$h_i$.
end $\Rightarrow$ for/
end $\Rightarrow$ for/
$G_i = 1$
start the GADVM generations/
begin
/ Evaluation of the chromosome/
for $i = 1toP$ do
begin
if $\sum_{e \in P(s,v)} Q_j(e) \leq \Delta, \quad \vee \in\in M, j = 1, 2, ....., n$
$F(h_i) = 1 - \frac{C(h_i)}{C(L)}$
else
$F(h_i) = 0$
end $\Rightarrow$ for/
Sort the chromosomes according to their fitness values such that
$F(h_1) \geq F(h_2) \geq ...... \geq F(h_p)$
/Stopping Criteria
if($G_i$¿MaxGen) then
begin
$T = decode(h_1)$ where $T$ is the multicast tree constructed        $h_1$
end
else
$G_i = G_i + 1$
Discard the duplicated chromosomes
for $i = 1$ to $P$ do
begin
for $j = j + 1$ to $P$ do
begin
```

```
if ( chromosome $h_i$ = chromosome $h_j$)
chromosome $h_j$ =ramdomly generate a new chromosome, $h_i \neq h_j$
end
end Begin genetic operations
*Reproduction*
generate a random integer $p$, $1 \leq P \leq P$
The new chromosomes $h'_1, h'_2, ....h'_p$ are reproduced from $h_1, h_2, ...h_p$.
*Crossover*
$i = p$
while ($i \leq P$ ) DO
begin
Randomly choose two chromosomes, $h_x$,$h_y$, from population pool P
Randomly choose two values $q, r$ such that ( $1 \leq q \leq r \leq Msize$)
for $j = 1$ to $M_size$ do
begin if ($j \leq q$) or( $j \geq$) then
begin
the $jth$ gene of the new chromosome $h'_{i+1}$ = the $jth$ gene of the chromosome $h_x$
the $jth$ gene of the new chromosome $h'_{i+2}$ = the $jth$ gene of the chromosome $h_y$
end $\Rightarrow$ if
else
the $jth$ gene of the new chromosome $h'_{i+1}$ = the $jth$ gene of the chromosome $h_y$
the $jth$ gene of the new chromosome $h'_{i+2}$ = the $jth$ gene of the chromosome $h_x$
begin
end $\Rightarrow$ else
end $\Rightarrow$ for
$i = i + 2$ discard chromosome $h'_{i+2}$
end $\Rightarrow$ while
*Mutation*
for $i = 1$ to $P$ do
begin
generate random number $r$, $(0 \leq r \leq 1)$
if ($r \leq MuRate$) then
begin
generate a random number $p$, $(1 \leq p \leq Msize)$
the $pth$ gene of new chromosome $h`_i$ = randomly choose a new route
```

```
end
end ⇒ for
replace $h_1, h_2, ...., h_p$ with $h'_1, h'_2, ...h'_p$
end
Output $T$ as a multicast tree end
```

3.5 GADVM Parameters Setup

Parameters of the GADVM algorithm, such as the size of the chromosome pool, mutation rate, and the number of generations, must be properly selected to yield the best performance. In this section we present simulation results from different values of the above parameters and we use the best values obtained later in chapter 4 when comparing the performance evaluation of the proposed GADVM algorithm.

- *Choosing the chromosome's pool size*: We first examine the effect of the size of the chromosome pool on the performance of the GADVM algorithm. Figure 3.4 shows the minimum cost obtained after 200 generations with different chromosome pool sizes under 60-node random graphs where the size of the destination group is set to be 20% of the graph nodes. The mutation rate is set to be 0.4. From Fig. 3.4, we can observe that after 200 generations, the chromosome pool size of 90 yields the best performance.

- *Choosing proper mutation rate*: Figure 3.5 shows the effect of mutation rate on the performance of GADVM. It shows the cost under various mutation rates when the chromosome pool size is set to 90. We can observe that GADVM yields the best performance when mutation rate is set to 0.4.

- *Choosing the number of generations*: Figure 3.6 shows the effect of the number of generations and the size of the routing table on the performance of the algorithm. The chromosome pool size is set to 90 and the mutation rate is set to 0.4. The three curves in Fig. 3.6 show that minimum cost obtained at each generation when the sizes of the routing table are set to 60, 80, 100 respectively.

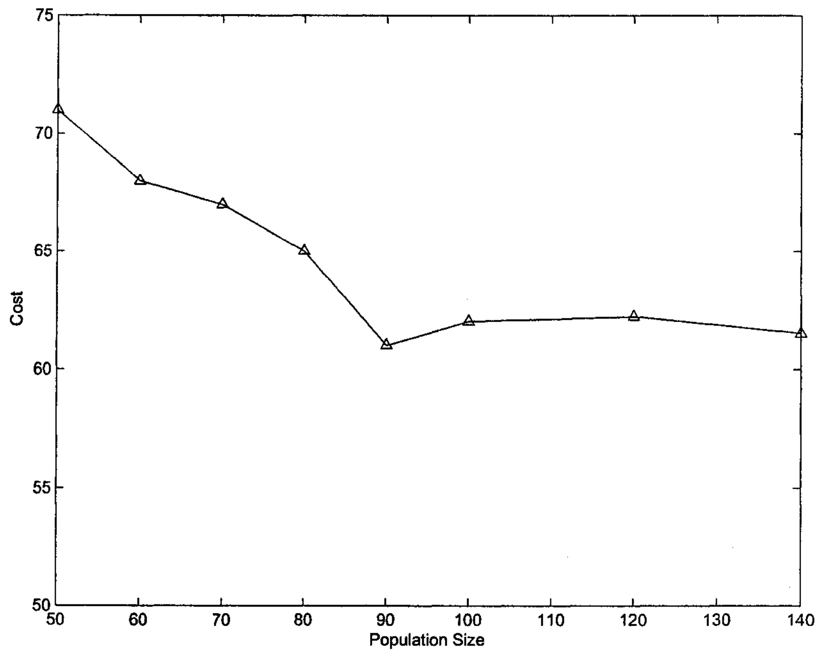

Figure 3.4: Crossover and Mutation operation

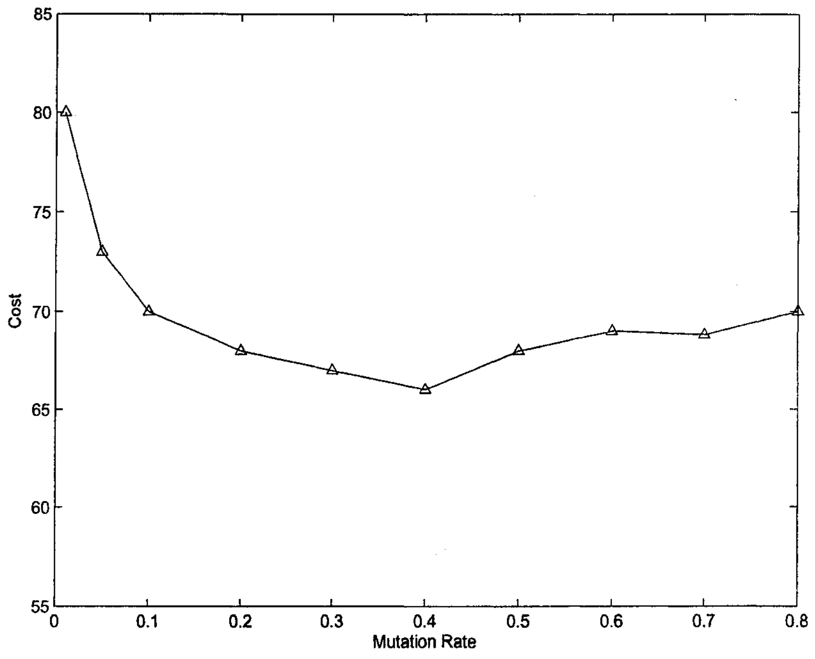

Figure 3.5: Crossover and Mutation operation

One might expect that a larger number of routes in routing table lead to better performance, because more diverse routes available in a large routing table can prevent the generic algorithm from getting trapped in the local optimum. However, we have observed that the smaller the routing table the better the performance. This may be due to the slow convergent rate of larger routing table size. That is, the process has not converged to the optimal solution yet. In our later simulations, we set the size of the routing table to 40. The number of generations is set to 250 since the cost does not decrease much after this number.

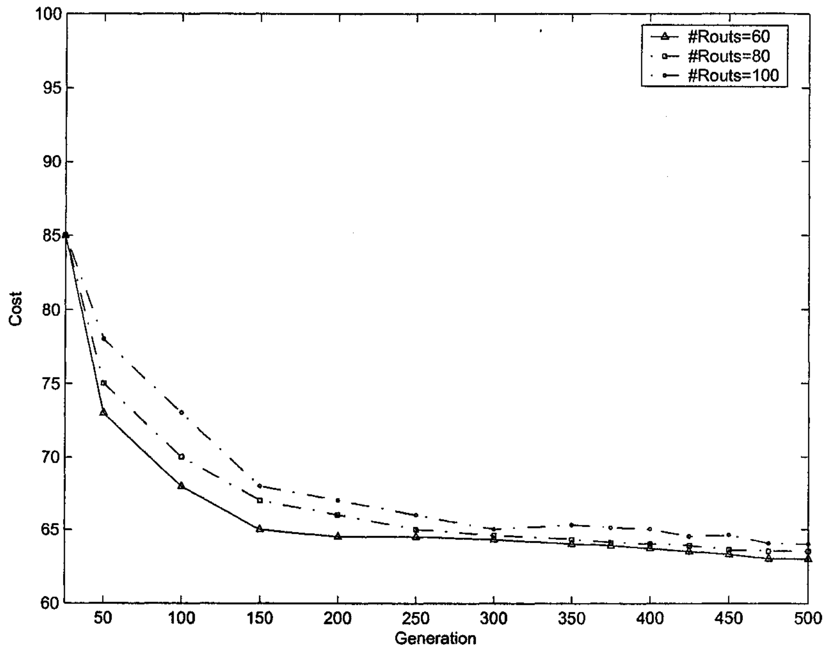

Figure 3.6: Crossover and Mutation operation

Chapter 4

Performance Evaluation of the Proposed GADVM

In this chapter, we report the results of comprehensive simulations to evaluate the proposed algorithm and compare its performance with other algorithms. We use the random network generator described in section 4.1. The simulated scenarios of algorithms are presented in section 4.2. In section 4.3, we present the performance metrics that are used to evaluate the different algorithms. The effect of relaxing the delay variation constraint is investigated in section 4.4. In section 4.5, we examine the effect of changing the multicast group size on the performance of different algorithms. The effect of increasing the number of nodes in the network is investigated in section 4.6. Section 4.7 describes a technique to improve the speed of the algorithm. Finally, the chapter is concluded in section 4.8.

4.1 Random network generator

To guarantee fair simulation results, we use the same graph generator [32] that is used in all problems related to multicasting. N nodes are randomly distributed over a rectangular area with size 2000 by 2000 where each node is placed at a location with integer coordinates. The probability of edge existence between any two nodes u and

v can be calculated from the function:

$$P(u, v) = \beta \exp(\frac{-d(u, v)}{L\alpha}) \tag{4.1}$$

where $d(u, v)$ is the distance between nodes u and v. L is the maximum distance between any two nodes, α and β are two parameters used to adjust the degree of the graph and the density of short and long edges. If $P(u, v)$ is greater than 0.5, then an edge exists between nodes u and v, otherwise, there is no edge between these two nodes. After calculating the preceding function for each pair, the resulting graph does not necessarily need to be connected, so, we add edges at random till we get a connected graph. The cost of any edge $e(u, v)$ is equal to the distance $d(u, v)$ and the delay of any edge is a random value according to uniform distribution between 1 and 10. Finally, for each algorithm to be correctly evaluated, we run it on 3000 different graphs with the same values of n, α, β and taking their average.

4.2 Simulated Algorithms

The random graph generator described in section 4.1 is used to simulate and compare the following:

- The shortest path delay tree (SPT) obtained by applying Dijkstra algorithm [3].
- CDKS algorithm [8], designed for the delay constrained shortest path tree. It first constructs the shortest path cost tree. Then, it looks for every destination and tests it according to the delay constraint Δ. If the destination satisfies Δ, then the shortest path cost for this destination will be in the resulting tree, otherwise, it replaces the path for this destination from the shortest path delay tree.
- The BSMA algorithm [41] that is described in section 2.4 for the delay constrained shortest path problem.

- The KPP [37,38], is a heuristic algorithm proposed for the delay constrained minimum Steiner tree problem. KPP is described in section 2.4.4.
- The proposed GADVM algorithm that is designed for the delay and delay variation constrained shortest path problem.

4.3 Performance Metrics

For all the simulated tests of the algorithms, we consider two performance measures, the failure rate and the average cost per path of the resulting tree.

- **Failure Rate**: Any algorithm can fail to find a suitable tree by not satisfying either the delay constraint or the delay variation constraint. From the definition of SPT and CDKS algorithms, all algorithms will find a delay constrained tree if one exists. So, all simulated algorithms have the same failure rate to find a delay constrained tree. Hence, we will not use the failure in the delay constraint as a performance measure. On the other hand, the failure of finding a tree that satisfies the delay variation constraint is different for all algorithms and gives an indication of how suitable these algorithms are in meeting the delay variation constraint. So, we define the failure rate as the rate that at which the algorithm failed to find a tree that satisfies the delay variation constraint.

- **Average Cost per Path** The second performance measure we consider is the average cost per path of the resulting tree. For the algorithms SPT, CDKS and BSMA that are designed mainly for the delay constrained shortest path tree problem, we calculate the cost whether the algorithm failed to satisfy the delay variation constraint or not. For the GADVM algorithm, when the failure happened, we calculate the cost of the tree with minimum possible delay variation.

4.4 Performance with respect to the Delay Variation

4.4.1 Failure Rate

Figures 4.1, 4.2, and 4.3 display the effect of changing the delay variation constraint on the failure rate. For the three figures, we set the number of nodes in the network N = 50 and the delay constraint $\Delta = 0.05$, while changing the delay variation constraint δ from 0 to 0.05. Setting the delay variation constraint to 0 means that all the destinations should get the data at the same time while setting $\delta = 0.05$, the same value as Δ, cancels the effect of the delay variation constraint. In Fig. 4.1, we set the average node degree d = 4 and the multicast group size M = 5. With a tight delay variation constraint, the performance of SPT, CDKS and BSMA is very poor with BSMA being the worst. The SPT algorithm is always better than CDKS and BSMA. This can be deduced from the fact that SPT tends to minimize the delay and hence all the destinations have the lowest possible delay. The failure rate of all algorithms is zero at $\delta = \Delta = 0.05$. This is because the problem is reduced to be a shortest path problem under delay constraint. Fig. 4.2 differs from Fig. 4.1 in the size of multicast group M which is set to 25 instead of 5. The increase in multicast group size raises the difficulty of satisfying the delay variation constraint. CDKS and BSMA algorithms have a very poor response for this increase where they give 100% failure at $\delta \leq 0.02$, also SPT gives 100% failure at $\delta \leq 0.015$. GADVM still dominates the other algorithms.

Fig. 4.3 differs from Fig. 4.1 in the average node degree d where we set it to 10 instead of 4. Increasing the average degree increases the number of available paths, which results in decreasing the failure rate. GADVM almost has zero failure rate for all delay variations even when $\delta = 0$. KPP gives zero failure rate $\delta \geq 0.005$. The performance of SPT, CDKS, and BSMA is enhanced but still gives almost 100% failure for small values of δ

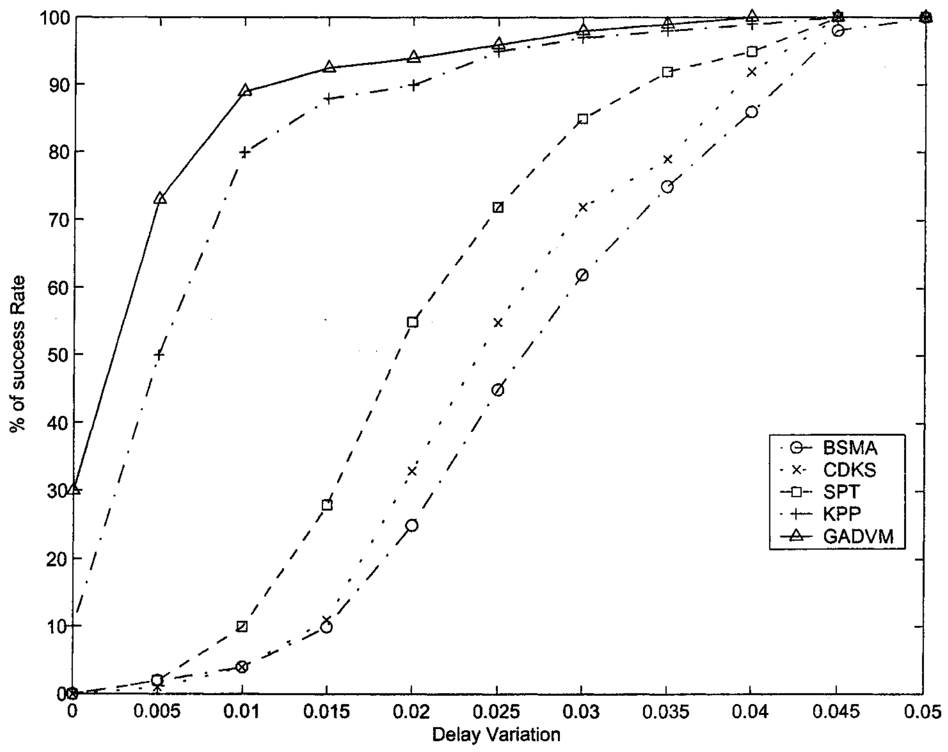

Figure 4.1: The effect of delay variation constraints on failure rate with network size=50, Multicast group=5, Average node degree = 4

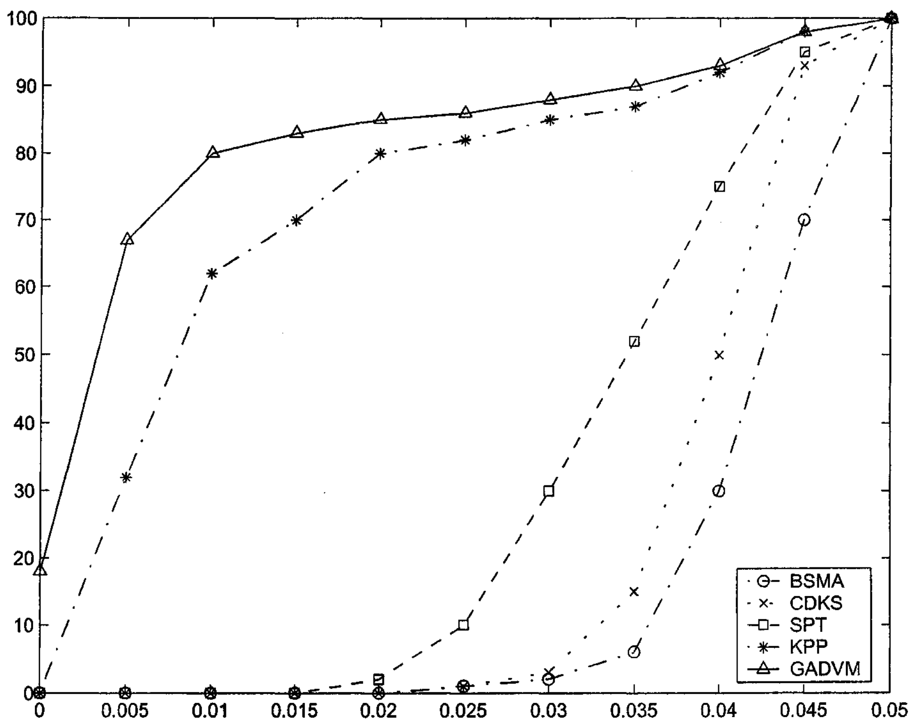

Figure 4.2: The effect of delay variation constraints on failure rate with network size=50, Multicast group=25, Average node degree = 4

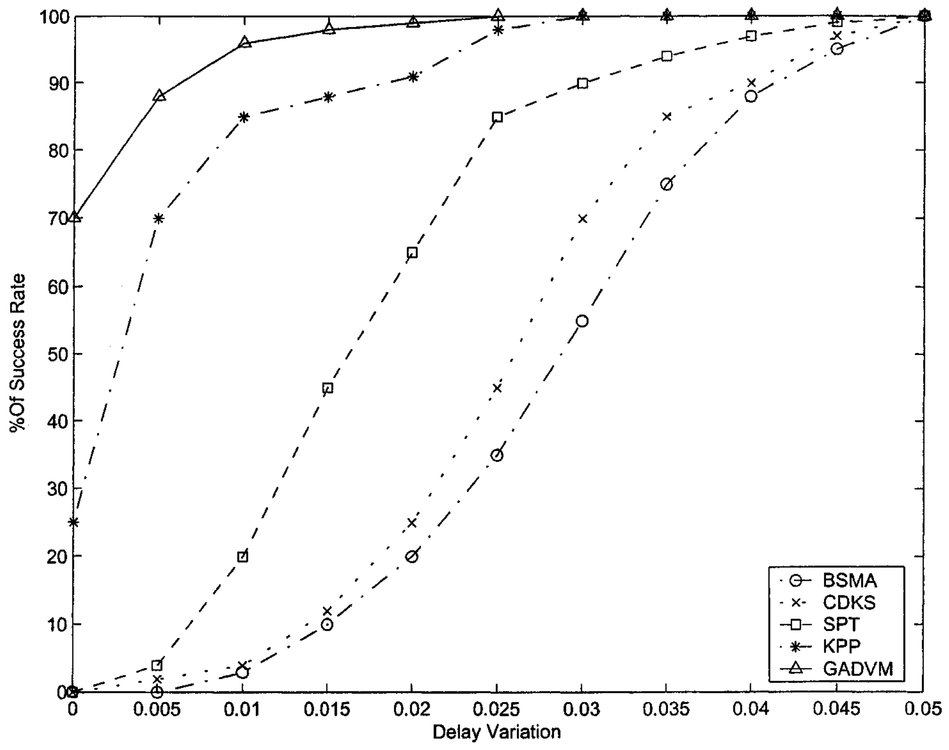

Figure 4.3: The effect of delay variation constraints on failure rate with network size=50, Multicast group=5, Average node degree = 10

4.4.2 Average Cost per Path

Figures 4.4, 4.5 and 4.6 show the effect of changing the delay variation constraint on average cost per path. For the three figures, we set the number of nodes in the network to N = 50 and the delay constraint $\Delta = 0.05$, while changing the delay variation constraint δ from 0 to 0.05.

For the three figures, it can be observed that the three lines of SPT, CDKS and BSMA algorithm are constant values. This means that the cost of the resulting tree is not dependent on the delay variation constraint. This can be deduced from the fact that the three algorithms are mainly developed for the problem of shortest path tree under delay constraint. So, they do not pay any attention to the delay variation constraint. The performance of BSMA is always better than CDKS which is better than SPT.

In Fig. 4.4, we set the average node degree d = 4 and the multicast group size M = 5. KPP gives very high cost that is not comparable to any other algorithm when $\delta \leq 0.02$, then it approaches the same performance of CDKS at $\delta \geq 0.04$. GADVM has a high cost at $\delta \leq 0.01$. This cost is considered to be paid for the added constraint. Recalling that at small δ, it was very difficult to obtain a solution for the other algorithms, so, finding a solution for small δ should be with a very high cost. The performance of GADVM is enhanced by relaxing the delay variation constraints and it dominates CDKS at $\delta \geq 0.025$, then it approaches BSMA but it did not overcome it. GADVM never dominates BSMA because GADVM uses an additional constraint which results in an additional cost. Comparing GADVM with CDKS when $\delta \geq 0.025$ yielding that, although GADVM has an additional constraint, it also gives better cost.

In Fig. 4.5, we increase the multicast group size M to 25, this gives almost the same graph of Fig. 4.4 with the same analysis. This indicates that changing the multicast group does not really have a significant effect on the average cost per path.

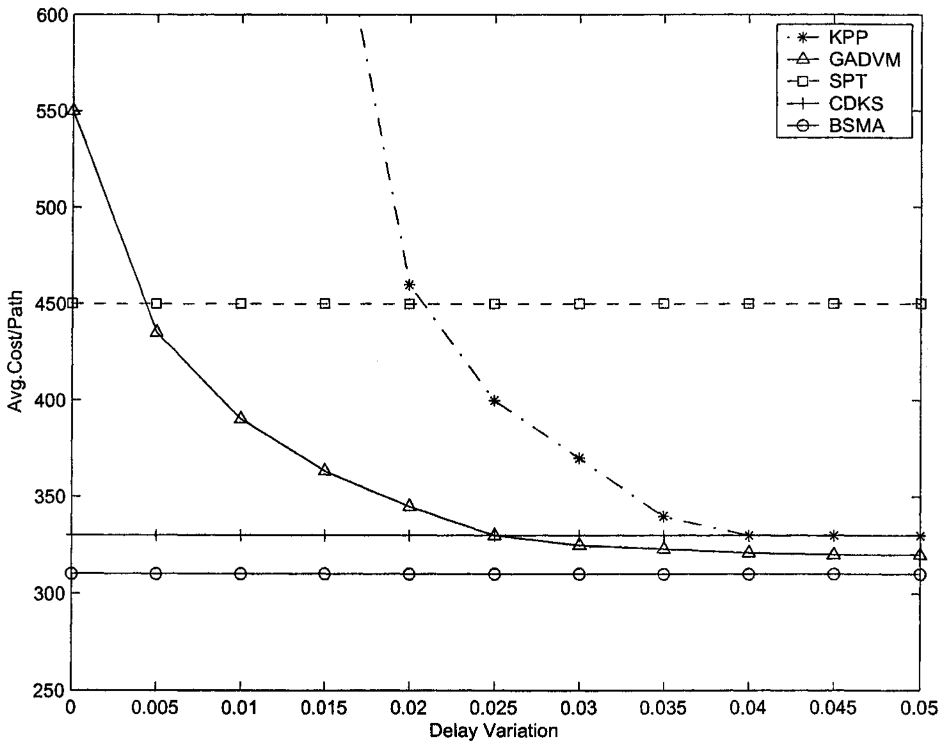

Figure 4.4: The effect of delay variation constraint on average cost/path rate with network size=50, delay constraint= 0.05, Multicast group=5, Average node degree = 4

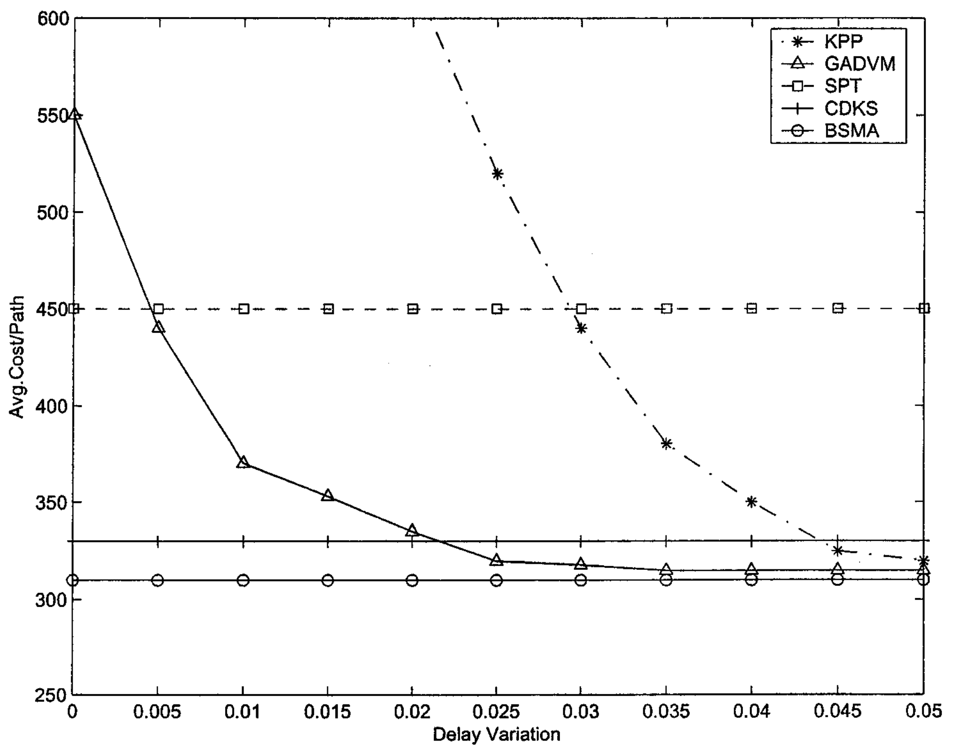

Figure 4.5: The effect of delay variation constraint on average cost/path rate with network size=50, delay constraint= 0.05, Multicast group=5, Average node degree = 4

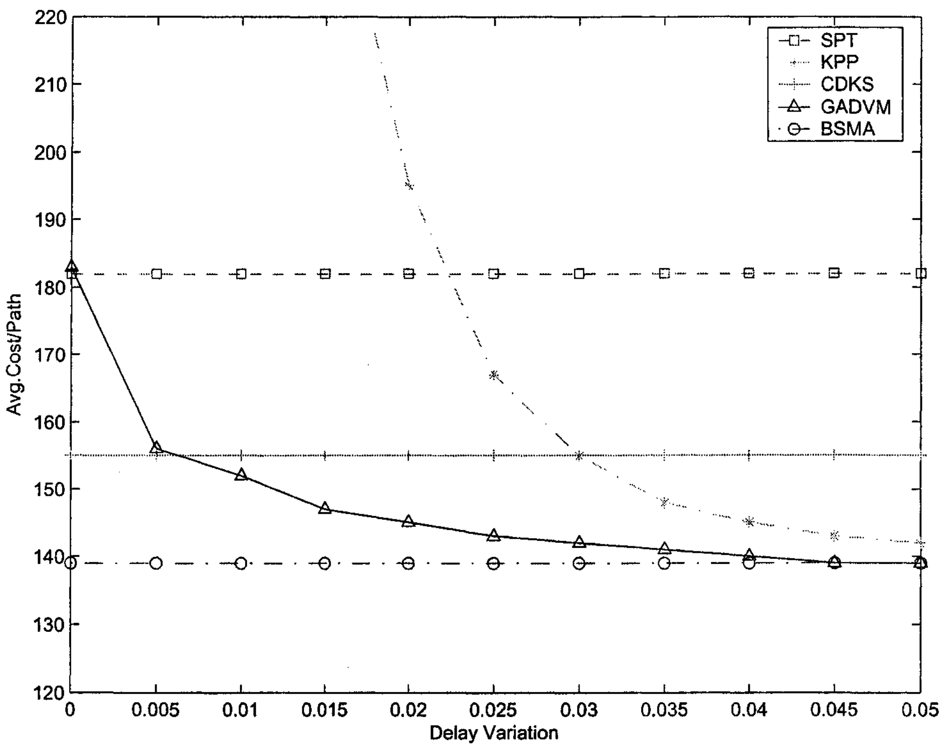

Figure 4.6: The effect of delay variation constraint on average cost/path rate with network size=50, delay constraint= 0.05, Multicast group=5, Average node degree = 4

This is because all algorithms compared deal with multicast as a special case of broadcasting, so, it does not matter how large is the the multicast group is. Fig. 4.6 differs from Fig. 4.4 in the average node degree d which is set to 10 instead of 4.

The performance of GADVM is enhanced with the increase of average node degree. The difference between GADVM and BSMA is decreased. Also, KPP dominates CDKS when $\delta \geq 0.03$. For all algorithms, the average cost per path is decreased to half of its value in Fig. 4.4. This is because increasing the average degree results in more available paths, which results in more available cheaper paths.

4.5 The Effect of Changing Multicast Group Size on GADVM Algorithm

4.5.1 Failure Rate

The effect of changing the multicast group size on the failure rate is investigated in Figures 4.7, 4.8 and 4.9. For the three figures, we set the network size N = 50, delay constraint $\Delta = 0.05$ while changing the multicast group size from 5 to 45. Increasing the size of the multicast group makes the problem of delay variation constraint more difficult, since finding suitable values for delay in 45 nodes is more difficult than finding it for 5 nodes.
In Fig.4.7, we set the average node degree d = 4 and we force a tight delay variation constraint $\delta = 0.05$. It can be observed that for tight delay variation, SPT, CDKS and BSMA algorithms do not work at all since they give 100% failure rate at $M > 10$. This indicates that these algorithms are not suitable for the delay variation constrained problem. On the other hand, GADVM gives relatively good results and its failure rate increases slightly with the increase of the size of multicast group.

This indicates that GADVM is working well even with tight delay constraint and in large multicast group size. KPP has a higher failure rate GADVM by almost a constant difference 15% in the different sizes of multicast group. In Fig. 4.8, we still keep the tight delay variation constraint while increasing the average node degree d to 10 instead of 4. This is to increase the number of available paths and to decrease the failure rate of all algorithms. However, CDKS and BSMA are still affected by the tight delay variation, so, they have no performance enhancement for the increase

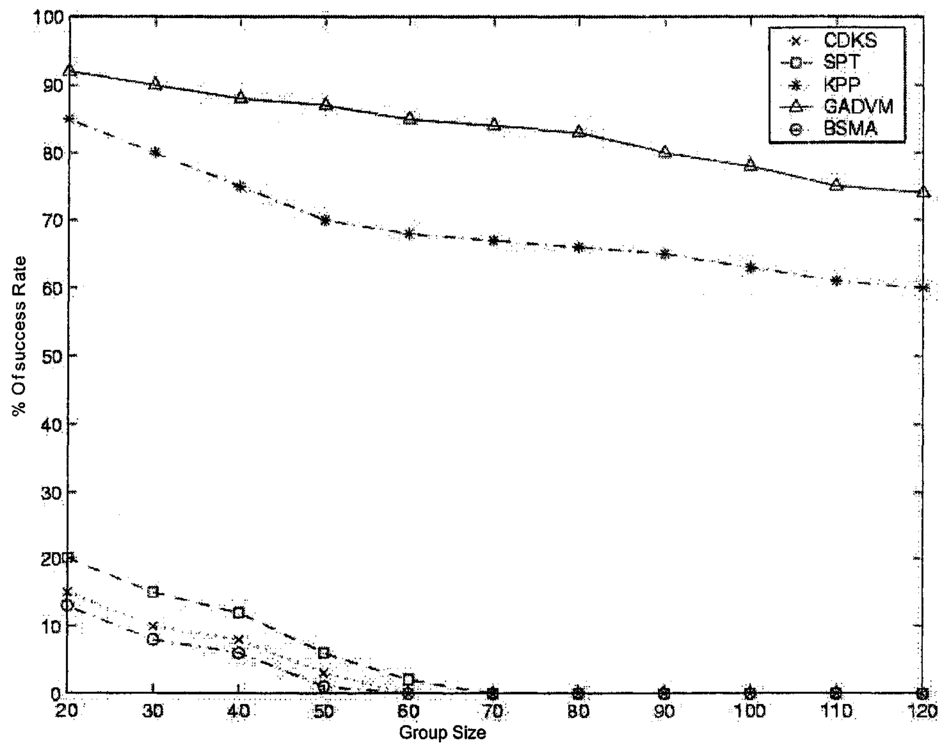

Figure 4.7: The effect of multicast group constraint on failure rate with network size=50, delay variation constraint= 0.05, delay constraint=0.2, Average node degree = 4

of the average node degree where they have 100% failure rate when M $\geq$ 10. Also, SPT has a failure rate of 100% when M$\geq$ 15. On the other hand, GADVM performs well with the increase of the average node degree even with the tight delay variation.

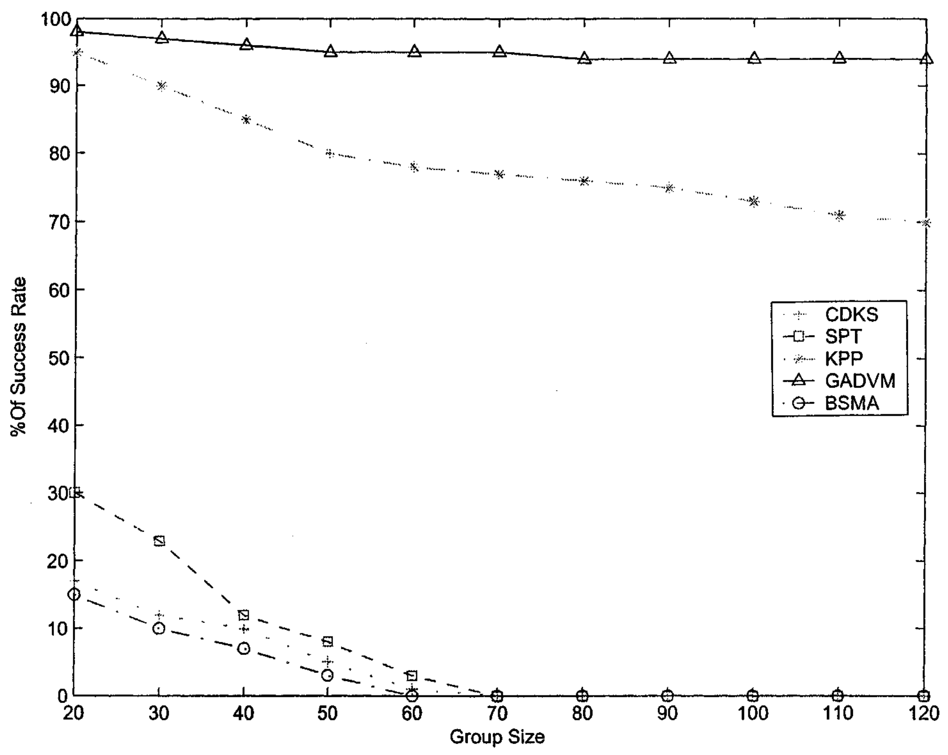

Figure 4.8: The effect of multicast group constraint on failure rate with network size=50, delay variation constraint= 0.05, delay constraint=0.2, Average node degree = 10

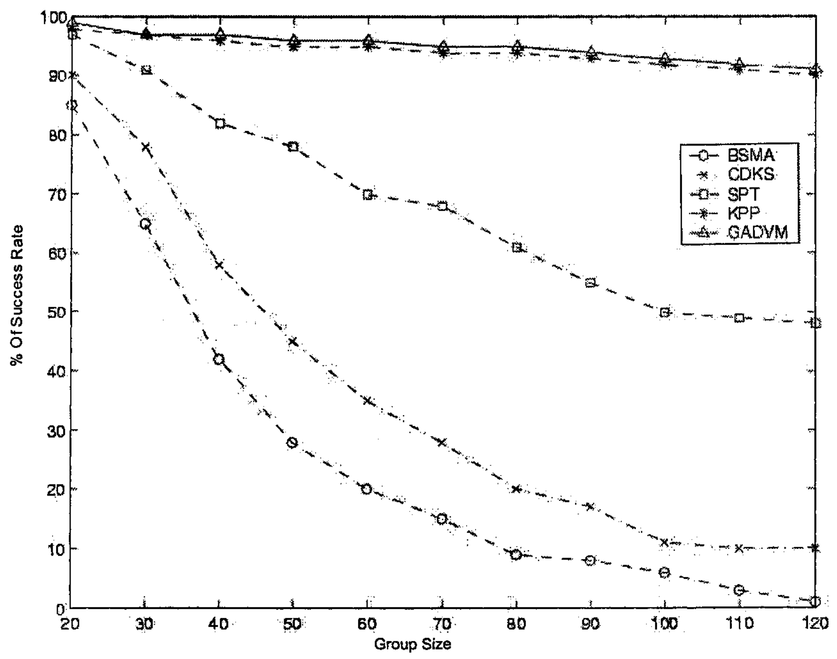

Figure 4.9: The effect of multicast group constraint on failure rate with network size=50, delay variation constraint= 0.15, delay constraint=0.2, Average node degree = 4

GADVM has almost zero failure for all different multicast group sizes. This indicates that GADVM works better for networks with high average node degree.
Fig. 4.9 differs from Fig. 4.7 in that we relax the delay variation constraint δ to be 0.15 instead of 0.05 while keeping the average node degree d = 4. The effect of relaxing the delay variation constraint is significant for all algorithms. CDKS and BSMA perform well in small multicast groups but they still cannot stand out with large multicast group size. SPT always has a failure rate below 50%, recalling that it was always 100% in Fig. 4.7 yielding that SPTD has the best response for relaxing delay variation. GADVM has a good performance and always results in a failure rate that is below 5%. The difference between KPP and GADVM is also decreased. This indicates that GADVM is preferable for applications with tight delay variations.

4.5.2 Average Cost per Path

Figures 4.10, 4.11 and 4.12 demonstrate the effect of changing the multicast group size M on the average cost per path for the resulting tree. For the three figures, we set the network size N = 50, the delay constraint $\Delta = 0.20$ while changing the multicast group size from 20 to 120.
In Fig. 4.10, we set the delay variation constraint $\delta = 0.05$ and the average node degree $d = 4$. BSMA always dominates the other algorithms, this is the result of the high failure rate. GADVM yields a cost which is worse than CDKS and better than SPTD. The additional cost that GADVM suffers from is due to its low failure rate. We should pay an additional cost to satisfy an additional constraint. KPP is not practical at all since it gives relatively high cost.
In Fig. 4.11, we increase the average node degree d to 10 while keeping the tight delay variation constraint. Comparing GADVM to CDKS, it is observed that GADVM gives better cost and better failure rate, which means that in all measures GADVM dominates CDKS in the case of large average node degree. BSMA still get the best result in terms of cost. KPP still gives a very high cost which makes it not practical. The increase of average node degree results in less cost for all algorithms.

Fig. 4.12 differs from Fig. 4.10 in that we relax the delay variation limit to be 0.15 instead of 0.05 while keeping the same average node degree d. Relaxing the delay variation constraint results in making the performance of GADVM better than CDKS. KPP gives reasonable results and always dominates SPT. This indicates that the performance of GADVM algorithm gives better results in the case of relaxing delay variation constraint.

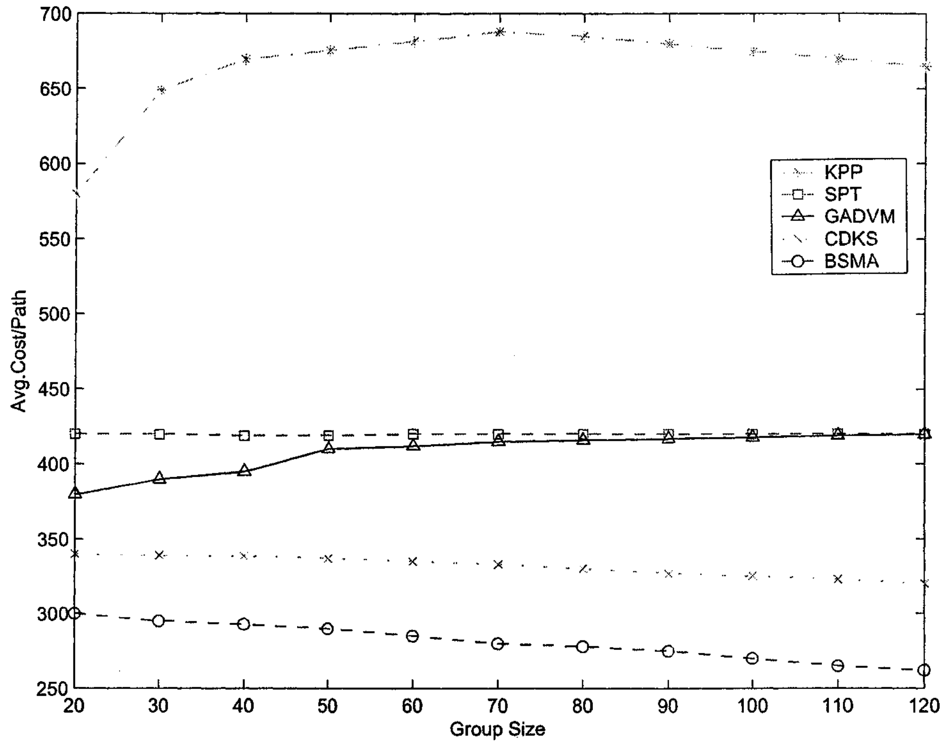

Figure 4.10: The effect of multicast group constraint on average cost/path rate with network size=50, delay variation constraint= 0.05, delay constraint=0.20, Average node degree = 4

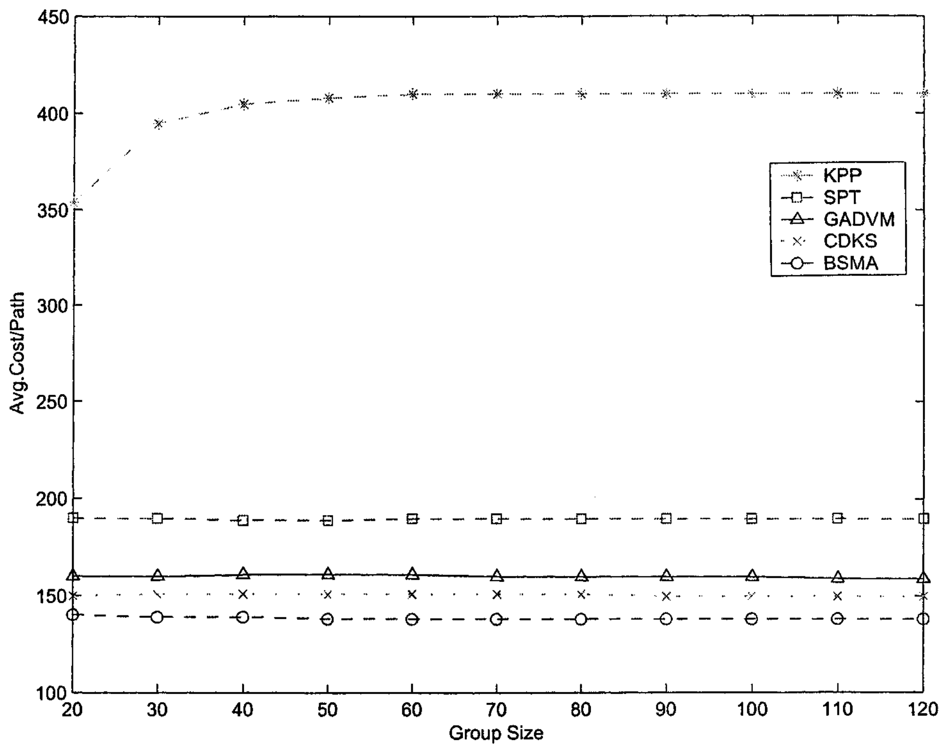

Figure 4.11: The effect of multicast group constraint on average cost/path rate with network size=50, delay variation constraint= 0.05, delay constraint=0.20, Average node degree =10

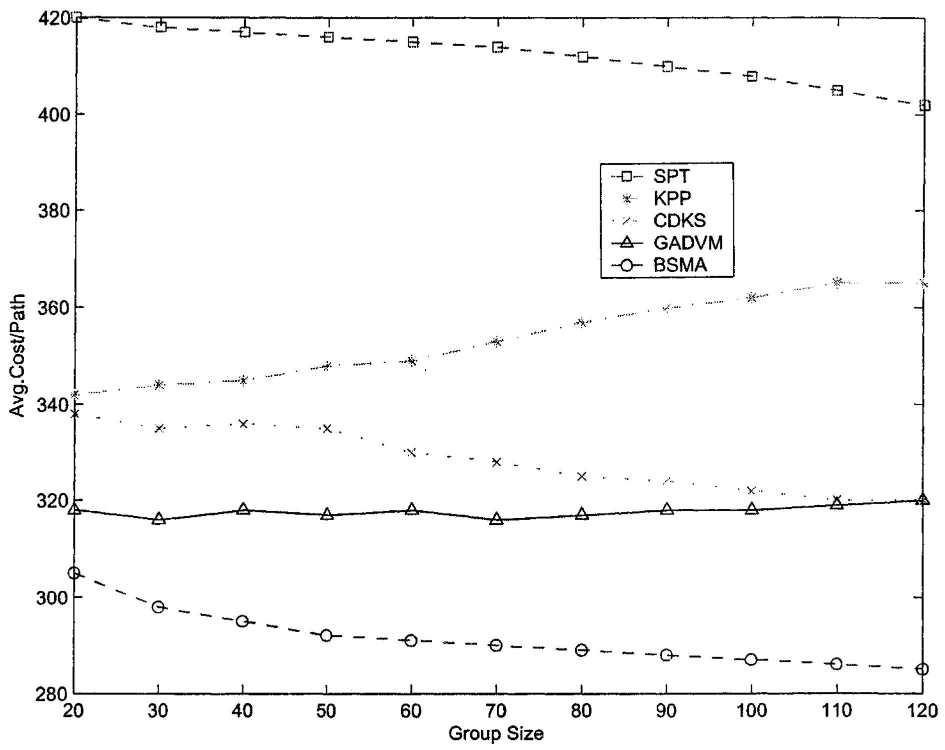

Figure 4.12: The effect of multicast group constraint on average cost/path rate with network size=50, delay variation constraint= 0.05, delay constraint=0.20, Average node degree = 4

4.6 The Effect of Changing Network Size on GADVM Algorithm

4.6.1 Failure Rate

The effect of changing network size on the failure rate is investigated in Figures 4.13 and 4.14. For both figures, we set the delay constraint $\Delta = 0.2$, the multicast group M = 5 and the average degree d = 8 while changing the network size from 20 to 90. In Fig. 4.13, we set the delay variation constraint $\delta = 0.05$. GADVM always gives zero failure rate for different network sizes. KPP gives 25% failure rate at N = 10, then the failure rate is decreased as the network size increases.
The performance of SPT is decreased with the increase in network size. This is because SPT tends to minimize the delay, so, for small network size, minimizing the delay gives almost the same values at all destinations. However for large networks, this is not necessary. So, at N = 10, SPT dominates KPP. CDKS and BSMA always give a failure rate above 90% in different network sizes.
In Fig. 4.14, we relax the delay variation constraint to be 15 instead of 5. GADVM gives almost zero failure rate . The performance of SPT starts by zero failure rate and then decrease as the size of the network increased. BSMA has the worst performance but it is better than Fig. 4.13.

4.6.2 Average Cost per Path

Figures 4.15 and 4.16 demonstrate the effect of changing network size on the average cost per path. The figures use the same parameters as in Figures 4.13 and 4.14.
In Fig. 4.15, we set the delay variation constraint δ =0.05. BSMA always dominates all other algorithms. CDKS dominates GADVM at $N \geq 30$. With large network size, GADVM dominates CDKS. This indicates that GADVM is more suitable in large network size. KPP gives a very high cost compared to all other algorithms.
In Fig. 4.16, we relax the delay variation constraint to be 0.15 instead of 0.05. BSMA

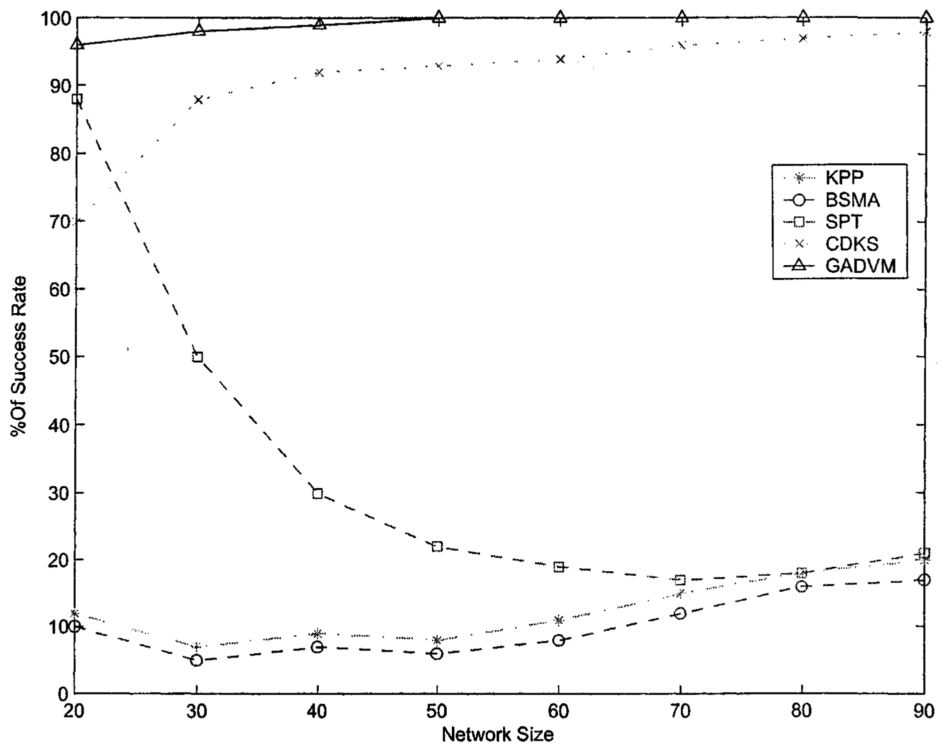

Figure 4.13: The effect of Network size on failure rate with delay variation constraint= 0.02, delay constraint=0.05, Average node degree = 8

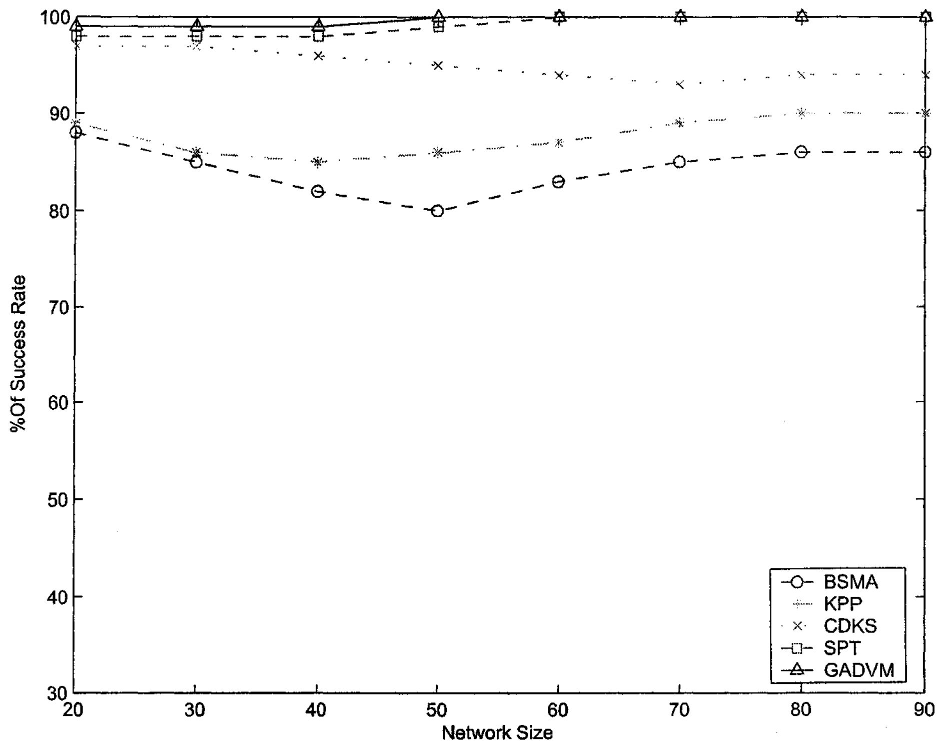

Figure 4.14: The effect of Network size on failure rate with delay variation constraint= 0.04, delay constraint=0.05, Average node degree = 8

still gets the best results. GADVM always dominates CDKS for all different network sizes. It can be observed that KPP is comparable with CDKS and even dominates it at N $\geq$ 30. This indicates that GADVM works very well for large network size and large delay variation.

4.7 Speeding up the GADVM

One of the key factors that dominates the computation complexity of GADVM is the number of generations. By reusing past solutions, the number of generations required to obtain a good solution can be significantly reduced.

In real networks, the solutions obtained for setting up a multicast connection can be reused as initial chromosomes for the next multicast request, either with the same source and destination set or not. Since the network status (e.g., cost, delay and loss probability on each link) will not change dramatically during a short period of time (e.g. the time between two consecutive multicast requests), chromosomes in the chromosome pool of the past solution are very likely to still be good solutions.

When two multicast connection requests have the same source node and destination set, the chromosomes in the past solution pool to the new initial pool is too large, the evolution process may not progress effectively. Therefore, we suggest only copying the top 50% of the chromosomes from the past solution pool. Fig. 4.15 shows the effect of reusing the past solutions. The simulations were done on a 20-node random graph. The number of destination nodes is set to 4. The dashed line with circle marks shows the costs of multicast trees when chromosomes in the initial chromosome pool are randomly generated. The solid line with trinagle marks shows the improvement on the convergence speed when the top 50% of the past solutions were reused as the initial chromosomes. Changing the cost and QoS parameters (delay, loss), simulates the fluctuations of the network traffic of each link randomly to a value within the interval of $(80\%, 120\%)$ of the original values. The same method is adopted for the following simulations. As we can observe from Fig. 4.16 that GADVM yields very good solutions very quickly, if past solutions are reused.

If two multicast requests do not have the same destination set but with the same

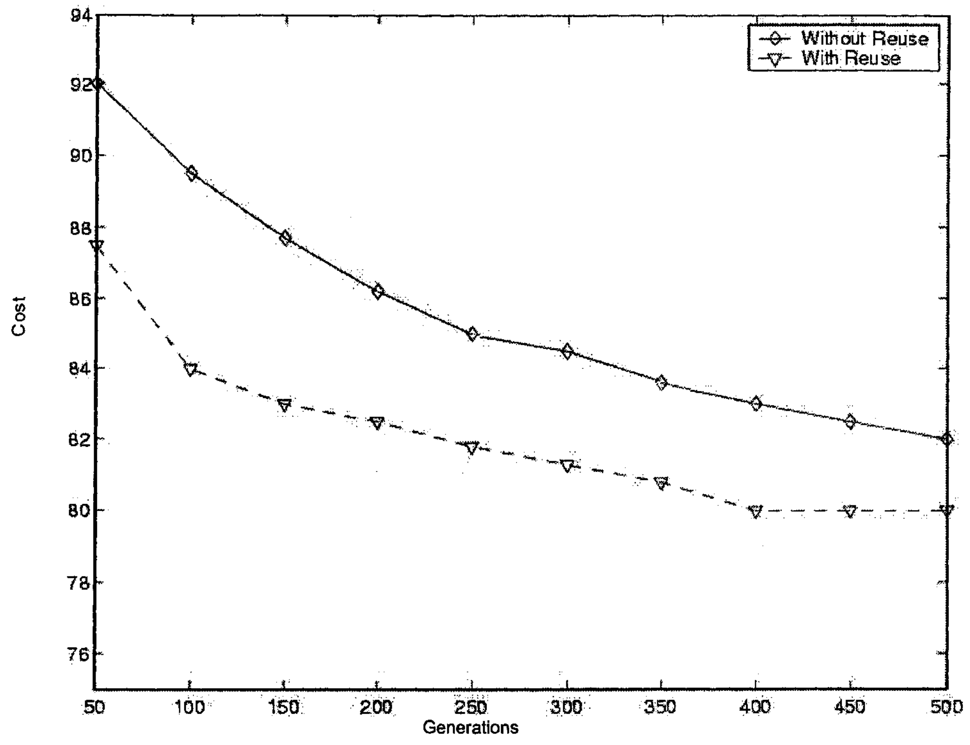

Figure 4.15: The effect of the past solution on the performance of GADVM algorithm with same source and set of destinations

source node, the chromosomes in the past-solution pool can still be reused. In particular, if the destination set of the current multicast request is a subset or superset of the previous request's destination set, the chromosomes can be reused as follows. If the current destination set is a superset of the previous destination set, the chromosomes are initialized such that genes that represent routes to destinations in the previous destination set are copied directly from the chromosomes in the past-solution pool while genes that represent new destinations are generated randomly (or with preference for shorter routes). on the other hand, if the current destination set is a subset of the previous destination set, genes of initial chromosomes are copied directly from the corresponding genes in the past solution pool.

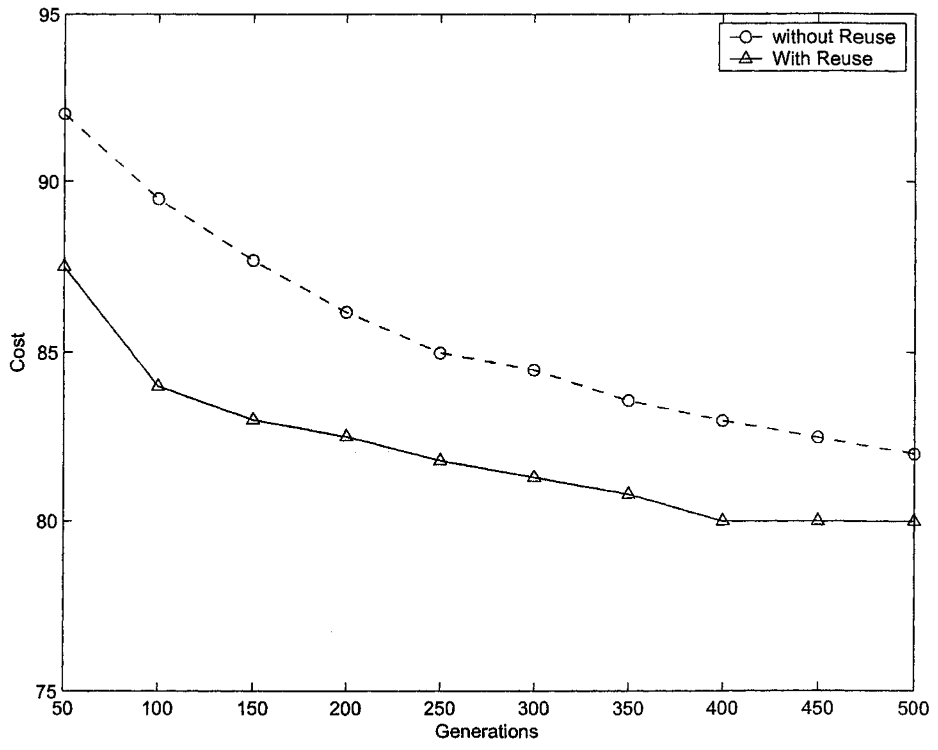

Figure 4.16: The effect of the past solution on the performance of GADVM algorithm when the destination set is different from the previous one

4.8 summary

Multimedia applications involved in real-time applications have multiple QoS requirements that must be guaranteed by the underlying network. The problem of constructing a multicast tree with multiple constraints is NP complete. In this work, we used a heuristic approach based on the genetic algorithm to tackle this problem. Our simulation results show that GADVM yields better performance than other algorithms under various sizes of random graphs.

We summarize the performance of the multiple constraint multicast algorithms studied in this work as follows. The SPT algorithm requires the lowest computation complexity, but yields the worst performance. The KPP algorithm requires the highest computation complexity, but yields worse performance than GADVM and BSMA. After all, all algorithms are able to construct low cost trees, which satisfies the given QoS parameters, but GADVM performs the best. The GADVM approach also has the advantages that no matter how QoS constraints may be satisfied, the time complexity will not increase significantly.

In this section, we have assumed that the link costs are randomly generated and the bandwidth required on each link of the multicast tree is fixed, e.g, obtained based on the effective bandwidth. However, if end to end QoS is required for each source-destination pair of a multicast connection, the bandwidth required at each link may not be the same, it would depend on the end to end QoS, the length of the path, current resource utilization of each link on the multicast tree, etc. Therefore, how to define link cost and perform QoS on the multicast tree, etc. is worth further study.

A large number of simulation experiments have been done to compare the proposed GADVM algorithm that was developed in chapter 3 for delay and delay variation shortest path problem with other algorithms mainly for delay constrained shortest path problem. We considered two performance measures, the failure rate and the average cost per path. For the failure rate, GADVM always dominates all other algorithms with a very great difference. For the average cost per path, GADVM sometimes gives results that dominate SPT and CDKS algorithms in case of large degree and large network size. On the other hand, BSMA algorithm always dominates

GADVM in terms of cost. GADVM suffers additional cost which is paid for satisfying the additional constraint of delay variation.

Chapter 5

Conclusions and Recommendations

5.1 Conclusion

This thesis considers the problem of multicast routing under special constraints that represent QoS parameters. A survey of current algorithms that deal with different kinds of constraints is presented with other problems that are related to multicast routing. The need for more algorithms for multicast routing problems was apparent.

In our work, we focus on constrained multicast routing problems . It is shown that there is a need for an algorithm for multicast routing in real-time networks that satisfies the requirements for the new applications like real time and multimedia applications.

We considered networks with delay and delay-variation constraints that represent real-time applications such as video conferencing, which not only needs to be real time, but also a certain balance between participants must be guaranteed. This balance is formulated as a delay variation constraint and added to the delay constraint which results in a new problem that is finding a shortest path route under delay and delay variation constraints. The problem studied is known to be NP-Complete. A new algorithm based on genetic algorithm for multicast routing with constraints called GADVM (Genetic Algorithm for Delay and Delay-Variation Multicast), is proposed and then applied for the delay and delay-variation constrained shortest path problem. An extensive analysis of GADVM performance is done by comparing it with other algorithms via large number of simulation experiments that simulate real networks.

Simulation results show that GADVM always dominates the other algorithms and its performance is increased at tight delay constraint and large average node degree.

Two performance measures are considered, the failure rate to satisfy the delay variation constraint and the average cost per path. GADVM gives a very low failure rate with respect to other algorithms. In addition, the cost of GADVM is comparable to other algorithms and even in sometimes dominates some algorithms.

5.2 Suggestions for Future Work

The work of this thesis can be extended in several areas:

- The proposed algorithms are centralized which means that all data must be stored in the source node. A need for a distributed implementation of these algorithms will be valuable and more scalable in case of large network size, where we will need that every node stores only a limited amount of information.
- The dynamic change of multicast group members should be considered to be embedded in the proposed algorithms so that we do not need to run the algorithm from the beginning with every change in the multicast group.
- The proposed algorithms should be incorporated in an appropriate protocol to be used in real networks.
- More constraints can be added to the proposed algorithms like the degree constraint which is suitable for the internet .

- The proposed algorithm minimizes the cost of each path individually which results in a shortest path tree, similar work could be done in minimizing the total cost of the multicast tree which is called minimum Steiner tree.
- The proposed algorithm deals with multicasting problems, similar work could be done for broadcasting and unicast problems.

Bibliography

[1] M. de Prycker,"Asynchronous Tranfer Mode, Solution for Broadband ISDN ". Prentice Hall, 3rd ed., 1995.

[2] M. Macedonia and D. Brutzman, MBone Provides Audio and Video Across the Internet, IEEE Computer, vol. 27, no. 4, pp. 30-36, April 2001.

[3] D. Cheriton and S. Deering, Host Group: "A Multicast Extension for Datagram Internetworks," in Proceedings of the Ninth ACM/IEEE Data Communications Symposium, pp. 172-179, 1985.

[4] Hussein F. Salama and S. Deering," Evaluation of multicast routing algorithms for real-time communication on high-speed networks", IEEE Journal of selec. areas in comm.,vol.15, No.3, April 1997.

[5] D. Bertsekas and R. Gallager, Data Networks. Prentice-Hall, 2nd ed., 1992.

[6] S. Casner and S. Deering," First IETF Internet Audiocast", in Proccedings of ACM SIGCOMM, Computer Communications Review, vol. 22, no. 3, July 1992.

[7] R. Karp, "Reducibility among Combinatorial Problems, in Complexity of Computer Computations" (R. Miller and J. Thatcher, eds.), pp. 85-103, Plenum Press, 1972.

[8] R. Prim, "Shortest Connection Networks and Some Generalizations",The Bell Systems Technical Journal, vol. 36, no. 6, pp. 1389-1401, November 1957.

[9] K. Clay, G.Polyzos, and H. Braun, "TraC Characteristics of the T1 NSFNET Backbone", in Proceedings of IEEE INFOCOM '93, pp. 885-893, 1993.

[10] V. Paxson, "Growth Trends in Wide-Area TCP Connections", IEEE Network, August 1994.

[11] R. Bellman, Dynamic Programming. Princeton University Press, 1975.

[12] E. Dijkstra, "Two Problems in Connection with Graphs", Numerische Mathematik, vol. 1, no. 5, pp. 269-271, October 1959.

[13] B. Awerbuch, A. Bar-Noy, and M. Gopal, "Approximate Distributed Bellman-Ford Algorithms", in Proceedings of IEEE INFOCOM '91, pp. 1206-1213, 1991.

[14] D. Bertsekas, Linear Network Optimization: Algorithms and Codes. MIT Press, 1991.

[15] Y. Dalal and R. Metcalfe, Reverse Path Forwarding of Broadcast Packets, Communications of the ACM, vol. 21, no. 12, pp. 1040-1048, December 1978.

[16] S. Deering and D. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs", ACM Transactions on Computer Systems, vol. 8, no. 2, pp. 85-110, May 2002.

[17] M. Garey and D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. New York: W.H. Freeman and Co., 1979.

[18] R. Widyono, "The Design and Evaluation of Routing Algorithms for Real-Time Channels", Tech. Rep. ICSI TR-94-024, University of California at Berkeley, International Computer Science Institute, June 1994.

[19] Q. Sun and H. Langendoerfer, "Efficient Multicast Routing for Delay-Sensitive Applications", in Proceedings of the Second Workshop on Protocols for Multimedia Systems (PROMS '95), pp. 452458, October 1995.

[20] S. Wi and Y. Choi, "A Delay-Constrained Distributed Multicast Routing Algo- rithms", in Proceedings of the Twelveth International Conference on Computer Communication (ICCC '95), pp. 833-838, 2000.

[21] F. Hwang and D. Richards," Steiner Tree Problems, Networks", vol. 22, no. 1, pp. 55-89, January 1992.

[22] P. Winter, "Steiner Problem in Networks: A Survey", Networks, vol. 17, no. 2, pp. 129-167, Summer 2001.

[23] L. Kou, G. Markowsky, and L. Berman, "A Fast Algorithm for Steiner Trees", Acta Informatica, vol. 15, no. 2, pp. 141-145, 1981.

[24] H. Takahashi and A. Matsuyama, An Approximate Solution for the Steiner Problem in Graphs, Mathematica Japonica, vol. 24, no. 6, pp. 573-577, February 1980).

[25] V. Rayward-Smith, "The Computation of Nearly Minimal Steiner Trees in Graphs" , International Journal of Mathematical Education in Science and Tech nology, vol. 14, no. 1, pp. 15-23, January/February 1983.

[26] D. Wall, "Selective Broadcast in Packet-Switched Networks", in Proceedings of the Sixth Berkeley Workshop on Distributed Data Management and Computer Networks, pp. 239-258, February 1982).

[27] D. Wall, “Mechanisms for Broadcast and Selective Broadcast”. Ph.D. thesis, Stanford University, June 1980.

[28] M. Doar and I. Leslie, "How Bad is Naive Multicast Routing”, in Proceedings of IEEE INFOCOM ’93, pp. 82-89, 1993.

[29] M. Doar, "Multicast in the Asynchronous Transfer Mode Environment”. Ph.D. thesis, University of Cambridge, January 1993.

[30] V. Rayward-Smith and A. Clare, “On Finding Steiner Vertices”, Networks, vol. 16, pp. 283-294, 1986.

[31] X. Jiang, “Routing Broadband Multicast Streams”,Computer Communications, vol. 15, no. 1, pp. 45-51, January, February 1992.

[32] X. Jiang, “Distributed Path Finding Algorithm for Stream Multicast”, Computer Communications, vol. 16, no. 12, pp. 767-775, December 1993.

[33] S. Ramanathan, “An Algorithm for Multicast Tree Generation in Networks with Asymmetric Links”, in Proceedings of IEEE INFOCOM ’96, pp. 337-344, 1996.

[34] C.H. Chow, “On Multicast Path Finding Algorithms”, in Proceedings of IEEE INFOCOM ’99, pp. 1274-1283, 1999.

[35] Y.W. Leung and T.S. Yum, “Efficient Algorithms for Multiple Destinations Routing”, in Proceedings of the IEEE International Conference on Communications (ICC ’91), pp. 1311-1317, 1991.

[36] F. Bauer and A. Varma, "Distributed Algorithms for Multicast Path Setup in Data Networks", IEEE/ACM Transactions on Networking, vol. 4, no. 2, pp. 181-191, April 1998.

[37] V. Kompella, J. Pasquale, and G. Polyzos, "Multicast Routing for Multimedia Communication", IEEE/ACM Transactions on Networking, vol.1, no.3, pp. 286-292, June 1993.

[38] V. Kompella, J. Pasquale, and G. Polyzos, "Multicasting for Multimedia Applications", in Proceedings of IEEE INFOCOM '92, pp. 2078-2085, 1992.

[39] C. Noronha and F. Tobagi, "Optimum Routing of Multicast Streams", in Proceedings of IEEE INFOCOM '94, pp. 865-873, 1994.

[40] V. Kompella, J. Pasquale, and G. Polyzos, "Two Distributed Algorithms for the Constrained Steiner Tree Problem", in Proceedings of the Second Interna- tional Conference on Computer Communications and Networking (IC 3N '93), pp. 343-349, 1993.

[41] Q. Zhu, M. Parsa, and J. Garcia-Luna-Aceves, "A Source-Based Algorithm for Delay-Constrained Minimum-Cost Multicasting", in Proceedings of IEEE INFOCOM '95, pp. 377-385, 1995.

[42] B. Waxman, "Routing of Multipoint Connections", IEEE Journal on Selected Areas in Communications, vol. 6, no. 9, pp. 1617-1622, December 1988.

[43] A. Waters, "A New Heuristic for ATM Multicast Routing", in Proceedings of the Second IFIP Workshop on Performance Modeling and Evaluation of ATM Networks, pp. 81-89, July 1999.

[44] H. Salama, D. Reeves, Y. Viniotis, and T.-L. Sheu, "Comparison of Multicast Routing Algorithms for High-Speed Networks", Tech. Rep. TR 29.1930, IBM, September 1994.

[45] B. Waxman, "Performance Evaluation of Multipoint Routing Algorithms", in Proceedings of IEEE INFOCOM '93, pp. 980-986, 1993.

[46] J. Kadirire, Minimising Packet Copies in Multicast Routing by Exploiting Geographic Spread, Computer Communication Review, vol. 24, no.3, pp.47- 62, July 1994.

[47] J. Kadirire and G. Knight, "Comparison of Dynamic Multicast Routing Algorithms for Wide-Area Packet Switched (Asynchronous Transfer Mode) Networks", in Proceedings of IEEE INFOCOM '95, pp. 212-219, 1995.

[48] E. Biersack and J. Nonnenmacher, WAVE: "A New Multicast Routing Algorithm for Static and Dynamic Multicast Groups", in Proceedings of the Fifth International Workshop on Network and Operating System Support for Digital Audio and Video, pp. 228-239, 1995.

[49] F. Bauer and A. Varma, ARIES: "A Rearrangeable Inexpensive Edge-Based On-Line Steiner Algorithm", in Proceedings of IEEE INFOCOM '96, pp. 361- 368, 1996.

[50] K. Barath-Kumar and J. Jae, "Routing to Multiple Destinations in Computer Networks", IEEE Transactions on Communications, vol. COM-31, no. 3, pp. 343-351, March 1983.

[51] G. Rouskas and I. Baldine, "Multicast Routing with End-to-End Delay and Delay Variation Constraints", in Proceedings of IEEE INFOCOM '2000', pp. 353 360, 2000.

[52] H. Tode, Y. Sakai, H. Okada, and Y. Tezuka, "Multicast Routing Algorithm for Nodal Load Balancing", in Proceedings of IEEE INFOCOM '92, pp. 2086-2095, 1992.

[53] F. Bauer and A. Varma, "Degree-Constrained Multicasting in Point-to-Point Networks", in Proceedings of IEEE INFOCOM '95, pp. 369-376, 1995.

[54] M. Ammar, S. Cheung, and C. Scoglio, "Routing Multipoint Connections Using Virtual Paths in an ATM Networks", in Proceedings of IEEE INFOCOM '93, pp. 98-105, 1993.

[55] S.B. Kim, "An Optimal VP-Based Multicast routing in ATM Networks", in Proceedings of IEEE INFOCOM '96, pp. 1302-1309, 1996.

[56] Y. Tanaka and P. Huang, "Multiple Destination Routing Algorithms", IEICE Transactions on Communications, vol. E76-B, no. 5, pp. 544-552, May 2002.

[57] L. Wei and D. Estrin, "The Trade-offs of Multicast Trees and Algorithms", in Proceedings of the Third International Conference on Computer Communications and Networking (IC 3N '94), pp. 17-24, 1994.

[58] C. Noronha and F. Tobagi, "Evaluation of Multicast Routing Algorithms for Multimedia Streams", in Proceedings of the IEEE International Telecommunications Symposium, August 1994.

[59] C. Semeria and T. Maufer, Introduction to IP Multicast Routing. Internet Draft, May 1999.

[60] S. Deering, "Host Extensions for IP Multicasting". Internet RFC 1112, http://ds.internic.net/rfc/rfc1112.txt, August 1989.

[61] W. Fenner, "Internet Group Management Protocol", Version 2. Internet Draft, May 1996.

[62] O. Hermanns and M. Schuba, "Performance Investigations of the IP Multicast Architecture", Computer Networks and ISDN Systems, vol. 28, no. 4, pp. 429- 439, February 1996.

[63] D.Waitzman, C. Partridge, and S. Deering, "Distance Vector Multicast Routing Protocol". Internet RFC 1075, http://ds.internic.net/rfc/rfc1075.txt, November 1998.

[64] V. Kumar, MBone: "Interactive Multimedia On The Internet". Macmillan Pub- lishing, Simon and Schuster, 2000.

[65] C. Hedrick, "Routing Information Protocol". Internet RFC 1058, http://ds.internic.net rfc/rfc1058.txt, June 1988.

[66] G. Malkin, RIP Version 2, "Carrying Additional Information".Internet RFC 1723, http://ds.internic.net/rfc/rfc1723.txt, November 1994.

[67] T. Pusateri, "Distance Vector Multicast Routing Protocol".Internet Draft, July 1996.

[68] J. Moy, "Mutlicast Extension to OSPF". Internet RFC 1584, http://ds.internic.net/ rfc/rfc1584.txt, May 1994.

[69] J. Moy, "MOSPF, Analysis and Experience". Internet RFC 1585, http://ds.internic.net/ rfc/rfc1585.txt, May 1994.

[70] J. Moy, "Multicast Routing Extensions for OSPF", Communications of the ACM, vol. 37, no. 8, pp. 61-66, August 1994.

[71] J. Moy, "OSPF Version 2. Internet RFC 1583", http://ds.internic.net/ rfc/rfc1583.txt, March 1994.

[72] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei, "The PIM Architecture for Wide-Area Multicast Routing", IEEE/ACM Transactions on Networking, vol. 4, no. 2, pp. 153-162, April 1996.

[73] D. Estrin et al., "Protocol Independent Multicast-Dense Mode (PIM-DM): Protocol Specification". Internet Draft, January 1996.

[74] S. Deering et al., "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification". Internet Draft, June 1996.

[75] A. Ballardie, P. Francis, and J. Crowcroft, "Core Based Trees (CBT): An Architecture for Scalable Inter-Domain Multicast Routing", in Proceedings of ACM SIGCOMM '93, pp. 85-95, September 2002.

[76] Xin Yuan, "Heuristic algorithms for multiconstrained Quality-of-service routing", IEEE/ACM Transactions on Networking, Vol.10, No2, April 2002 .

[77] A. Ballardie,"Core Based Trees (CBT) Multicast Architecture". Internet Draft, February 1996.

[78] A. Ballardie, S. Reeve, and N. Jain, "Core Based Trees (CBT) Multicast: Protocol Specification". Internet Draft, April 1996.

[79] T. Billhartz, J. Cain, E. Farrey-Goudreau, D. Fieg, and S. Batsell, "Performance and Resource Cost Comparisons for CBT and PIM Multicast Routing Protocols in DIS Environments", in Proceedings of IEEE INFOCOM '96, pp. 8593, 1996.

[80] Z. Zhang, C. Sanchez, B. Salkewicz, and E. Crawly, "Quality of Service Extensions to OSPF (QOSPF)". Internet Draft, June 1996.

[81] E. Crawley,"Multicast Routing over ATM". Internet Draft, February 1996.

[82] J. H. Holland, "Adaption in Natural and Artificial Systems", Boston, MA: MIT Press,1992.

[83] H. Pan and I. Y. Wang, "The bandwidth allocation of ATM through genetic algorithm", IEEE GLOBECOM '91, 1991, pp. 125-129.

[84] C. C. Palmer and A. Kershenbaum, "Two algorithms for finding optimal communication spanning trees", Technical Report, IBM T. J. Watson Research Center, Yorktown Heights, NY, 1993.

[85] V. J. Rayward-Smith, "The computation of nearly minimal Steiner trees in graphs", International Journal Mathematic Educational Science Technology, Vol. 14, No. 1,1983, pp. 15-23.

[86]Yee Leung,Guo Li, and Zong-Ben Xu , "A genetic Algorithm for the Multiple destination routing problems" , IEEE Transaction on evolutionary computation, Vol.2, No. 4, November 1998.

[87] M. Hamdan and M. El-Hawary , "A novel genetic algorithm searching approach for dynamic constrained multicast routing" , CCECE , May 2003.

[88] M. Hamdan and M. El-Hawary , "Genetic algorithm for mulitcast routing with delay and delay variation constraints", Accepted for publication, CCECE , May 2004.

[89] Chang wook Ahn and R.S. Ramakrishna, "A genetic algorithm for shortest path routing problem and the sizing of populations", IEEE Transaction on Evolutionary computation, Vol.6, No.6, December 2002.

[90]D. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning , The MIT Press, 1996.

[91] A. Ballardie, S. Reeve, and N. Jain, " The P Versus NP Problem ", http://www.claymatl
vs-NP[2004,5 January].

Appendix A

Complexity Theory and NP problem

There are many practical problems for which no polynomial-time algorithm is currently known to exist. Computational complexity theory has shown that there are many problems which indeed have no polynomial-time or even exponential-time solution. There exist problems of arbitrary complexity. Many problems we encounter usually admit a good network or integer *LP* model and the question arises why it is so hard to find a polynomial-time algorithm for them. In 1971, Cook[91], showed that the essence of the question is a major open problem in complexity theory: the P-versus-NP question.

In this appendix a brief description of the theory of NP-completeness is given, .

The definition of polynomial-time computability and similar notions requires an underlying model of computation with a fair instruction set.

A.1 Size of the Problem

When we refer to the size of a problem (or variable, or instance) we mean the size of its specification in bits, unless explicitly stated otherwise. Problem specifications must also be finite. Thus, if any numbers are involved, they are normally restricted to being integers or rationals. The size of problem instances and other bit strings x will be denoted by x. An algorithm is said to be polynomial if its running time on instances of size $n = x$ is bounded by n^k for some fixed k.

Problem size is sometimes measured more loosely. For example, for networks G with n nodes one often uses $|G| = n$, even though it may take $O(nlogn + mlogn) = O(n2logn)$ bits to specify the network and its m edges. Note that for integers n, $|n| = logn$. An algorithm has a single integer n as input, e.g. a primality tester, has polynomial running time if it runs in time O((log n).

A.2 Computational Problems

Computational Problem can be defined as, Given an input x that is encoded over the alphabet $\{0, 1\}$, find an output y that satisfies some property. The computational problem is then described by the property that the output has to satisfy given the input .

There are four types of computational problems, decision problem, research problem, optimization problems, and counting problems

Definition: The relation $R \subseteq \{0, 1\}\, x\, \{0, 1\}$ is polynomially-bounded if there exists a polynomial p such that for every$(x, y) \in R$ it holds that $|y| \leq p\,(|x)|$.

A.3 Search Problem

P as a natural class of search problems. With each polynomially bounded relation R, we associate the following search problem: given x find y such that $(x; y) \in R$ or state that no such y exists. The class P corresponds to the class of search problems that are solvable in polynomial time. NP as another natural class of search problems NP as another natural class of search problems. A polynomially bounded relation R is called an NP relation if given an alleged instance solution pair one can efficiently verify whether the pair is valid; that is, there exists a polynomialtime algorithm that

given x and y determines whether or not $(x; y) \in R$. It is reasonable to focus on search problems for NPrelations, because the ability to recognize a valid solution seems to be a natural prerequisite for a discussion regarding finding such solutions.
The P versus NP question in terms of search problems: states that, Is it the case that the search problem of every NP relation can be solved in polynomial time? In other words, if it is easy to test whether a solution for an instance is correct then is it also easy to find solutions to given instances? If P = NP then this would mean that if solutions to given instances can be efficiently verified for correctness then they can also be efficiently found (when given only the instance). This would mean that all reasonable search problems (i.e., all NP relations) are easy to solve. On the other hand, if $P \neq NP$ then there exist reasonable search problems (i.e., some NP relations) that are hard to solve. In such a case, the world is more interesting: some reasonable problems are easy to solve whereas others are hard to solve.

A.4 The decision Problem

For an NP relation R, we denote the set of instances having solution by L_R ; that is, $L_R = x : \exists y(x; y) \in R$. Such a set is called an NP set. Intuitively, an NP set is a set of valid statements (i.e., statements of membership of a given x in L_R that can be efficiently verified given adequate proofs (i.e., a corresponding NP witness y such that $(x; y) \in R$).

NP-proof systems. Proof systems are defined in terms of their verification procedures. Here we focus on the natural class of efficient verification procedures, where efficiency is represented by polynomialtime computations. (We should either require that the time is polynomial in terms of the statement or confine ourselves to "short proofs",that is, proofs of length that is bounded by a polynomial in the length of the statement.) An NPrelation R yields a natural verification procedure, which amounts to checking whether the alleged statementproof pair is in R. This proof system satisfies the natural completeness and soundness conditions: every true statement ($i.e., x \in L_R$) has a valid proof (i.e., an NP-witness y such that $(x; y) \in R$), whereas false statements (i.e., $x \notin L_R$) have no valid proofs (i.e., $(x; y) \notin R$ for all $y's$).

The P versus NP question in terms of decision problems: Is it the case that NP proofs are useless? That is, is it the case that for every efficiently verifiable proof system one can easily determine the validity of assertions without given suitable proofs. If that were the case, then proofs would be meaningless, because they would have no fundamental advantage over directly determining the validity of the assertion. Recall that P is the class of sets that can be decided efficiently (i.e., by a polynomial time algorithm). Then the conjecture $P \neq NP$ asserts that proofs are useful: there exists NPsets that cannot be decided by a polynomialtime algorithm, and so for these sets obtaining a proof of membership (for some instances)is useful (because we cannot determine membership by ourselves).

A.5 NP Complete

The NP problem is NP complete if it is NP is reducible to it, and the polynomial-bounded relation is NP complete if it is an NP relation and every NP-relation is reducible to it.