QOS MULTICAST ROUTING USING ANT AND GENETIC ALGORITHMS

A DISSERTATION

Submitted in partial fulfillment of the requirements for the award of the degree

of

MASTER OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

By

VEMURI KRISHNA SUMANTH

CENTRAL LIBRARY
G11918
Acc. No.
Date 10-11-04
I.I.T. ROORKEE

The Supercomputing People

IIT Roorkee - CDAC, NOIDA,
c-56/1, "Anusandhan Bhawan"
Sector 62, Noida-201 307

JUNE, 2004

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this dissertation titled "**QOS MULTICAST ROUTING USING ANT AND GENETIC ALGORITHMS**", in partial fulfillment of the requirements for the award of the degree of **Master of Technology** in **Information Technology**, submitted in **IIT-Roorkee - CDAC, NOIDA,** is an authentic record of my own work carried out during the period from June 2003 to June 2004 under the guidance of **Dr. MOINUDDIN**, Professor, Jamia Millia Islamia, New Delhi.

I have not submitted the matter embodied in this dissertation for the award of any other degree or diploma.

Date: 25/6/2004
Place: Noida

[illegible]
(VEMURI KRISHNA SUMANTH)

CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge and belief.

Date: 25/6/2004
Place: Noida

[illegible]
(Dr. MOINUDDIN),
Professor,
Jamia Millia Islamia,
New Delhi

ACKNOWLEDGEMENTS

I express my sincere gratitude to my guide **Dr. MOINUDDIN,** Professor, Jamia Millia Islamia, New Delhi, for his inspiring guidance and sustained interest throughout the progress of this dissertation.

I here by take the privilege to express my deep sense of gratitude to **Prof. PREM VRAT,** Director, Indian Institute of Technology, Roorkee, and **Mr. R.K.VERMA,** Executive Director, C-DAC, NOIDA for providing me with the valuable opportunity to carry out this work.

I am very grateful to **Prof. A.K.AWASTI,** Programme Director, **Prof. R.P. AGARWAL,** Course Coordinator, Indian Institute of Technology, Roorkee and **Mr. V.N.SHUKLA,** Course Coordinator, CDAC, NOIDA for providing the best of the facilities for the completion of this work and constant encouragement towards the goal.

I am thankful to **Mr. R.K.SINGH** and **Mr. MUNISH KUMAR,** project engineer, C-DAC NOIDA, for providing necessary infrastructure to complete the dissertation in time.

I thank my family members for their moral support.

[illegible]

(VEMURI KRISHNA SUMANTH)

Enroll. No. 029009

ABSTRACT

With the rapid development of Internet, mobile networks and high-performance networking technology, QoS multicast routing in networks has become a very important research issue in the areas of networks and distributed systems and considered as a challenging and hard problem for the next generation Internet and high-performance networks. This dissertation discusses the heuristic algorithms for multicast routing problem with multiple QoS constraints in networks. In this dissertation, Evolutionary Algorithms like Ant and Genetic Algorithms are developed that are capable to deal effectively by issues related with multicast routing for the internet and can provide QoS-sensitive paths in a scalable and flexible way. A multicast tree selection algorithm based on Non-dominated Sorting technique of Genetic Algorithms is also discussed to simultaneously optimize multiple QoS parameters. These algorithms optimize the network resources like end-to-end-delay and delay-jitter, and can converge to the optimal or near-optimal solution within a few iteration, even for the network environment with uncertain parameters. The incremental rate of the computational cost of these algorithms can be close to polynomial and is less than exponential rate. The simulations carried out shows that these heuristic algorithms converge to the near optimal solutions in lesser number of iterations. Among these algorithms, Ant Algorithm converges faster than the Genetic Algorithms. The scalability of these algorithms especially, the Ant Algorithm, proved to be noteworthy when compared to the traditional QoS multicast routing algorithms.

CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Multicast is a communication service that allows simultaneous transmission of the same message from one source to a group of destination nodes. To implement a multicast session, a network must minimize the session's resource consumption while meeting the quality of service (QoS) requirements. An efficient allocation of network resources to satisfy the different QoS requirements is the primary goal of QoS-based multicast routing. However the inter-dependency and confliction among multiple QoS parameters make the problem difficult. It has been demonstrated that it is NP-complete to find a feasible multicast tree with two independent additive path constraints [1]. NP-complete problems cannot be solved in the polynomial time and heuristics are required to solve these kind of problems. Although QoS routing over the communication networks is an active research area in the recent years, QoS-based multicast is a relatively new research topic. In addition to requiring scalable and efficient network support, multicast applications usually demand stringent QoS requirements.

1.1 Motivation

With the introduction of QoS, the multicast problem becomes more challenging. The QoS requirements can be classified into link constraints (e.g., bandwidth), path constraints (e.g., end-to-end delay) and tree constraints (e.g., delay jitter). Over the past decades, many unconstrained or single constrained multicast routing algorithms have been developed. Typical approaches [1] include 1) Applying Dijkstra's algorithm to find shortest path, 2) seeking the minimum network cost using Steiner tree routing algorithm, and 3) finding multicast trees that the paths between source nodes and the destinations are connected and their cost is minimized.

A number of efficient heuristics or nature based algorithms have been proposed. Worth noting is that the number of studies that apply the genetic and ant algorithms to solve the QoS Multicast Routing (QMR) Problems (with different types of QoS constraints) is increasing **[13-16, 38-40]**. Ant and Genetic algorithms are recently emerged as new heuristics that can efficiently solve large scale optimization problems. Ant algorithms are based on the ability of ants to find the shortest path between their nest and the food source during their looking for food. Some Genetic Algorithms have been used to solve the NP-complete problem from different aspects **[38-40]**. Although having solved the Steiner tree problem effectively, some of them only consider one evaluation metric and cannot be extended directly to solve the multicast routing problem with multiple QoS constraints. A careful analysis of these optimization schemes reveals that they suffer from the same drawback: multiple QoS objectives or constraints are combined to form a scalar single-objective function on an ad hoc basis, usually through a linear combination of weighted sum by different requirements. In most of the algorithms, the QoS constraints were often transformed into penalty functions and combined with a cost function to be taken as the fitness function. Therefore the solution not only becomes highly sensitive to the weight vector but also demands the user to have certain knowledge about the problem, e.g. weight of a particular constraint. GA can be modified to optimize multiple QoS requirements simultaneously by incorporating the concept of Pareto dominance in genetic selection operation **[31]**. Genetic Algorithms are well suited to Multi-Objective Optimization problems (MOP). Multiple parameters can be formulated as a Multi-Objective model. Multiobjective model based on GA is used to approximate Pareto front by generating a set of Nondominated solutions. This approach works in a source-based fashion, and it assumes the complete knowledge of a network is available.

1.2 Organization of the dissertation

The dissertation is organized as follows. Chapter 2 describes the related work. Chapter 3 gives the introduction of the Ant and Genetic algorithms It also explains the QoS Multicast Routing Problem. Chapter 4 gives the design of the algorithms for finding the optimal multicast trees satisfying the QoS requirements. Chapter 5 gives the implementation details of the algorithms. Chapter 6 presents the results obtained from the simulations which is followed by Conclusion and Future work.

Chapter 2

LITERATURE SURVEY

2.1 Introduction

This chapter deals with the literature survey pertaining to the previous work on multicast routing, its importance, properties of multicast trees, and significance of the delay and delay-jitter constraints. The next two sections deals with the background of Ant and Genetic Algorithms and its significances followed by the mathematical formulation of the QoS multicast Routing Model.

2.2 Issues in QoS Multicast Routing

Multicast provides the efficient way of disseminating the data from a source to all the members in the multicast group. Instead of sending a separate copy of the data to each individual group member, the source just sends a single copy to all the destinations members. An underlying multicast routing protocol determines, with respect to certain optimization objectives, a multicast tree connecting the source and the group members. Data generated by the source node flows through the multicast tree, traversing each tree edge exactly once.

The approaches for constructing the multicast trees can be classified into two categories [1]: 1) The Source-based Multicast Tree Approach e.g., Distance Vector Multicast Routing Protocol (DVMRP) and 2)The Core-based Multicast Tree Approach e.g., the Core-based Tree Protocol (CBT). In the Source-based approach, a tree rooted at the source node is constructed and connected to every member in the multicast group. Data packets originated at the source node are sent to all destination nodes via the links of the multicast tree. In the core-based approach, one node for each group is selected as the core. A tree rooted at the core is the constructed to all the group members. Data packets flow from any source to its parent and children. A node forwards packets to its parents and children except the one from which the data packets arrive.

Several optimization objectives have been used to construct the multicast trees **[49].** One popular optimization method is to minimize the sum of the costs on the links of a multicast tree. The minimum cost tree is called as Steiner Tree, and finding such a tree is NP-Complete problem. That means it could not be solved in the polynomial time. Several researchers have formulated, under one-to-many source based multicast paradigm, the problem of constructing multicast trees as a Steiner tree problem with side constraints such as delay, delay-jitter and bandwidth. While the total tree cost is an important measure of bandwidth efficiency, it is not sufficient to characterize the Quality of Service (QoS) required by either embedded real-time applications or multimedia-integrated services, where the bounded delay and/or a bounded delay-jitter are usually a major criteria.

The quality of a Multicast tree is specified with the following two parameters **[47,48]**:

- Source-destination Delay Bound, D, represents an upper bound on the acceptable end-to-end delay along the path fon the source node to the each of the destination.. This parameter specifies that packets transmitted from node v at time t must be received by the destination node by the time $t+D$.
- Inter-Destination Delay-Jitter Bound, represents the maximum difference that can be tolerated between the end-to-end delays along the paths from a source nodes to any two destinations in the multicast group. This parameter specifies the requirement that the times at which data packets are received at the destination nodes have to be synchronized within the window of size equal to the delay-jitter bound.

The need for a bounded end-to-end delay is well justified. The situation in which a bounded interdestination delay jitter among all the group members arises is not rare. For example, one possible scenario occurs during a teleconference in which any current speaker should be heard by all the participants at approximately the same time to achieve the feeling of multi-party interactive face-to-face discussions.

Although the bounded source-destination delay jitter is also an important QoS requirement, it is not considered because of the following reasons[1]:

1) While both the source-destination delay and the interdestination delay jitter are related to how a tree is constructed, the source-destination delay jitter is mainly an attribute of the path from the source to the destination, rather than a tree property.
2) The source-destination delay jitter requirement is often met by buffering packets at the destination so as to absorb the delay jitter.

Existing Algorithms like Jia's Distributed Algorithm [4], QoSMIC [42] are not suitable for large scale communication networks due to their excessive message processing overheads. Determining multicast routes between a single source and the multiple destinations is computationally intractable as the problem is NP-Complete. The general approach in solving such problems is to use certain heuristics to get a near-optimal solution in polynomial time. More recently, researches in determining QoS-based multicast routes clearly demonstrate the power of ant and genetic algorithms to get a near optimal solution satisfying the QOS constraints in computationally feasible time.

2.3 Ant Algorithms and its Background

Ant algorithms were first proposed by Dorigo and colleagues [6,7] as a multi-agent approach to difficult combinatorial optimization problems such as the traveling salesman problem (TSP) and the quadratic assignment problem . There is currently much ongoing activity in the scientific community to extend and apply ant-based algorithms to many different discrete optimization problems [8,9]. Recent applications cover problems such as routing in communications networks, QoS multicast Routing and so on.

Ant algorithm is a simulated evolution algorithm based on population and ant colony behaviors. The behaviors of a single ant is simple, however a population of ants may behave very complicated, which can complete complex tasks and even adapt to the change of environment. For example, when an obstacle appears on the moving path of an ant population, ants can find a new optimal path quickly.

Although the bounded source-destination delay jitter is also an important QoS requirement, it is not considered because of the following reasons[1]:

1) While both the source-destination delay and the interdestination delay jitter are related to how a tree is constructed, the source-destination delay jitter is mainly an attribute of the path from the source to the destination, rather than a tree property.
2) *The source-destination delay jitter requirement is often met by buffering packets* at the destination so as to absorb the delay jitter.

Existing Algorithms like Jia's Distributed Algorithm [4], QoSMIC [42] are not suitable for large scale communication networks due to their excessive message processing overheads. Determining multicast routes between a single source and the multiple destinations is computationally intractable as the problem is NP-Complete. The general approach in solving such problems is to use certain heuristics to get a near-optimal solution in polynomial time. More recently, researches in determining QoS-based multicast routes clearly demonstrate the power of ant and genetic algorithms to get a near optimal solution satisfying the QOS constraints in computationally feasible time.

2.3 Ant Algorithms and its Background

Ant algorithms were first proposed by Dorigo and colleagues [6,7] as a multi-agent approach to difficult combinatorial optimization problems such as the traveling salesman problem (TSP) and the quadratic assignment problem . There is currently much ongoing activity in the scientific community to extend and apply ant-based algorithms to many different discrete optimization problems [8,9]. Recent applications cover problems such as routing in communications networks, QoS multicast Routing and so on.

Ant algorithm is a simulated evolution algorithm based on population and ant colony behaviors. The behaviors of a single ant is simple, however a population of ants may behave very complicated, which can complete complex tasks and even adapt to the change of environment. For example, when an obstacle appears on the moving path of an ant population, ants can find a new optimal path quickly.

An ant excrete a material, called pheromone, along the path on which it moves. Ants can sense this material and detect its intensity. They can then use pheromone intensity as a guide o move and tend to move toward that direction of higher intensity, thus the ants can find the food by this kind of information exchange.

The key features of an ant algorithm include distributed computation, positive feedback, and constructive greedy heuristic. These features can help to provide premature convergence ant find a very good solution in a shorter period of time. Since it inception, the algorithm has emerged as a new heuristic to solve many stochastic combinatorial optimization problems.

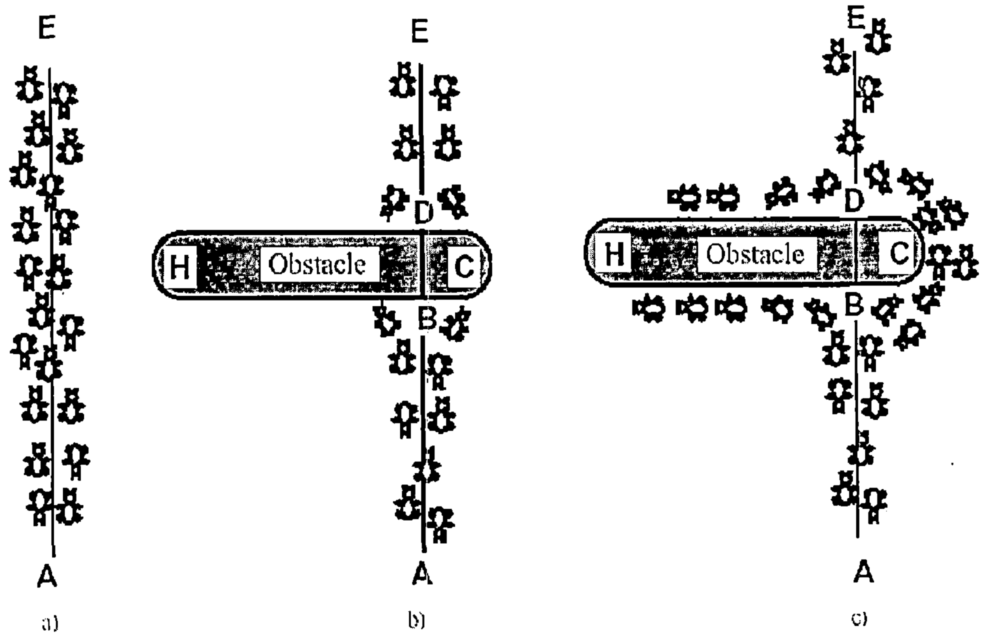

Fig 2.1: *An example with real ants.*
a) Ants follow a path between points A and E.
b) An obstacle is interposed; ants can choose to go around it following one of the two different paths with equal probability.
c) On the shorter path more pheromone is laid down.

Ant System (AS) is an algorithm, inspired by the ways ants organize themselves in nature, to optimize certain hard problems. The classic example tries to solve the traveling salesman problem (TSP) and is thus called AS-TSP. Assume a problem with n cities and k ants. Every ant makes a tour, visiting each city exactly once. In each city the ant makes a decision on where to go next with a certain probability based on visibility and pheromone level between that city and every other city. When all ants have completed their tour, the new pheromone levels in the problem are calculated; a certain percentage of pheromones evaporates and new pheromones are deposited based on the length of the tours that visited that road (shorter tour implies more pheromones). The roads on the tour of the best ant, are given an additional pheromone dose.

2.3.1 The Ant Colony Optimization Approach

In the Ant Colony Optimization (ACO) metaheuristic a colony of artificial ants cooperates in finding good solutions to difficult discrete optimization problems. Cooperation is a key design component of ACO algorithm: The choice is to allocate the computational resources to a set of relatively simple agents (artificial ants) that communicate indirectly by stigmergy. Good solutions are an emergent property of the agents' cooperative interaction. Artificial ants have a double nature. On the one hand, they are an abstraction of those behavioral traits of real ants that seemed to be at the heart of the shortest path-finding behavior observed in real ant colonies. On the other hand, they have been enriched with some capabilities that do not find a natural counterpart. In fact, we want ant colony optimization to be an engineering approach to the design and implementation of software systems for the solution of difficult optimization problems. It is therefore reasonable to give artificial ants some capabilities that, although not corresponding to any capacity of their real ant counterparts, make them more effective and efficient.

Important characteristics of Artificial Ants are the following:

- Artificial ants have an internal state. This private state contains the memory of the ants' past actions.
- Artificial ants deposit an amount of pheromone that is a function of the quality of the solution found.
- Artificial ants live in a discrete world and their moves consist of transitions from discrete states to discrete states.
- Artificial ants' timing in pheromone laying is problem dependent and often does

not reflect real ants' behavior. For example, in many cases artificial ants update pheromone trails only after having generated a solution.

➢ To improve overall system efficiency, ACO algorithms can be enriched with extra capabilities such as lookahead, local optimization, backtracking, and so on that cannot be found in real ants. In many implementations ants have been hybridized with local optimization procedures (as discussed in **[10, 11, 12]**). There are no examples of backtracking procedures added to the basic ant capabilities, except for simple recovery procedures used by Di Caro and Dorigo **[13]**.

2.3.2 Ant System

Ant System was the first developed ACO algorithm **[15, 16]**. Its importance resides mainly in being the prototype of a number of ant algorithms that have found many interesting and successful applications. In AS, artificial ants build solutions (tours) of the TSP by moving on the problem graph from one city to another. During each iteration m ants build a tour executing n steps in which a probabilistic decision (state transition) rule is applied. In practice, when in node i the ant chooses the node j to move to, and the arc (i,j) is added to the tour under construction. This step is repeated until the ant has completed its tour.

Three AS algorithms have been defined **[17, 18, 19, 20]** that differ by the way pheromone trails are updated. These algorithms are called ant-density, ant-quantity, and ant-cycle. In ant-density and ant-quantity ants deposit pheromone while building a solution, while in ant-cycle ants deposit pheromone after they have built a complete tour. Preliminary experiments run on a set of benchmark problems **[18,19,20]** have shown that ant-cycle's performance was much better than that of the other two algorithms. Consequently, research on AS was directed toward a better understanding of the characteristics of ant-cycle, which is now known as Ant System, while the other two algorithms were abandoned. As we said, in AS after ants have built their tours, each ant deposits pheromone on pheromone trail variables associated to the visited arcs to make the visited arcs become more desirable for future ants (i.e., online delayed pheromone update is at work). Then the ants die. In AS no daemon activities are performed, while the pheromone evaporation procedure, which happens just before ants start to deposit pheromone, is interleaved with the ants' activity.

The amount of pheromone trail $\tau_{ij}(t)$ associated to arc (i,j) is intended to represent the learned desirability of choosing city j when in city i (which also corresponds to the desirability that the arc (i,j) belongs to the tour built by an ant). The pheromone trail information is changed during problem solution to reflect the experience acquired by ants during problem solving. Ants deposit an amount of pheromone proportional to the quality of the solutions they produced: The shorter the tour generated by an ant, the greater the amount of pheromone it deposits on the arcs that it used to generate the tour. This choice helps to direct search toward good solutions. The main role of pheromone evaporation is to avoid stagnation, that is, the situation in which all ants end up doing the same tour.

The memory (or internal state) of each ant contains the already visited cities and is called tabu list (in the following we will continue to use the term tabu list to indicate the ant's memory). The memory is used to define, for each ant k, the set of cities that ant located on city i still has to visit. By exploiting the memory, therefore, an ant k can build feasible solutions by an implicit state-space graph generation (in the TSP this corresponds to visiting a city exactly once). Also, memory allows the ant to cover the same path to deposit online delayed pheromone on the visited arcs.

The ant-decision table $\mathrm{A}_i = [a_{ij}(t)]_{|N_i|}$ of node i is obtained by the composition of the local pheromone trail values with the local heuristic values as follows:

$$a_{ij}(t) = \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}{\sum\limits_{l \in \mathrm{N}_i}[\tau_{il}(t)]^{\alpha}[\eta_{il}(t)]^{\beta}}, \forall j \in \mathrm{N}_i \qquad - \quad - \quad - \quad (2.1)$$

where $\tau_{ij}(t)$ is the amount of pheromone trail on arc (i,j) at time t and $\eta_{ij} = 1/d_{ij}$ is the heuristic value of moving from node i to node j , N_i is the set of neighbors of node i, and α and β are two parameters that control the relative weight of pheromone trail and heuristic value. The probability with which an ant k chooses to go from city i to city $j \in N_i^k$, while building its tour at the t-th algorithm iteration is

$$P_{ij}^k(t) = \frac{a_{ij}(t)}{\sum_{l \in N_i^k} a_{il}(t)} \quad - \quad - \quad - \quad - \qquad (2.2)$$

where $N_i^k \subseteq N_i$ is the set of nodes in the neighbourhood of node i that ant k has not visited yet (nodes in N_i^k are selected from those in N_i by using the ant private memory M^k). The role of the parameters α and β is the following: if α=0, the closest cities are more likely to be selected: This corresponds to a classical stochastic greedy algorithm (with multiple starting points since ants are initially randomly distributed on the nodes). If, on the contrary, β=0 , only pheromone amplification is at work: This method will lead to the rapid emergence of a stagnation situation with the corresponding generation of tours that, in general, are strongly suboptimal **[20]**. A trade-off between heuristic value and trail intensity therefore appears to be necessary. After all ants have completed their tour, pheromone evaporation on all arcs is triggered, and then each ant k deposits a quantity of pheromone $\Delta\tau_{ij}^k(t)$ on each arc that it has used:

$$\Delta T_{ij}^k(t) = 1/L^k(t) \qquad if(i,j) \in T^k(t) \quad - \quad - \quad - \qquad (2.3)$$

where $T^k(t)$ in the symmetric TSP arcs, and for which the pheromone trail level on the arcs (i,j) and (j,i) can considered to be bidirectional so that arcs (i,j) and (j,i) are always updated contemporaneously (in fact, they are the same arcs). Different is the case of the asymmetric TSP, where arcs have a directionally be different. In this case, therefore, when an ant moves from node i to node j only arc (i,j) and not (j,i) is updated. It is clear from Equation 4 that the value $\Delta\tau_{ij}^k(t)$ depends on how well the ant has performed: The shorter the tour done, the greater the amount of pheromone deposited. In practice, the addition of new pheromone by ants and pheromone evaporation are implemented by the following rule applied to all the arcs:

$$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \text{ where } \Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau^k{}_{ij}(t) \quad - \quad - \qquad (2.4)$$

where m is the number of ants at each iteration (maintained constant), and $\rho \in (0,1]$ is the pheromone trial decay coefficient. The initial amount of pheromone $\tau_{ij}(0)$ is set to a same small positive constant value τ_0 on all arcs, and the total number of ants is set to m=n, while α , β and ρ are respectively set to 1, 5, and 0.5; these values were experimentally found to be good by Dorigo **[6]**.Dorigo et al. **[19]** introduced also elitist ants, that is, a daemon action by which the arcs used by the ant that generated the best tour from the beginning of the trial get extra pheromone.

2.3.3 Salient Features of Ant Algorithms

Ant Algorithms have the following advantages:

1. **Scalability**: Population of the agents can be adapted according to the network size. Scalability is also promoted by local and distributed agent interactions.
2. **Fault tolerance**: Swarm intelligent processes do not rely on a centralized control mechanism. Therefore the loss of a few nodes or links does not result in catastrophic failure, but rather leads to graceful, scalable degradation.
3. **Adaptation**: Agents can change, die or reproduce, according to network changes.
4. Speed: Changes in the network can be propagated very fast, in contrast with the Bellman-Ford algorithm
5. **Modularity**: Agents act independently of other network layers
6. **Autonomy**: Little or no human supervision is required.
7. **Parallelism**: Agent's operations are inherently parallel.

2.4 Genetic Algorithms

Genetic algorithms (GAs) are optimization techniques based on the concepts of natural selection and genetics. In this approach, the variables are represented as genes on a chromosome. GAs features a group of candidate solutions (population) on the response surface. Through natural selection and the genetic operators, mutation and recombination, chromosomes with better fitness are found. Natural selection guarantees that chromosomes with the best fitness will propagate in future populations. Using the recombination operator, the GA combines genes from two parent chromosomes to form two new chromosomes (children) that have a high probability of having better fitness than their parents. Mutation allows new areas of the response surface to be explored. GAs offer a generational improvement in the fitness of the chromosomes and after many generations will create chromosomes containing the optimized variable settings.

2.4.1 Genetic Representation

A chromosome of the proposed GA consists of sequences of positive integers that represent the IDs of nodes through which a routing path passes. Each locus of the chromosome represents an order of a node (indicated by the gene of the locus) in a routing path. The gene of first locus is always reserved for the source node. The length of the chromosome is variable, but it should not exceed the maximum length , where is the total number of nodes in the network, since it never needs more than number of nodes to form a routing path. A chromosome (routing path) encodes the problem by listing up node IDs from its source node to its destination node based on topological information database (routing table) of the network. The information can be easily obtained and managed in real-time by routing protocols in wired or wireless environments, It is noted that the topological information database of the network can be constructed easily and rapidly by such routing protocols.

S	N_1	N_2	N_3	N_4	……..	N_{k-1}	N_k	D

Fig 2.2: Example of routing path and its encoded chromosome

An example of chromosome (routing path) encoding from node S to node D is shown, The chromosome is essentially a list of nodes along the constructed path, ($S \rightarrow N_1 \rightarrow N_2 \rightarrow N_3 \rightarrow$……….$N_{k-1} \rightarrow N_k \rightarrow D$). The gene of the first locus encodes the source node, and the gene of second locus is randomly or heuristically selected from the nodes connected with the source node(S) that is represented by the front gene's allele. A chosen node is removed from the topological information database to prevent the node from being selected twice, thereby avoiding loops in the path. This process continues until the destination node is reached. It is noted that an encoding is possible only if each step of a path passes through a physical link in the network.

2.4.2 Population Initialization

In general, there are two issues to be considered for population initialization of GAs: the initial population size and the procedure to initialize the population [27, 22]. It was felt that the population size needed to increase exponentially with the complexity of the problem (i.e., the length of the chromosome) in order to generate good solutions. Recent studies have shown, however, that satisfactory results can be obtained with a much smaller population size. To summarize, a large population is quite useful, but it demands excessive costs in terms of both memory and time [27, 22]. As would be expected, deciding adequate population size is crucial for efficiency. Secondly, there are two ways to generate the initial population: heuristic initialization and random initialization. Although the mean fitness of the heuristic initialization is already high so that it may help the GAs to find solutions faster, it may just explore a small part of the solution space and never find global optimal solutions because of the lack of diversity in the population [22]. Physically, the random initialization chooses genes (nodes) from the topological information database in a random manner during the encoding process. It is possible that the algorithm encounters a node for which all of whose neighboring nodes have already been visited. In this case, the defective chromosome is refreshed and reinitialized. This may induce a subtle bias in which some partial paths are more likely to be generated. However, the meager bias does not significantly affect the performance of the algorithm. It is doubly so because it (the bias) vanishes after evolving just a few generations.

2.4.3 Fitness Function

The fitness function interprets the chromosome in terms of physical representation and evaluates its fitness based on traits of being desired in the solution [21,22]. Is is desirable that the fitness function must accurately measure the quality of the chromosomes in the population. The definition of the fitness function, therefore, is very critical [22,24]. The fitness function in the Shortest Path routing problem is obvious because the Shortest Path computation amounts to finding the minimal cost path. Therefore, the fitness function that involves computational efficiency and accuracy (of the fitness measurement) is defined as follows:

$$f_i = \frac{1}{\sum_{j=1}^{l_i - 1} C_{g_i(j), g_i(j+1)}} \quad - \quad - \quad - \quad - \qquad (2.5)$$

where f_i represents the fitness valueof the ith chromosome, l_i is the length of the ith chromosome, g_i (j) represents the gene(node) of the jth locus in the ith chromosome, and C is the cost between the links. The fitness function of GAs is generally the objective function that requires to be optimized **[27, 22]**. The fitness function has a higher value when the fitness characteristic of the chromosome is better than others. In addition, the fitness function introduces a criterion for selection of chromosomes.

2.4.4 Selection of Chromosomes

The selection operator is intended to improve the average quality of the population by giving the high-quality chromosomes a better chance to get copied into the next generation. The selection thereby focuses the exploration on promising regions in the solution space. Selection characterizes the selection schemes. It is defined as the ratio of the probability of selection of the best chromosome in the population to that of an average chromosome. Hence, a high selection pressure results in the population's reaching equilibrium very quickly, but it inevitably sacrifices genetic diversity (i.e., convergence to a suboptimal solution).

There are two basic types of selection scheme used commonly in current practice: proportionate and ordinal-based selection **[27,22]**. Both selection schemes suffer when the selection pressure is inadequate (low or high). Proportionate selection picks out chromosomes based on their fitness values relative to the fitness of the other chromosomes in the population. It is generally more sensitive to selection pressure. Hence, a scaling function is employed for redistributing the fitness range of the population in order to adapt to the selection pressure. Examples of such a selection type include roulette wheel selection, stochastic remainder selection, and stochastic universal selection. Ordinal-based selection schemes select chromosomes based not upon their fitness, but upon their rank within the population. The chromosomes are ranked according to their fitness values. It is noted that the selection pressure (intensity) is independent of the fitness distribution of the population, and is based solely on the relative ranking of the population. Since the selection pressure is the degree to which the better chromosomes are favored, it drives the GA toward improved population fitness over succeeding generations. However, it can become malicious when the selection pressure does not impose an adequate level. In other words, it may also suffer from high selection pressure.

Tournament selection, selection, truncation selection, and linear ranking selection schemes are included in the ordinal-based selection type. On the other hand, tournament selection without replacement is perceived as an effort to keep the selection noise as low as possible [27]. Recall that tournament selection without replacement works by means of choosing non-overlapping random sets of chromosomes (tournament size) from the population and then selecting the best chromosome from each set to serve as a parent for the next generation. Typically, the tournament size is two (pair wise tournament), and it would adjust the selection pressure: the selection pressure increases as the tournament size becomes larger [22]. Recall that the selection pressure is the expected average fitness of the population after selection. As selection pressure increases, the probability of making the wrong decision increases exponentially although the convergence of the GAs may be fast. Therefore, the pair wise tournament selection without replacement is employed for the proposed GA: two chromosomes are picked and the one that is fitter is selected. However, the same chromosome should not be picked twice as a parent.

2.4.5 Focusing Effect

This focusing effect allows GAs to quickly concentrate the search into favorable sub-dimensions of the overall problem, and results in a fairly rapid convergence to a reasonably good solution. Unfortunately, this solution will often not be the overall optimal solution. If the optimal solution is desired, this focusing effect should be reduced.

There are several factors that promote focusing. These include

- Using a small population size.
- Using a Mating Pool, or choosing both parents based upon their fitness.
- Not allowing for mutation.
- Not selecting both an offspring and its compliment.

- Immediately replacing the weakest member of the population with an offspring (unless it was one of the parents).

There are several factors that retard focusing. These include

- Using a large population size.
- Choosing one parent randomly.
- Allowing for significant mutations.
- Selecting both an offspring and its compliment.
- Allowing a member of the current population to mate before it is removed.

2.4.6 Steps for Genetic Algorithms

1. Determine a Coding Scheme, a Genetic Alphabet, and a Genetic Vector Form.
2. Determine the population size; i.e. the number of solutions you want to store.
3. Create an initial population. This can be randomly generated solutions, with or without local optimization, or the results obtained from a different search.
4. Choose how you want to select the parents.
    a. Choose one parent randomly and one based upon its fitness.
    b. Choose both parents based upon their fitness.
    c. Place members of the population in a Mating Pool in numbers proportional to thier fitness and randomly select from this pool.
    d. Choose one parent randomly, or biased by its fitness, and the other by choosing the solution that is most similar to it.
5. Determine a Mating Operator and the probability of mating (P_{mate})
6. Choose a Mutation Operator and a probability of mutation (P_{mutate}). In GAs, the Mutation Operator has a minor role, and P_{mutate} is often set to 0. Otherwise, the operator generally changes a value in the Genetic Vector by a small amount.
7. Determine whether or not you want to use a Maturation Operator. Though this is not discussed much in the literature, I have found that the results of a search can be greatly improved if the offspring (and initial population if necessary) are

subjected to an optimization, or some other check, before their fitness is determined. Possible operators include

   a. requiring an offspring to be unique.
   b. a local optimization.
   c. a Simulated Annealing or other non-local search.

8. Determine which offspring should be kept. Possible choices include
   a. use either a single mating or multiple matings for each pari of parents chosen.
   b. generate only a single offspring from each mating or the complimentary pair of offspring.
   c. keep only the best offspring from the generated set, or the best n offspring, or keep the best offspring and its compliment.
9. Determine what you want ot do with the new offspring. Some choices include
   a. placing the new offspring in a new population. When the new population has reached a given size (usually the size of the initial population) you can
      i. replace the current population with the new population, with or without copying the best solution from the current population to the new one. Keeping the best solution is known as the elitist strategy.
      ii. Deterministically choose the best solutions from the combined current + new population.
      iii. Probabilistically choose solutions from the combined current + new population.

These are called generational algorithms since a new offspring cannot be used as a parent until the new offspring population is large enough to consider some or all of it as the new parent population (e.g. a new generation of parents).

   b. Replace members of the current population with the chosen offspring. Possible replacements include
      i. Replace the weakest member in the population.
      ii. Replace the weaker parent.

iii. Replace the parent that is most similar to the offspring.

iv. Replace a member using a combination of lack of fitness and similarity to the offspring.

These are non-generational algorithms since each new offspring can be considered as parents in the next mating.

2.4.7 Advantages of Using Genetic Algorithms

- Discontinuities present on the response surface have little effect on overall optimization performance.
- They are resistant to becoming trapped in local optima.
- They perform very well for large-scale optimization problems.
- Can be employed for a wide variety of optimization problems.

2.5 Multi Objective Genetic Algorithms

Many real world problems require simultaneous optimization of multiple objectives, which is quite different from the single objective optimization. However, the single best solution may not exist with respect to all objectives under consideration. Instead, there might exist a set of solutions superior to the rest in the entire search space. Such a solution set is termed as "Pareto-optimal"[32]. Since none of the solutions in this set is absolutely better than any other, the user is given the freedom to choose the best possible solution that conforms to the application specific requirements. A Pareto-optimal solution is defined as follows:

Definition[32]: A point x is Pareto-optimal if for every x either $\cap_i (f_i(x) = f_i(x^*))$ or there is at least one i such that $(f_i(x) > f_i(x^*))$, $\forall i \in I$ (set of integers), where $f_i(x)$ is the fitness function. In other words, x^* is Pareto-optimal if there exists no feasible vector x which would decrease some criterion without causing a simultaneous increase in at least one other criterion.

A common difficulty with multi-objective optimization problem is the conflict between the objectives. In general, none of the feasible solutions allow simultaneous optimal solutions for all objectives. Thus, mathematically the most favorable Pareto-optimum is the solution that offers the least objective conflict. In order to find such solutions, classical methods scalarize the objective vector into one objective. The simplest of all classical techniques is the weighted sum method. It aggregates the objectives into a single and parameterized objective through a linear combination of the objectives. However, setting up an appropriate weight vector also depends on the scaling of each objective function. It is likely that different objectives take different orders of magnitude. When such objectives are weighted to form a composite objective function, it would be better to scale them appropriately so that each has more or less the same order or magnitude. Moreover, the solution obtained through this strategy largely depends on the underlying weight vector.

2.5.1 Pareto-Based Approach

In order to overcome such difficulties, Pareto-based evolutionary optimization has become an alternative to classical techniques such as weighted sum method. This approach was first proposed by Goldberg in [21] and it explicitly uses Pareto dominance in order to determine the reproduction probability of each individual.

The idea behind Non-dominated Sorting Genetic Algorithm (NSGA) is that a ranking selection n method is used to emphasize good points and a niche method is used to maintain stable subpopulations of good points. It varies from simple genetic algorithm only in the way the selection operator works. The crossover and mutation remain as usual. Before the selection is performed, the population is ranked on the basis of an individual's non domination. The non dominated individuals present in the population are first identified from the current population. Then, all these individuals are assumed to constitute the first non dominated front in the population and assigned a large dummy fitness value. The same fitness value is assigned to give an equal reproductive potential to all these non dominated individuals. In order to maintain the diversity in the population, these classified individuals are then shared with their dummy fitness values. Sharing is achieved by performing selection operation using degraded fitness values obtained by dividing the original fitness value of an individual by a quantity proportional to the number of individuals around it. Thereafter, the population is reproduced according to the dummy fitness values. Since individuals in the first front have the maximum .fitness value, they get more copies than the rest of the population. The efficiency of NSGA lies in the way multiple objectives are reduced to a dummy fitness function using non dominated sorting procedures.

Niched-Pareto Genetic Algorithm (NPGA) uses tournament selection based on non dominance. In the original proposal of the NPGA, the idea was to use a sample of the population to determine who is the winner between two candidate solutions to be selected, and to choose one of them based on nondominance with respect to the sample taken. To adapt the NPGA to solve single-objective parameter called selection ratio (S_r), which indicates the minimum number of individuals that will be selected through dominance-based tournament selection. The remainder will be selected using a purely probabilistic approach. In other words, (S_r – 1) individuals in the population are probabilistically selected. Each candidate has a probability of 50% of being selected.

When comparing two individuals, we can have three possible situations:

1. Both are feasible. In this case, the individual with a better fitness value wins.

2. One is infeasible, and the other is feasible. The feasible individual wins, regardless of its fitness function value.

3. Both are infeasible. The individual with the lowest amount of constraint violation wins, regardless of its fitness function value.

Our approach does not require niching or any other approach to keep diversity, since the value of S_r will control the diversity of the population. For the experiments a value close to one (>0.8) was adopted [31].

The pseudocode is presented where *oldpop* is the current population, t_{dom} is the size of the comparison set and flip is a function that returns TRUE with probability P:

```
function select
begin
    candidate_1 = tournlist[0];
    candidate_2 = tournlist[1];
    i f (flip(Sr)) /* fitness-feasibility-nondominance based tournament */
    begin
        i f (oldpop[candidate_1]=feasible AND oldpop[candidate_2]=feasible)
        i f (oldpop[candidate_1].fitness >= oldpop[candidate_2].fitness)
            winner=candidate_1; /* fitness checking */
        else
            winner=candidate_2;
        else /* feasibility checking */
        i f (oldpop[candidate_1]=feasible AND oldpop[candidate_2]=nonfeasible)
            winner=candidate_1;
        else
        if (oldpop[candidate_1]=nonfeasible AND oldpop[candidate_2]=feasible)
            winner=candidate_2;
        else
            begin
                shuffle(tournlist); /* nondominance checking */
                candidate_1_dominated = FALSE;
                candidate_2_dominated = FALSE;
                for (i=2 to tdom+2)
                begin
                    comparison_individual=tournlist[i];
                    if          (oldpop[comparison_individual]          dominates
                oldpop[candidate_1])
                        candidate_1_dominated=TRUE;
                    if          (oldpop[comparison_individual]          dominates
                oldpop[candidate_2])
                        candidate_2_dominated=TRUE;
                end
                i           f           (candidate_1_dominated=TRUE             AND
            candidate_2_dominated=FALSE)
                    winner=candidate_2;
                else
                i           f           (candidate_1_dominated=FALSE            AND
            candidate_2_dominated=TRUE)
                    winner=candidate_1;
                else /* tie        break with accumulated constraint violation */
                i f (oldpop[candidate_1].sumviol < oldpop[candidate_2].sumviol)
                    winner=candidate_1;
                else
                    winner=candidate_2;
            end
    end
    else
    i f (flip(0.5)) /* pure probabilistic selection */
        winner=candidate_1;
    else
        winner=candidate_2;
    return(winner);
end
```

Pareto optimal solutions are also termed non-inferior, admissible, or efficient ones. Their corresponding vectors are termed Nondominated [13]. The Nondominated vectors plotted in objective space are known as the Pareto front. Conventional optimization techniques, such as gradient-based and simplex-based methods are difficult to extend to the multiobjective case. GAs have been recognized to be well-suited to multiobjective optimization, because many individuals can search for multiple good solutions in parallel,

eventually taking advantage of gene similarities available in the family of possible solutions to the related problem. The ability to handle complex problems, involving features such as discontinuities and disjointed feasible spaces reinforces the potential effectiveness of GAs in multiobjective optimization. An approach to Multi-objective genetic algorithm (MOGA) varies from ordinary GAs only in its selection operator. Before performing the selection, the population is ranked on the basis of individual's chromosomes non-domination. A set of non-dominated strings are those which are better than others in the current population, when all the objective parameters are considered. All such non-dominated strings are included to form the initial Pareto-optimal front [32]. As MOGA iterated in every generation, the non-dominated , Pareto-optimal solutions are found and genetic operations are performed on them to improve their fitness values. The non-dominated solution set quickly proceeds towards the global optimal and gets saturated at a near optimal point. However, the procedue for obtaining non-domination between a pair of strings might result in a tie (both are either dominated or non-dominated). The concept of Niche Sharing [31] can be used to resolve such ties. Given an optimization function has several; peaks, the goal of fitness sharing is to distribute the population over the different peaks in the search space, where each peaks receives a fraction of the entire population according to its height. One practical way to achieve such fitness-sharing is to degrade an individuals strings fitness, by dividing it by niche count for that individual. The intuition behind the niche count is that it is a good estimate about the crowd of the neighborhood of a particular individual .

2.6 Mathematical Formulation of QoS Multicast Routing Model

The objective of QMR is to find the optimal path, which starts from the source node and passes through all destination nodes, that meets all QoS constraints with minimum cost or reaches a specific service level in a distributed network. For convenience, the network is considered as a connected, undirected and weighted graph.

Let N(V,E) represents a network, where V denotes the set of network nodes and E denotes the set of bi-directional links,

s Є V, is the source node in multicast group.

$M \subseteq \{V - \{s\}\}$, is the set of destination nodes in multicast group.

R_+, denotes the set of positive real numbers and

R^+ denotes the set of non-negative real numbers.

Assume that there are 4 QoS measures associated with each edge: network delay refers to the average time an IP packet needs to be transmitted through the network; network jitter refers to the range of time an IP packet need to be transmitted through the network; network bandwidth is the determinative factor that reduces the end-to-end delay. Thus, for any link eЄE, we denote:

$$\left.\begin{array}{l}\text{Delay Function, delay (e):} E \rightarrow R_+ \\ \text{Delay jitter Function, delay\_jitter (e):} E \rightarrow R^+, \\ \text{Bandwidth Function, bandwidth (e): } E \rightarrow R_+ \\ \text{Cost Function, cost (e):} E \rightarrow R_+.\end{array}\right\} \quad - \quad - \quad (2.6)$$

Similarly, fore each node nЄV, the four measures can be denoted as:

$$\left.\begin{array}{l}\text{Delay Function, delay (n): } V \rightarrow R_+, \\ \text{Packet Loss-rate Function, Packet\_loss(n): } V \rightarrow R^+, \\ \text{Cost Function, cost (n): } V \rightarrow R_+,\end{array}\right\} \quad - \quad - \quad (2.7)$$

Delay jitter Function, delay_jitter(n): $V \rightarrow R^{+}$.

IP packet may be damaged or lost in the transmission, if the loss rate is too high, obviously the data will be damaged.

Given a source node s€V and a set of destinations, M, the following relationships exist in the multicast tree T(s, M) composed of s and M.

$$\text{Delay}(P_T(s,t)) = \sum_{e \in pT(s,t)} \text{delay}(e) + \sum_{n \in pT(s,t)} \text{delay}(n) \quad - \quad - \quad - \qquad (2.8)$$

$$\text{Cost}(T(s,M)) = \sum_{e \in pT(s,t)} \text{cost}(e) + \sum_{n \in pT(s,t)} \text{cost}(n) \quad - \quad - \quad - \qquad (2.9)$$

$$\text{Bandwidth}(P_T(s,t)) = \min\{\text{bandwidth}(e), e \in PT(s,t)\} \quad - \quad - \quad - \qquad (2.10)$$

$$\text{Delay\_jitter}(P_T(s,t)) = \sum_{e \in pT(s,t)} \text{delay\_jitter}(e) + \sum_{n \in pT(s,t)} \text{delay\_jitter}(n) \quad - \qquad (2.11)$$

$$\text{Packet\_loss}(P_T(s,t)) = 1 - \prod_{n \in pT(s,t)} (1 - \text{packet\_loss}(n)) \quad - \quad - \qquad (2.12)$$

Where $P_T(s,t)$ is the routing path from source s to destination t in the multicast tree T(s,M). The objective of the QoS multicast routing problem is to find a multicast tree T(s,M), which satisfies:

$$\left.\begin{array}{l} \text{Delay constraint: delay}(PT(s,t)) <= D_t \\ \text{Bandwidth constraint: bandwidth}(PT(s,t)) >= B \\ \text{Delay jitter constraint: delay\_jitter}(PT(s,t)) <= DJ_t \\ \text{Packet Loss-rate constraint: Packet\_loss}(PT(s,t)) <= PL_t \end{array}\right\} \quad - \quad - \qquad (2.13)$$

Such that Cost (T(s,M)) is minimized for all multicast trees that satisfy the above constraints. Where B is the Bandwidth constraint, D_t, DJ_t, PL_t are the Delay, Delay jitter and Packet loss-rate constraints of the destination node t respectively. In this model, all bandwidth of the multicast destination nodes are assumed to be identical, however, delay, delay jitter and packet loss rate constraints can be different.

2.7 Summary

With the increasing demand for real-time services in next generation communication networks, QoS based Multicast Routing offers significant challenges. This chapter gives the overview of the multicast routing and the significance of delay and delay-jitter its QoS constraints. The background of two Evolutionary Algorithms namely Ant and Genetic Algorithms are discussed. Ant Algorithms are simulated evolution algorithm based on the population and ant colony behavior. Genetic Algorithms are the optimization techniques based on the concept of natural selection and survival of the fittest. Multiobjective optimization techniques like Pareto-optimality are also discussed. This chapter is concluded by describing the mathematical formulation of QoS Multicast Routing Model.

Chapter 3

QOS MULTICAST ROUTING USING ANT ALGORITHM

3.1 Introduction

An Ant Colony based heuristic is presented to solve the QoS Constrained Multicast Routing problem. This Ant based algorithm considers the QoS metrics like delay and delay-jitter to find the multicast tree that minimizes the total cost.

3.2 Description of the Algorithm

Given the values of $(d_i, dj_i, pl_i.c_i)$ for all nodes, $(d_{ij}, dj_{ij}, b_{ij}, c_{ij})$ for all edges, and the value of the constraints D, DJ, B,PL, the procedure can be divided into following steps:

Step1) Initialize network nodes.

Set T: =0(t is a timer and can be omitted.). NC:=0 (NC is a loop counter);

Assign an initial value $\tau_{ij}(t) = c$ to the pheromone intensity of every edge (i,j) and $\Delta\tau_{ij} = 0$;

Put m ants to the source node.

Step 2) Check PL (packet loss rate) of all nodes, delete the edges linking those nodes that do not meet PL constraint.

Step 3) Check B (bandwidth) of all edges, delete those edges that do not satisfy the bandwidth requirements.

Step 4) Setup tabu table

```
Set s:=1
For k:=1 to m
Put the value of source node into tabuk(s)
```

Here tabu is used to save the nodes that were reached before t. tabuk(s) denotes the s-th node visited by the k-th ant in the current route and s is the index of tabu table.

```
Step 5) Repeat this step until the tabu is full.
        Set s:=s+1.
        For K:=1 to m
            Choose a node (or the next node) j according to the following probability:
```

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} \bullet [\eta_{ij}]^{\beta}}{\sum_{k \in allowed_k} [\tau_{ij}(t)]^{\alpha} \bullet [\eta_{ij}]^{\beta}}, & j \in allowed_k \\ 0 & ,\text{otherwise} \end{cases} \quad - \quad (3.1)$$

Compute the delay and delay jitter to reach node j, and compare the result with delay D and the delay jitter DJ. If the result exceeds the constraints, choose a new node; otherwise move the k-th ant to node j.

Put j into $tabu_k(s)$,where $\tau_{ij}(t)$ is the pheromone intensity of edge (i,j) at t, and α,β denote the information accumulated during the movement of ants and the different effects of factors in the path selection.

$Allowed_k=\{0,1,2...N-1\}-tabu_k(s)$ denotes the node that the k-th ant can select in the next step. When using ant algorithm to solve TSP (Traveling Salesman Problem), $\eta_{ij} = 1/d_{ij}$ where d_{ij} is the delay between the nodes i and j. However, in this study it is not the length of the path that will affect the probability, but the next node, the delay of edges linking them and the delay jitter. Thus, we set $\eta_{ij} = 1/(d_{ij} + dj_{ij})$

```
Step 6) Compute $\Delta\tau^{k}{}_{ij}$ and $\Delta\tau_{ij}$:
        For k:=1 to m
        For every edge(i,j)
        For k:=1 to m
```

$$\left.\begin{array}{l} \text{Set} \quad \Delta\tau_{ij}{}^{k} = \dfrac{Q}{L_k} if(i,j) \in tabu_k \\ \qquad\qquad =0, \quad \text{otherwise} \\ \text{Set } \Delta\tau_{ij} = \Delta\tau_{ij} + \Delta\tau_{ij}^{k} \end{array}\right\} \quad - \quad - \quad - \quad - \ (3.2)$$

$\Delta\tau_{ij}^{k}$ is the pheromone amount left by the k-th ant at edge(i,j) during the period from t to t+n. $\Delta\tau_{ij}$ is the sum of pheramone amount left at edge(I,j) in this loop. L_K is the total path length of the k-th ant . Here L_K is the summary of d and dj of all nodes and edges passed by the k-th ant through its path, and Q is a constant.

```
Step 7) Compute $\tau_{ij}(t+n)$ for every edge (i,j);
        Set $\tau_{ij}(t+n) = \rho * \tau_{ij}(t) + \Delta\tau_{ij}$
        t:=t+n; NC:=NC+1;
        Set $\Delta\tau_{ij} = 0$ to every edge (i,j);
```

Where parameter ρ must be set to a value less than 1 to avoid the infinite accumulation of pheromone.

```
Step 8) Check stop condition.
        If (NC<NCMAX) and (not develop state)
          Then
              Empty all tabu; go to step 2
          Else
              Output the minimum cost path until all nodes have been passed.
```

3.3 Summary

Ant algorithm is a simulated evolution algorithm based on population and ant colony behaviors in which information exchange and collaboration among units play an important role. The key features of the algorithm include distributed computation, positive feedback and constructive greedy heuristic. These features can help to avoid premature convergence and find a good solution in shorter period of time.

3.4 Dataflow Diagram of the Ant Algorithm

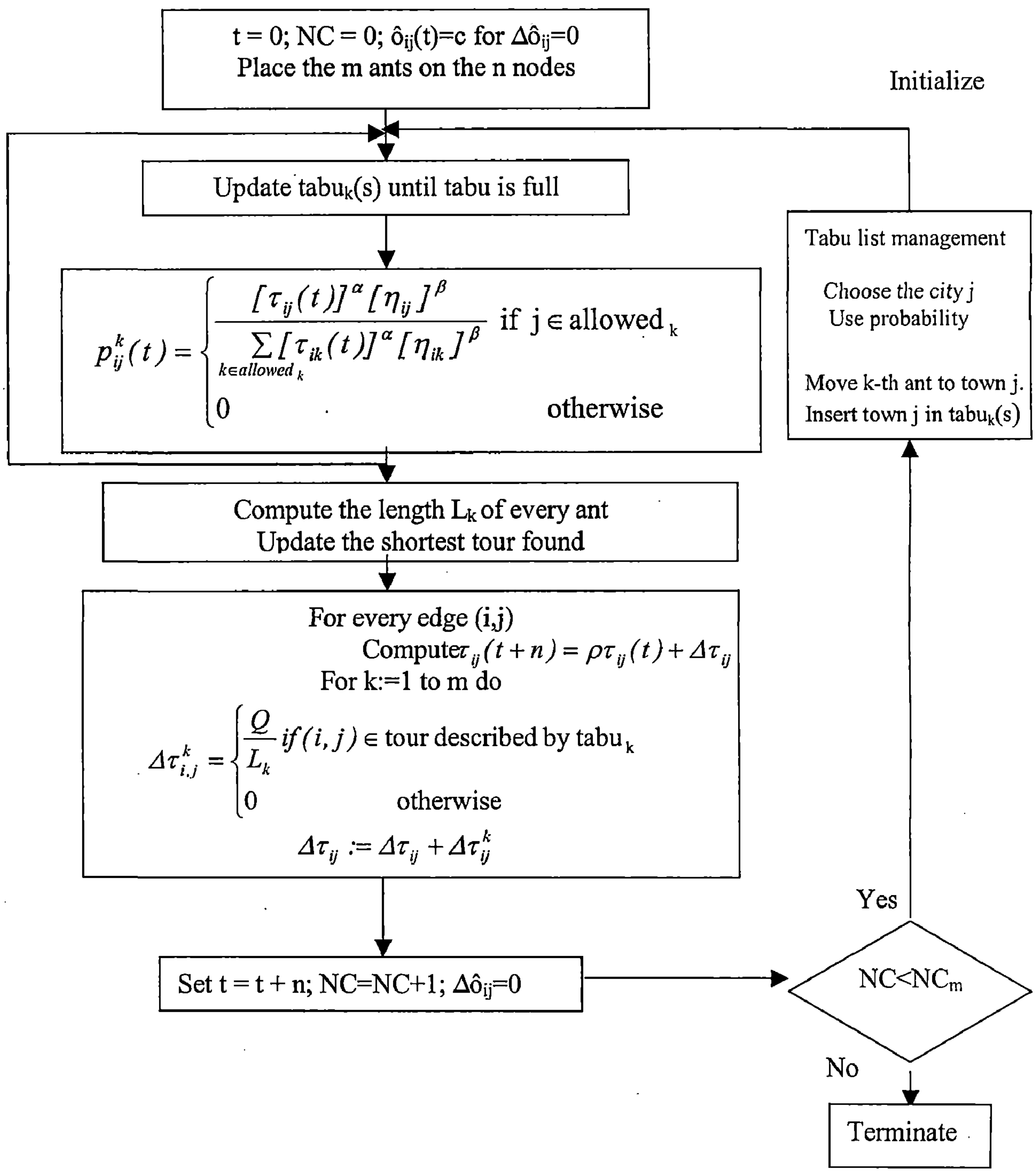
t = 0; NC = 0; ôij(t)=c for Δôij=0
Place the m ants on the n nodes
Initialize
Update tabuk(s) until tabu is full
$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{k \in allowed_k} [\tau_{ik}(t)]^\alpha [\eta_{ik}]^\beta} & \text{if } j \in \text{allowed}_k \\ 0 & \text{otherwise} \end{cases}$
Tabu list management
Choose the city j
Use probability
Move k-th ant to town j.
Insert town j in tabuk(s)
Compute the length Lk of every ant
Update the shortest tour found
For every edge (i,j)
Compute $\tau_{ij}(t+n) = \rho\tau_{ij}(t) + \Delta\tau_{ij}$
For k:=1 to m do
$\Delta\tau_{i,j}^k = \begin{cases} \frac{Q}{L_k} if(i,j) \in \text{tour described by tabu}_k \\ 0 \quad \text{otherwise} \end{cases}$
$\Delta\tau_{ij} := \Delta\tau_{ij} + \Delta\tau_{ij}^k$
Set t = t + n; NC=NC+1; Δôij=0
Yes
NC<NCm
No
Terminate

Fig 3.1 Ant Algorithm for Solving QoS Multicast Routing

Chapter 4

QOS MULTICAST ROUTING USING GENETIC ALGORITHM

4.1 Introduction

Genetic Algorithms are optimization techniques based on the concepts of natural selection and genetics. In this approach, variables are represented as genes on the chromosomes. The main operations performed in Genetic Algorithm are Selection, Cross-over and Mutation. Genetic Algorithms are well suited to solve the NP-Complete problems like QOS Multicast Routing. This algorithm finds the optimal multicast tree satisfying the QoS constraints like delay and delay-jitter.

4.2 Coding Scheme

To conform with the requirements of solving the QoS multicast routing problem using a Genetic Algorithm, the possible solutions of the problem needs to be mapped to a symbolic state space. The possible solutions form what is called a the population from which we try to select the best possible solutions after improving the overall quality of the solutions in the population using genetic operators like selection, cross-over and mutation. Since we are dealing with the multicast routing problem, a solution should actually depicts a tree with source as the root and the destinations as the leaves. Therefore the solution is encoded as follows:

First we list all the possible paths between the source and a particular destination and keep them in a pool. Note that a path is a sequence of network nodes. Now for the larger networks, there can be a huge number of such paths. However, we actually select those paths which conform to our bandwidth requirement. (ie., those paths having their capacities less than the required bandwidth are excluded). Still the number of possible paths could be large, especially for the large networks. These pools of valid paths between the source and the set of destinations are prepared.

After creating such pools we take one solution randomly from each pool and concatenate all the solutions to giver a multicast solution. The multicast solution string can be at-most of length c*n, where c is the number of destinations and n is the number of network nodes. In genetic algorithm terminology, the solution strings thus formed are called chromosomes and the components of the chromosomes are called genes.

1	2	7	3	4	-1	1	8	4	5	-1	1	2	7	6	-1	1	2	3	7	-1

Fig 4.1 Chromosome Coding Scheme

Consider the above figure 4.1, the above chromosome represents the multicast tree with the source 1 and the destinations 4,5,6,7. -1 represents a marker to distinguish the paths between each pair of source and destination. The initial population is selected from these pools by randomly picking up path for each set of destination

4.3 Fitness Function

Fitness function should describe the performance f the selected individuals. The individual with good performance has high fitness level, and the individual with bad performance level has low fitness level. The fitness function of the algorithm to calculate the QOS parameter like end-to-end delay ,delay-jitter, cost is $F(T)=f_c\ (Af_d + Bf_{dj} + Cf_{dj})$, where f_c ,f_d ,f_{dj} ,f_{dj} denotes cost, end-to-end delay , delay-jitter and bandwidth functions. A,B,C are the positive weight of f_d ,f_{dj} ,f_{dj} respectively. by means of iterative applications of GA operations, it is possible to find out the best representative solution from the entire population within a few iterations. The best solutions for multicast tree M are selected in each iteration with the help of an fitness function F. Several fitness functions can be designed from the different objectives. One choice is linear combination and the other is quadratic combination.

With the help of fitness function F , the quality of solutions is the improved by applying the genetic algorithm operations, namely selection, cross-over and mutation. These three operations are mainly string manipulation operations in which the string elements are changed or mixed so as to get a good variation of properties from each solution, and to preserve the good qualities of the solution for the next generation. This improvement ultimately converges with iterations resulting in optimal or near-optimal solution.

4.4 Selection Operation

Selection is mainly the operation to increase the number of good solution in the population in every iteration of the algorithm. The solutions are assigned a probability of being selected in the next generation which is proportional to its fitness function, F. In this way better solutions replaces the inferior ones in the next generation. The selection of the next generation solutions is done by the roulette wheel scheme once the probabilities of selection is given.

4.5 Cross-Over Operation

Cross-over operation mingles the genes or component of the solution to generate new representative solutions containing the properties of more then two solutions. Thus, this operation helps in getting out of localization problem which is common in an optimization procedure

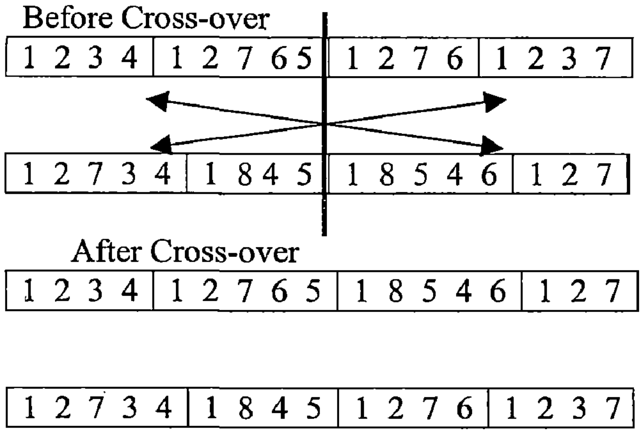

Fig 4.2 Cross-over operation between two chromosomes

. In our case the multicast trees are represented as concatenation of paths between the source and the destinations. We pick two solutions randomly from the population for cross-over operation. The first m genes or paths out of total c paths (m < c) of the first solution is concatenated with the last c-m genes of the second solution to get the new solution. The two remaining sections of the original solutions are again concatenated to form another new solution. These two new solutions replaces the original solutions in the new population.

4.6 Mutation

Mutation changes or mutates gene in a chromosome to introduce a new solution in the solution space. This operation also has the similar role in the optimization procedures that of the cross-over operation. Here we pick one solution from the population for mutation operation. One of the constituting genes is chosen randomly from the selected solution and is replaces with the other gene or path between the source and destination as it was in the original gene. To illustrate the operation , a chromosome such as [1 2 3 4 -1 1 2 7 6 5 -1 1 2 7 6 -1 1 2 3 7] can undergo the mutation operation with a certain low probability and yield another chromosome such as [1 2 3 4 -1 1 2 3 4 5 -1 1 2 7 6 -1 1 2 3 7]. In this operation the path between node 1 and 5 (ie., 1 2 7 6 5) is changes with another alternative path (ie., 1 2 3 4 5).

Successive application of these operations in each iteration make it possible to get a near-optimal solution within few iterations, even for larger networks.

4.7 Summary

Genetic Algorithm acquires the solution by representing a multicast tree as a chromosome so as to save the coding spaces and reduce the decoding operations (compared with binary coding mechanisms). The optimal multicast tree is acquired in successive iterations by applying the Genetic operations like Selection, Cross-over ad Mutation.

4.8 Dataflow Diagram of Genetic Algorithm

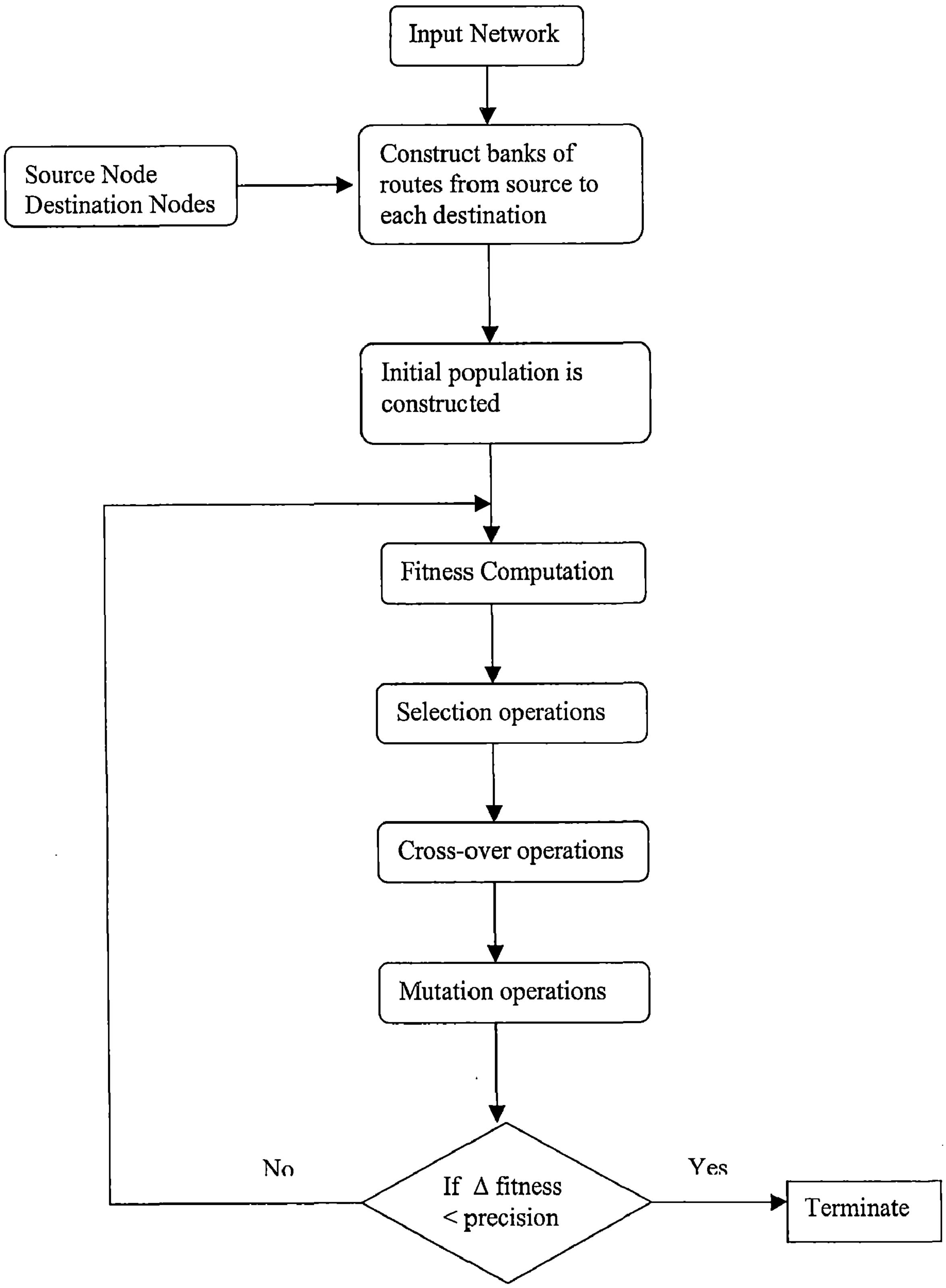

Fig 4.3 Genetic Algorithm for Solving QoS Multicast Routing

Chapter 5

QOS MULTICAST ROUTING USING MULTI-OBJECTIVE GA

5.1 Introduction

Multi-objective Genetic Algorithm differs from the simple Genetic Algorithm only in its Selection operation. The underlying novelty in the Multi-objective algorithm is that it does not combine the constraint functions on an ad hoc basis to form a scalar objective function, but attempts to tackle the problem from the perspectives of multi-objective optimizations. The motivation behind developing such an algorithm is to provide the user with a set of Pareto-optimal solutions, and give him the liberty to choose the best one, depending on his own requirements (if any), from this set.

5.2 Description of the Algorithm

The algorithm consists of following steps:

The Network-generation part of the algorithm is quite simple. It takes the number of nodes as input and generates the graph dynamically with random connectivity to represent the network.

The major part of the algorithm is QoS based multicast route discovery. This function takes the source node V_S and a specific number of multicast destination nodes, say, V_{d1} V_{d2} V_{d3} V_{d4} ....V_{dn} as input.

It calls the function *path finding* to find all possible multicast paths from V_s to each of V_{d1} V_{d2} V_{d3} V_{d4} ....V_{dn}, using the basic depth first search(dfs) algorithm. This gives birth to the initial set of multicast trees. The primary objective of our algorithm is to find the multicast trees, from this set, which will satisfy the three above-mentioned QoS parameters.

Since the underlying approach is based on multi-objective genetic algorithms(MOGA), our next step is to map the problem in a search space suitable to MOGA. Each of all the generated multicast trees is mapped to a string consisting of the sequence of nodes along the path from the source V_s to each of destinations V_{d1} V_{d2} V_{d3} V_{d4} ......V_{dn}. To mark the end of a path from a source to a single destination, we use -1 as sentinel. Figure 1 below gives a clear view of this scenario where a multicast tree is represented by a string. The set of all such strings constitute the initial population. The size of this population *popsize* depends on how the strings are created, which in turn depends on the network topology and the number of multicast destination nodes. The role of multi-objective-genetic algorithms now comes into the feature. The fitness computation() function computes the values of the three pre-defined QoS parameters individually. The objective of the algorithm now boils down to a search for different multicast paths which will improve the values of these QoS parameters at each iteration.

The *Niched Pareto Optimization* function comes next. The key idea to develop this approach is to use a ranking selection method to emphasize the good points and incorporate the concept of niching to maintain stable subpopulations of good points. The binary relation of domination leads to a binary tournament between any two randomly selected individuals. However, we wanted to get more domination pressure together with the control over that pressure. In order to achieve this goal, a comparison set of individuals are picked at random from the population.

The size of this comparison set t_{dom} gives us a good control over the selection pressure. If a small t_{dom} is chosen , only a few pareto optimal points will be found. Instead,choosing of a very large t_{dom} might result into a *premature convergence*. In this algorithm we have taken t_{dom} =0.20(popsize). Each of the two randomly selected individuals is now compared against each individual in the comparison set. If one candidate is dominated and the other is not then the latter is selected for selection. On the other hand, if both of the individuals are dominated or if both are non-dominated then we use the concept of *niched sharing* to resolve the tie.

To incorporate this idea of fitness sharing we compute the value of niche count for every individual string present in the population, as:

$$m_i = \sum_{j=1}^{popsize} S_h[d_{s1,s2}] \quad - \quad - \quad - \qquad (5.1)$$

where $d_{s1,s2}$ is he distance between individuals s1 and s2 and $S_h[d_{s1,s2}]$ is the sharing function. For simplicity, triangular sharing function has been used:

$$s_h[d_{s1,s2}] = 1 - \frac{d_{s1,s2}}{\sigma_{share}} \quad - \quad - \quad - \qquad (5.2)$$

for $d <= \sigma_{share}$ and S_h [d]=0 otherwise. Here σ_{share} is the niche radius, and it is a good estimate of minimal separation expected between the goal of solutions. Individuals within σ_{share} distance of each other degrade each other's fitness, as they are in the same niche.

A new idea of adaptive sharing is introduced, i.e. the value of σ_{share} is no longer kept fixed. Depending on the fitness values of the particular string chosen and the population density in the search space σ_{share} is dynamically updated in every iteration of the algorithm.

To implement this adaptive sharing, there are two choices:

(a) Genotypic Sharing, which is based on the distance between the individual genes of the chromosome and

(b) Phenotypic Sharing based on the distance between the fitness values of the entire chromosome.

Genotypic sharing can sometimes gives a good picture of the internal chromosomal structure, but results of recent researches had focused the superiority of the Phenotypic sharing over it. Moreover, the intuition and logic behind the distance between the corresponding genes(nodes) of the two chromosomes(strings) is also quite fuzzy in our case. This phenotypic distance between two strings is nothing but the Euclidian distance between their different fitness values:

$$\left.\begin{aligned}
d_{s1,s2} &= \sqrt{(\delta_{delay_{s1,s2}})^2 + (\delta_{bw_{s1,s2}})^2 + (\delta_{jitter_{s1,s2}})^2} \\
\text{where } \delta_{delay_{s1,s2}} &= \Pr\,(d_{s1} < t) - \Pr\,(d_{s2} < t)\,, \\
\delta_{bw_{s1,s2}} &= \Pr_{s1}(B) - \Pr_{s2}(B)\,, \\
\delta_{jitter_{s1,s2}} &= \Pr\,(dj_{s1} < t) - \Pr\,(dj_{s2} < t)
\end{aligned}\right\} \quad - \quad - \quad (5.3)$$

where $\Pr\,(d_{s1} < t)$ is the probability that the delay d_{s1} of the selected tree s1 will meet the specific delay constraint, is obtained by taking the product of delays over individual paths in that tree. $\Pr_{s1}(B)$ is the probability with which the bandwidth guarantee of B is satisfied for an entire multicast tree s1. $\Pr\,(dj_{s1} < t)$ is the probability that the delay-jitter dj_{s1} of the selected tree s1 will meet the specific delay-jitter constraint, is obtained by taking the product of delay-jitter over individual paths in that tree.

Similarly, We compute the niche radius σ_{share} as some fraction (precisely one-fourth) of the maximum separation possible in the population ie.,

$$\left.\begin{aligned}
\sigma_{share} &= \frac{\sqrt{(\delta_{delay_{MAX}})^2 + (\delta_{bw_{MAX}})^2 + (\delta_{jitter_{MAX}})^2}}{4} \\
\text{where } \delta_{delay_{MAX}} &= \Pr_{MAX}(d < t) - \Pr_{MIN}(d < t) \\
\delta_{bw_{MAX}} &= \Pr_{MAX}(B) - \Pr_{MIN}(B) \\
\delta_{jitter_{MAX}} &= \Pr_{MAX}(dj < t) - \Pr_{MIN}(dj < t)
\end{aligned}\right\} \quad - \quad - \quad (5.4)$$

As the algorithm executes, at every iteration the genetic operations dynamically update the chromosomes and try to improve the corresponding probabilities. Hence, the values of $\delta_{delay_{MAX}}$, $\delta_{bw_{MAX}}$, $\delta_{jitter_{MAX}}$ are updated, which in turn update σ_{share} to reflect the correct niche radius. A careful study into the niche sharing, concept discussed in [29] reveals that it is not necessary to decrease the fitness values of the two non-dominated chromosomes by dividing them by niche-count. We make the algorithm much simpler by just evaluating the niche counts and selecting the string having less niche count.

As the algorithm executes, at every iteration we get a set of non-dominated strings whose fitness values represent the Pareto-optimal solutions for that iteration. The niche sharing helps to converge the algorithm quickly by applying a dynamically controlled selection pressure. In fact, simulation results also demonstrate that the density of the Pareto-optimal solutions increase as the algorithm executes.

The cross over and mutation operations are same as normal genetic algorithms. But, we have to take care of the fact that these operations must not produce any illegal paths. A close look into the structure of the chromosome reveals that these genetic operations can not be performed on any arbitrary gene (network nodes), as that can gives birth to some paths which do not exist at all. Both the cross-over and mutation operations can only be performed at the end of an existing path, i.e. immediately after a particular sentinel, represented by -1. To give an equal probability to all such possible cross-over and mutation points, we randomly select one such point. The crossover operation is performed by swapping the portion of the two consecutive chromosomes after the particular selected point. In case of mutation we just replace the part of the chromosome after the mutation point by a corresponding part of any other valid chromosome.

5.3 Summary

Careful analysis of existing optimization schemes for QoS Multicast Routing Algorithms reveals that most of them suffer from the same drawback: multiple objectives are combined to form the single scalar objective function on an ad hoc basis, usually in a linear combination of multiple attributes. This not only makes solution highly sensitive to the chosen weight vectors but also demands the user to have some knowledge about the priority of a particular objective parameter. The users will therefore be more interested in obtaining the set of acceptable non-dominated solutions, one of more of which can be selected according to the QoS requirements. We recognize that Genetic Algorithms are readily modified to deal with multiple objectives by incorporating the concept of Pareto Non-dominance in its selection operation.

5.4 Dataflow Diagram of the Algorithm

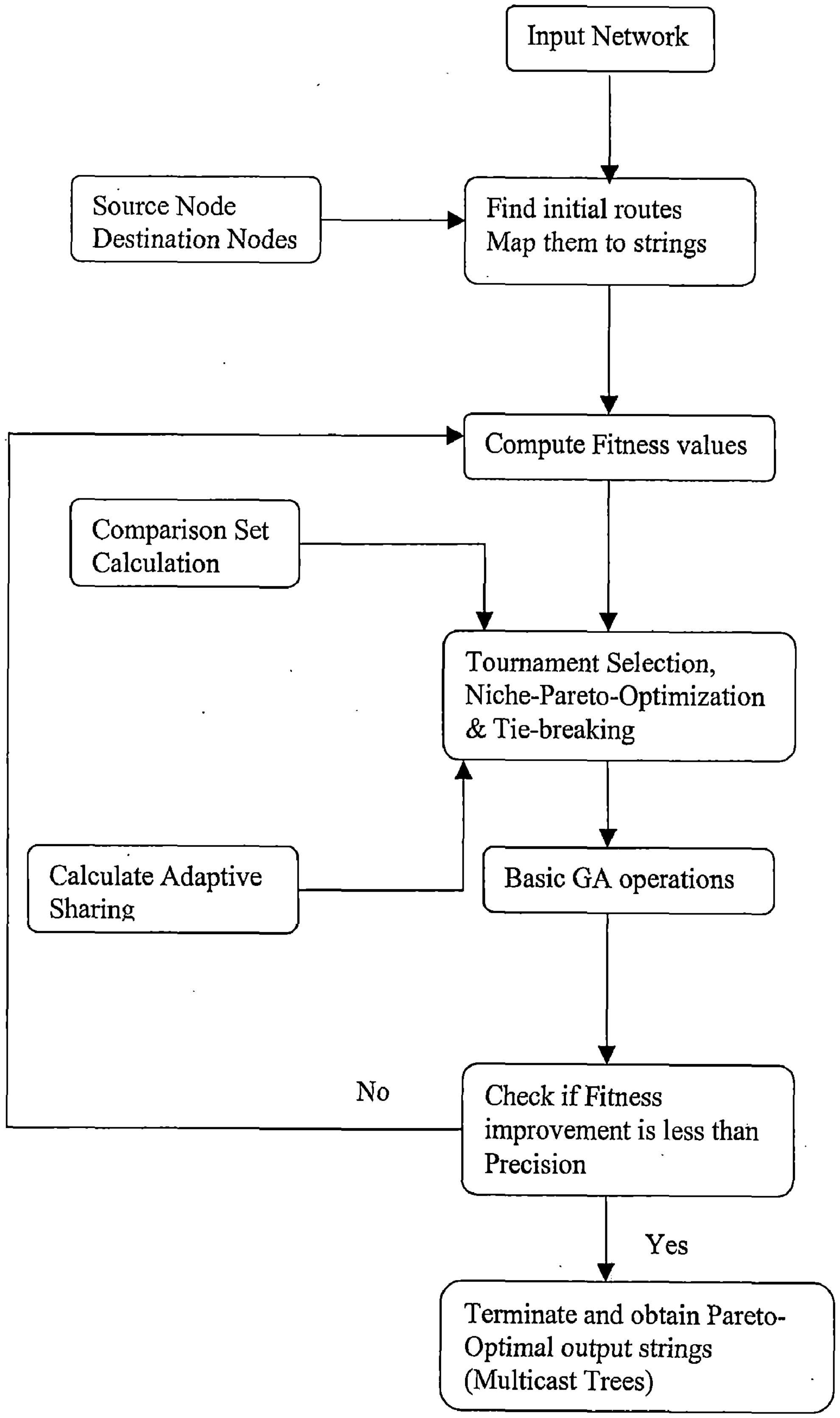

Fig 5.1 Multiobjective Genetic Algorithm for solving QoS Multicast Routing

Chapter 6

IMPLEMENTATION

The first phase of the simulation is generation of the network with QoS parameters. Network topologies used in the simulations are deliberately manipulated to simulate wide area sparse networks. A large network is likely to be loosely interconnected. An n-node graph is considered to be sparse when less than 5% of he possible edges are present in the graph. The network graph is constructed using the basic Waxman method for the generation of the edges with cost, delay, jitter, bandwidth constraints. These constraints are assumed to follow the exponential distribution. In the simulations, group size are always made less than 20% of the total nodes, because multicast applications running in a wide area network usually involve only a small number of nodes in the network, such as video conference systems, co-operative editing systems etc.

The next phase of the simulation is finding out the optimal multicast trees from the generated topologies using the Ant and Genetic algorithms for different multicast group sizes. In the network topologies generated, the nodes in the graph are labeled starting form 1.Group members are selected by the users input. Large numbers of simulations are performed with varying number of multicast group size and network size. The results of the simulations are compared with results obtained using the exhaustive search methods. The simulations are carried out using GNU C++ compiler in UNIX environment. The graphs for the results obtained during the simulations are generated using Xgraph utility.

Pheromone Trail intensity	$\alpha = 1.0$
Pheromone Trail Visibility	$\beta = 5.0$
Evaporation Coefficient	$\rho = 0.5$
Initial Pheromone Deposit	$\tau = 0.01$
Constant	Q=100
Number of Ants	M=number of nodes, N

Fig 6.1 Parameters for Ant Algorithm

Cross-over probability	$P_c = 0.8$
Mutation Probability	$P_m = 0.1$

Fig 6.2 Parameters for Genetic Algorithm

Cross-over probability	$P_c = 0.8$
Mutation Probability	$P_m = 0.1$
Initial Population	$P_0 = 10$
Size of Comparison Set	$t_{dom} = 0.20$(popsize)

Fig 6.3 Parameters for Multi-Objective Genetic Algorithm

Chapter 7

SIMULATION RESULTS

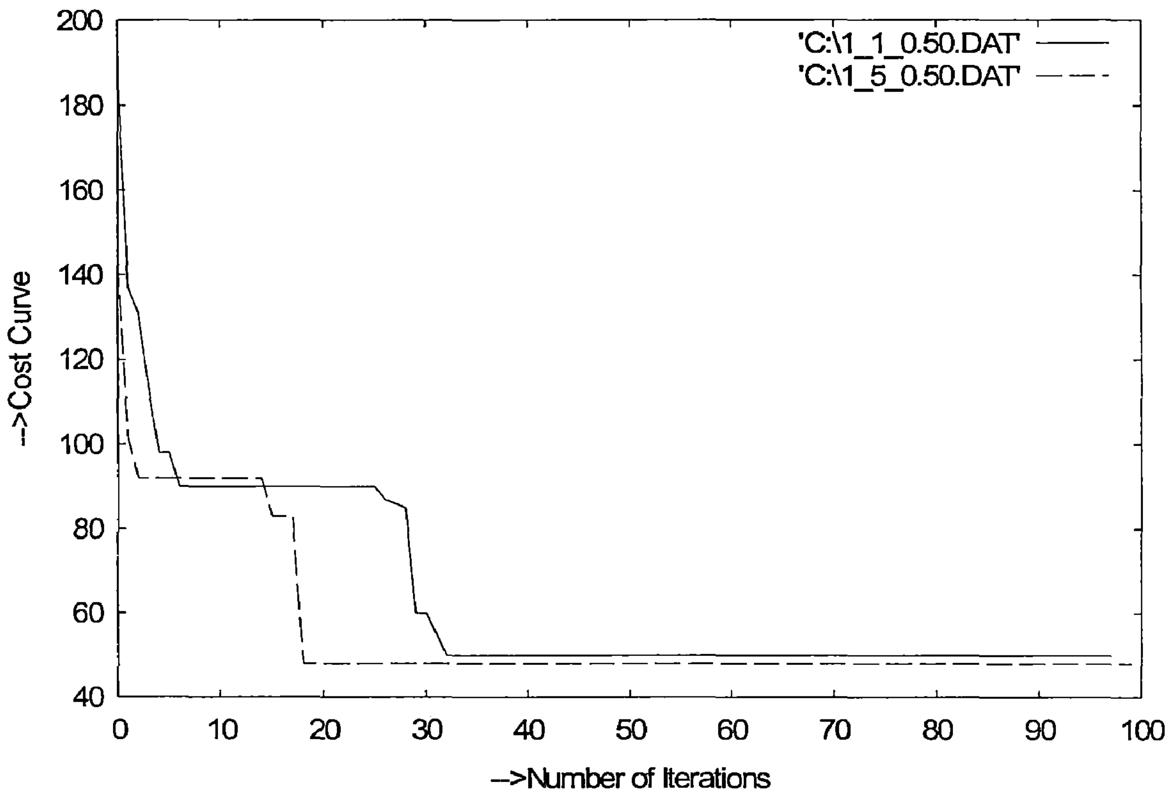

Fig 7.1 Performance Evaluation of the Parameters

In Fig7.1, the parameters of the Ant Algorithm are considered and their performance is checked. The dashed line denotes the cost curve with parameters α=1, β= 5, ρ=0.5. The straight line denotes the cost curve with parameters α=1, β= 1, ρ=0.5. From the graph, it can be clearly stated that the algorithm has good convergence with the parameter β= 5.

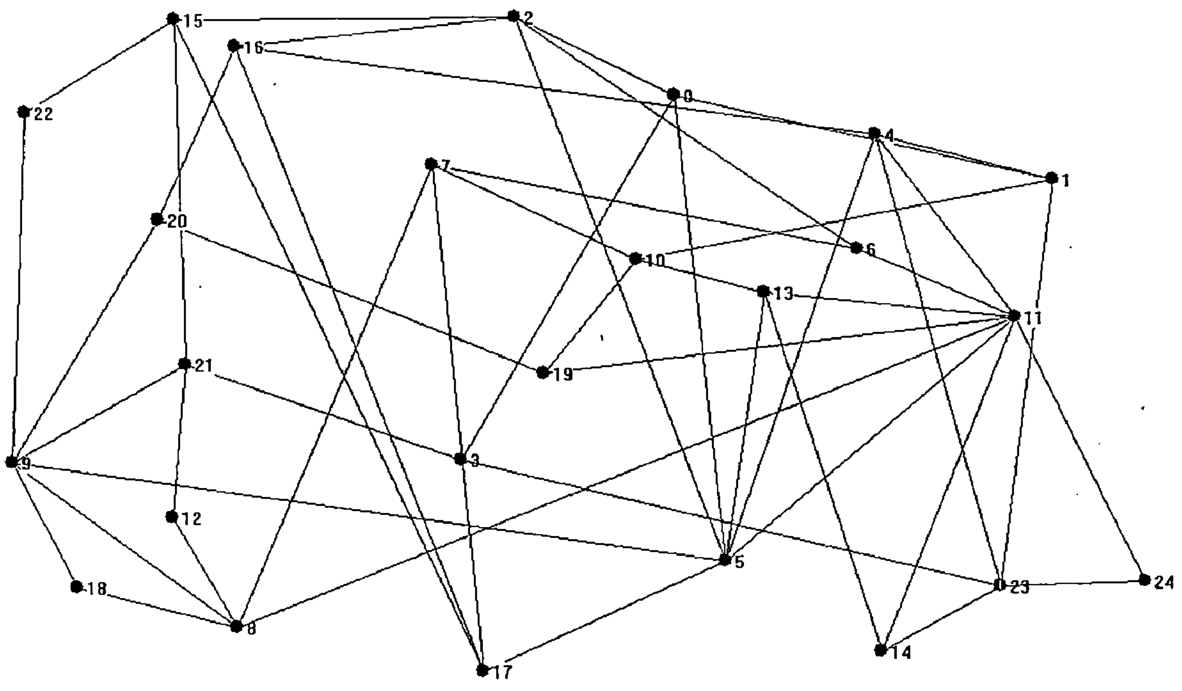

Fig 7.2 Network Graph

Fig 7.2 depicts the Sample Network Topology with 25 nodes. The network graph is constructed using the basic Waxman method for the generation of the edges with cost, delay, jitter, bandwidth constraints. Parameters used are α=0.26 and β=0.4 These constraints are assumed to follow the exponential distribution. In the simulations, group size are always made less than 20% of the total nodes, because multicast applications running in a wide area network usually involve only a small number of nodes in the network

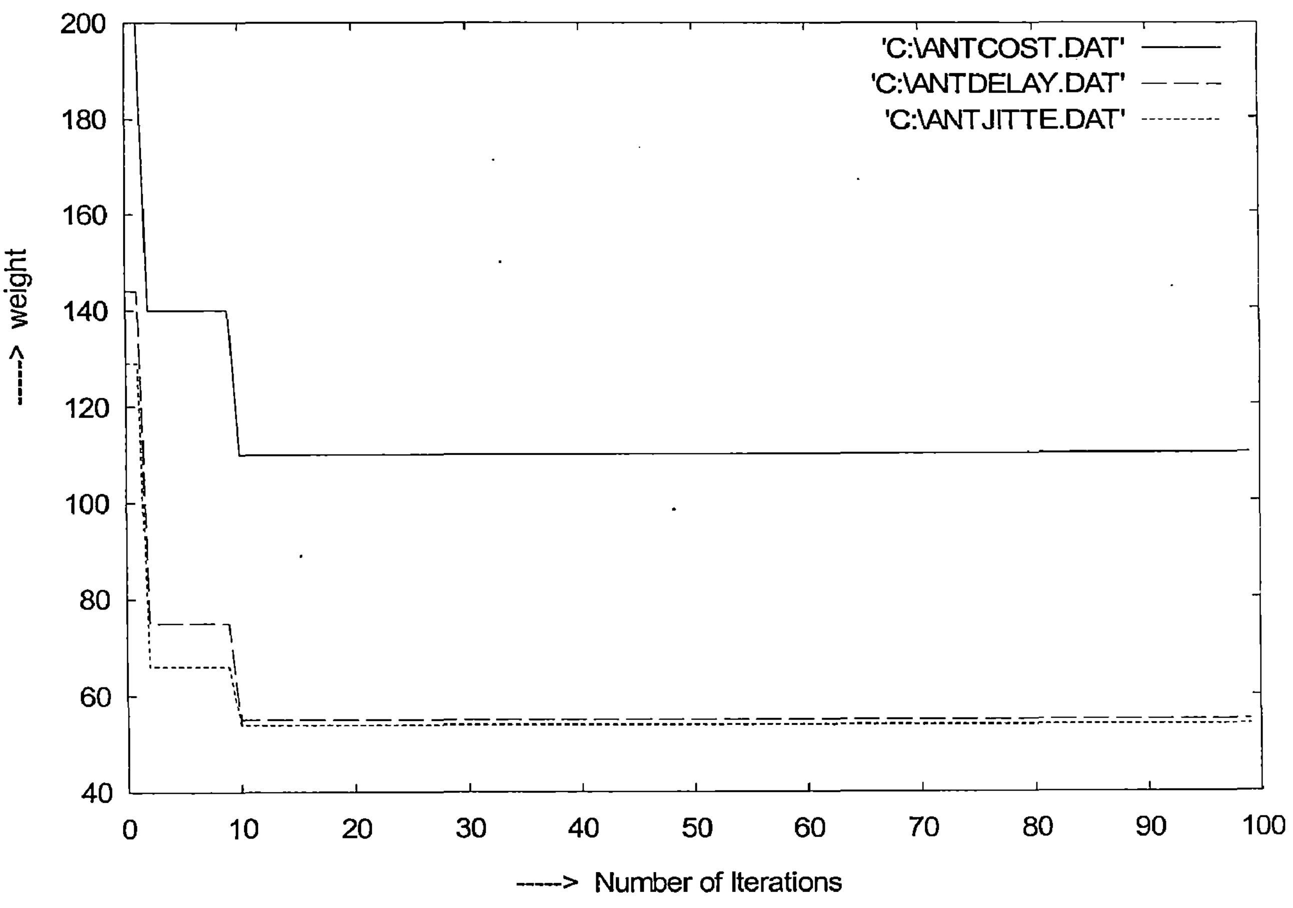

Fig 7.3 Convergence Graph Of Ant Algorithm

Fig 7.3 depicts the Convergence Graph of the Ant Algorithm. The straight line represents the cost of the multicast tree. The dash and dotted line represents the delay and delay-jitter of the multicast tree respectively. The Algorithm finds the optimal multicast tree with few iterations and the cost line becomes stable.

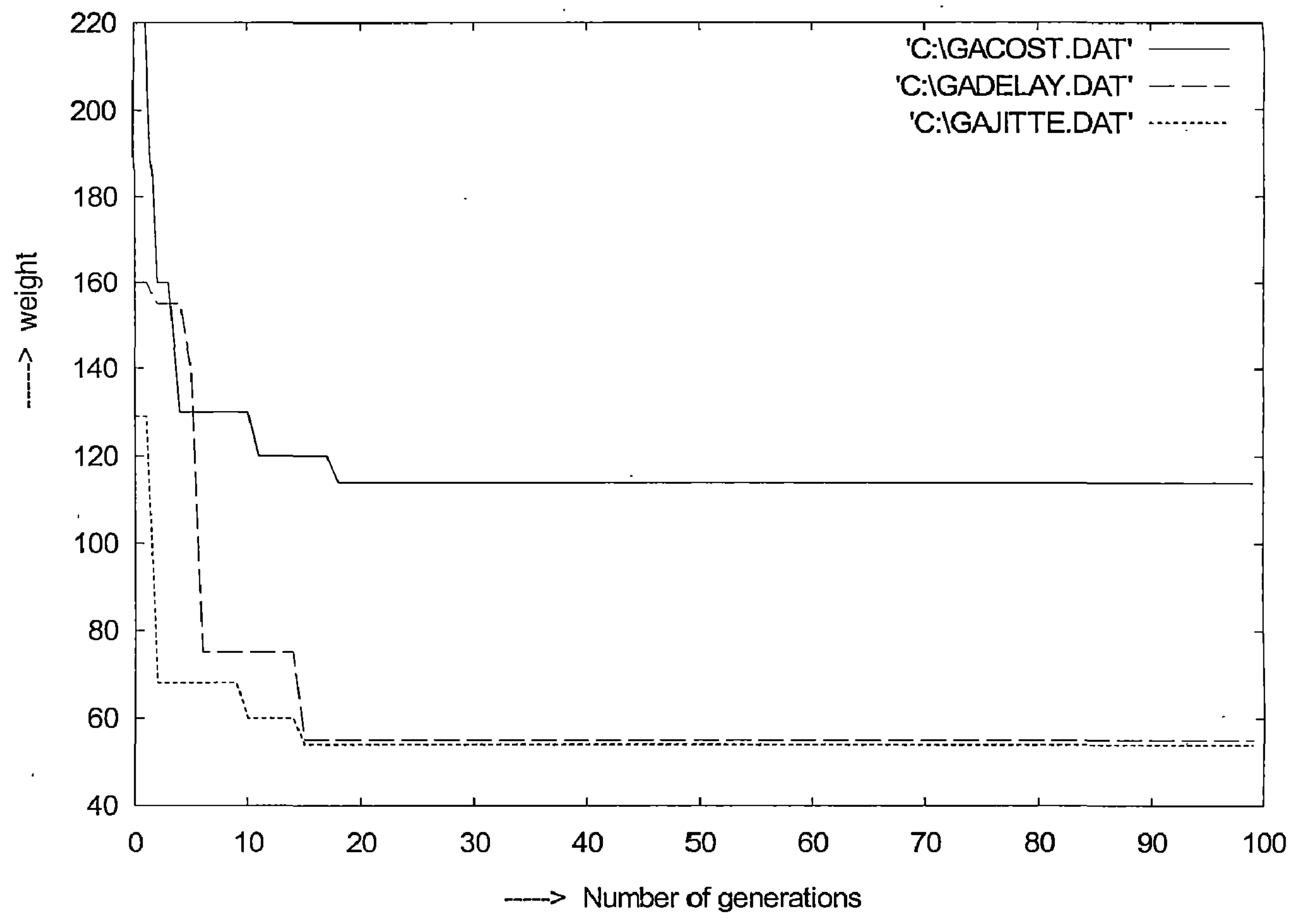

Fig 7.4 Convergence Graph Of Genetic Algorithm

Fig 7.4 depicts the Convergence Graph of the Genetic Algorithm. The straight line represents the cost of the multicast tree. The dash and dotted line represents the delay and delay-jitter of the multicast tree respectively. The Algorithm finds the optimal multicast tree within a few generations and the cost line becomes stable.

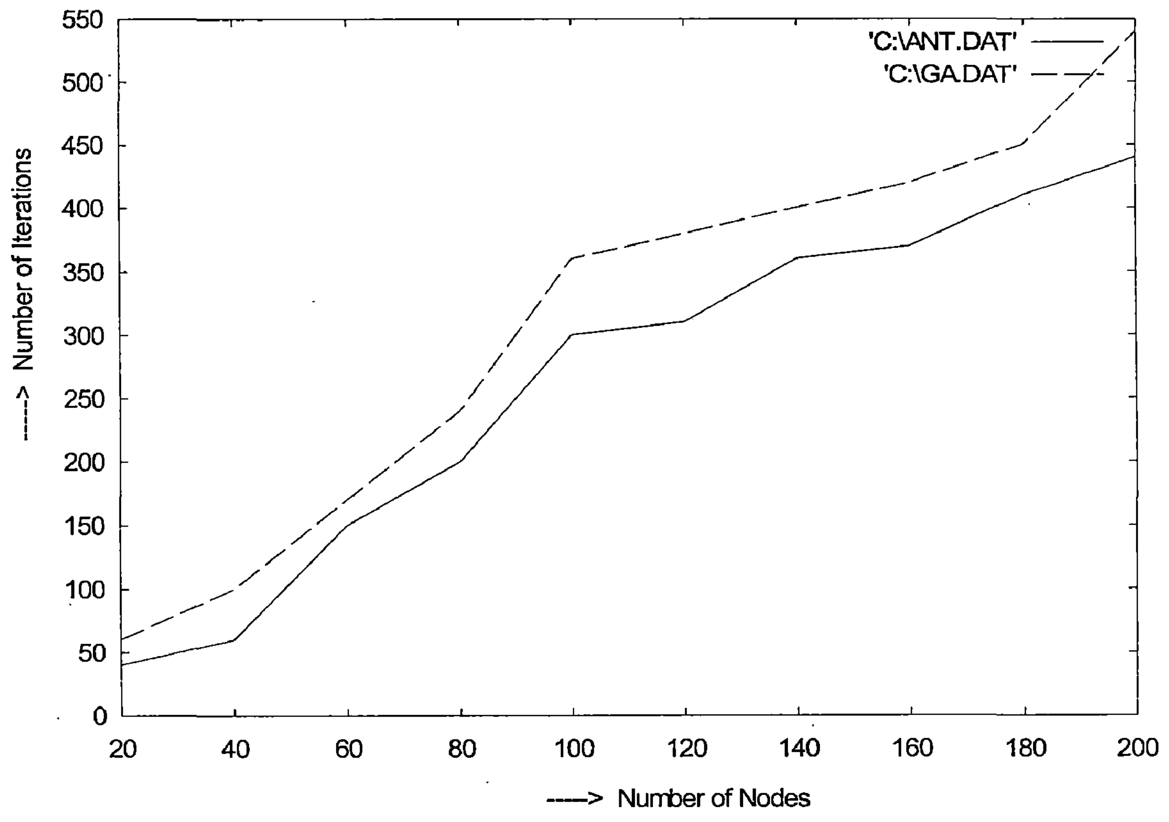

Fig 7.5. Comparison Graph for Ant and Genetic Algorithm

Fig 7.5 illustrates the performance of Ant and Genetic Algorithms. Simulations are performed with different multicast group sizes and the networks on these algorithms. From the graph, it is clear that Ant Algorithms take lesser number of iterations when compared to Genetic Algorithms to obtain the optimal or near optimal solutions. Ant algorithms show faster convergence towards optimal solution when compared to Genetic Algorithms.

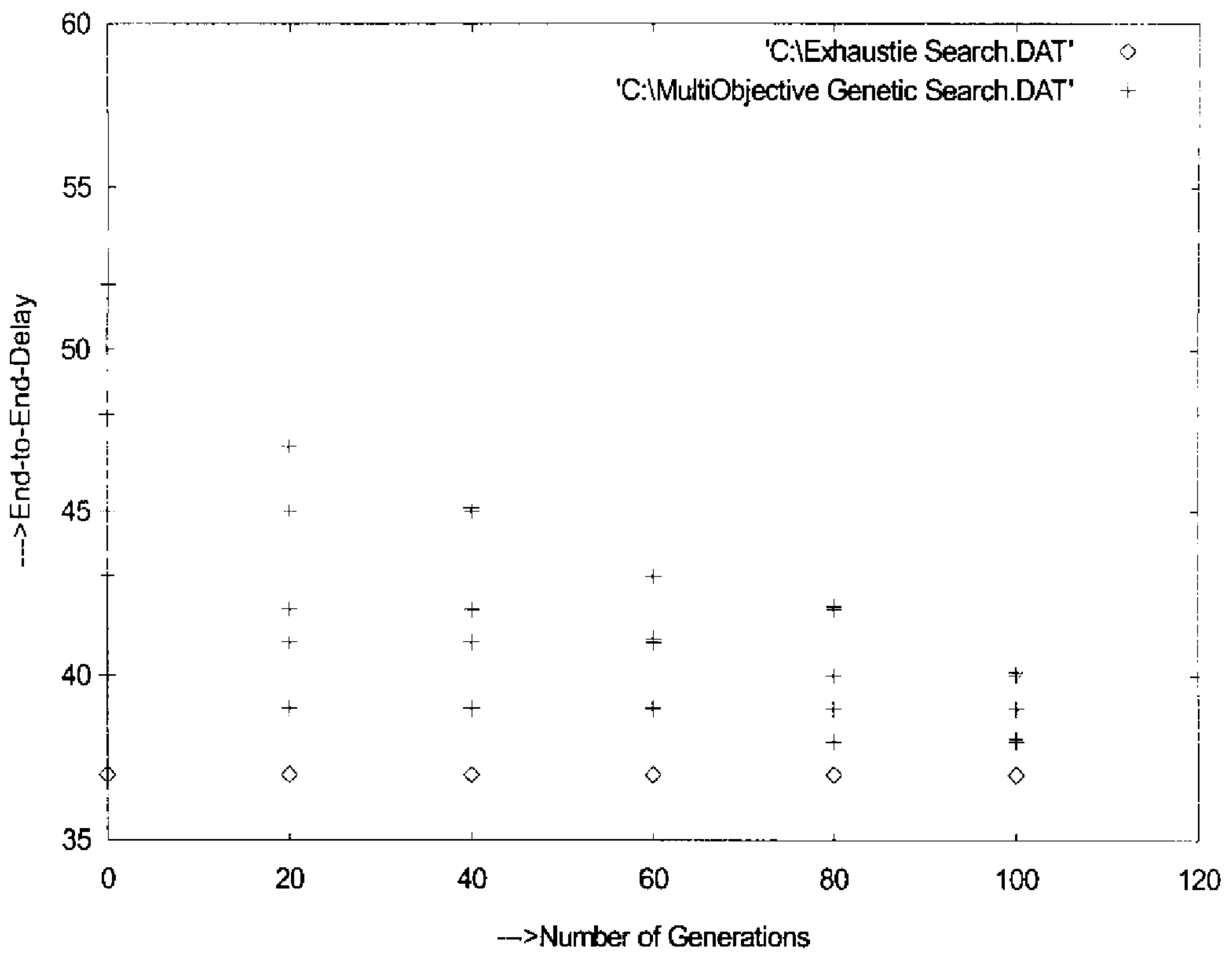

Fig 7.6 Convergence Graph of Pareto-optimal End-to-End Delay

Fig 7.6 explains the convergence of End-to-End Delay in Multi-Objective Genetic Algorithm. This vividly explains how the pareto-optimal fronts are developed in the Multi-objective Genetic Algorithm ,and proceeds towards global optima in a feasible time. The set of Non-dominated solutions converges to a narrow range and move towards the optimal solution

CENTRAL LIBRARY
Acc. No.........

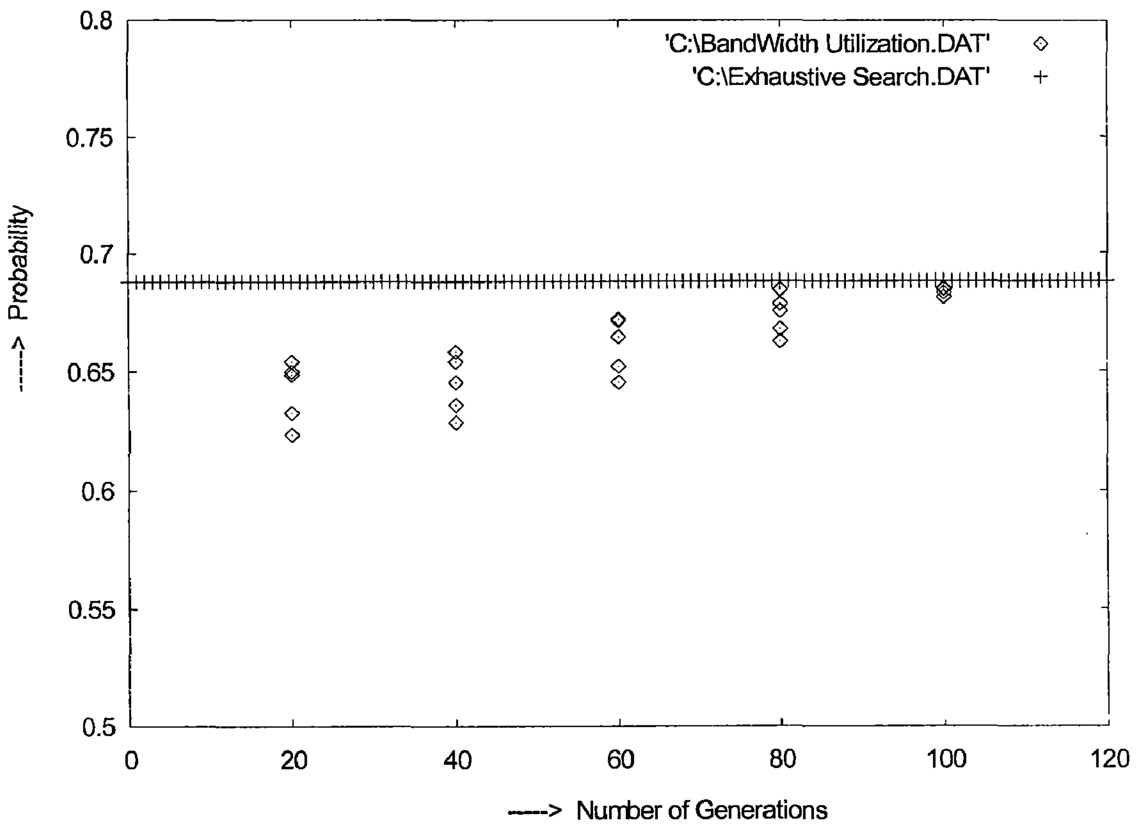

Fig 7.7 Convergence Graph of Pareto-Optimal Bandwidth Utilization

Fig 7.7 explains the convergence of pareto-optimal Bandwidth Utilization. This explains how the pareto-optimal fronts are developed in the Multi-objective Genetic Algorithm ,and proceeds towards global optima in a feasible time. The set of Non-dominated solutions converges to a narrow range and move towards the optimal solution

Chapter 8

CONCLUSIONS AND FUTURE WORK

In this thesis, the three heuristics are described for solving the QoS multicast routing problem. There are Ant Algorithm, Genetic Algorithm and Multi-objective Genetic Algorithm

The Ant algorithm described has the following characteristics: 1) the cost curve of this algorithm is somewhat stable, and the optimum or suboptimum can be found quickly Therefore, the algorithm is able to efficiently improve the transmission quality of the data packets in the network 2) the algorithm has good scalability. This is because the ant algorithm leaves the information on the path for the sake of next routing during the search process. The information is kept when more nodes are added. With this information the expanded network can be routed quickly. Applying Ant algorithm to solve the QoS multicast routing problem is a new research area. More extensive simulations are to be conducted on this algorithm in both wired and wireless networks. The relationship between the number of added nodes and the initial nodes should be explored.

Genetic Algorithms are well suited to Multiobjective optimization problems. Multiple parameters can be formulated as a Multiobjective model. In this thesis, instead of traditional bit string encoding of chromosomes, the path along the source to each of the destinations are considered and genetic operations are applied on the paths.

Careful analysis of existing optimization schemes for QoS Multicast Routing Algorithms reveals that most of them suffer from the same drawback: multiple objectives are combined to form the single scalar objective function on an ad hoc basis, usually in a linear combination of multiple attributes. This not only makes solution highly sensitive to the chosen weight vectors but also demands the user to have some knowledge about the priority of a particular objective parameter. The users will therefore be more interested in obtaining the set of acceptable non-dominated solutions, one of more of which can be selected according to the QoS requirements. We recognize that Genetic Algorithms are readily modified to deal with multiple objectives by incorporating the concept of pareto Non-dominance in its selection operation.

In this thesis, a Multiobjective model based on Genetic Algorithm is described to approximate Pareto front by generating a set of non-dominated solutions. These algorithms are source based fashion, and it assumes the complete knowledge of a network is available. Simulation results delineate the efficiency, performance and scalability of the algorithms. Researches in the QoS routings are mostly done to optimize the QoS parameters by combining their different. Conflicting characteristics into a single scalar function with the real intuition and logic behind the combination being often fuzzy. A future interest is to mathematically model this protocol to analyze its performance and complexity.

Since these heuristics are source based fashioned, these are well suited for the Integrated Services (INTSERV) QoS environment. However further research interests are in extending these algorithms for the Differentiated Services (DIFFSERV) environment.The Ant and Genetic Algorithms proved to be the significant research areas in the area of QoS multicasting, as these algorithms converge faster and can be used to find the efficient QoS multicast trees even for larger networks.

REFERENCES

[1] Bin Wang and Jennifer C. Hou, "Multicast routing and its QoS extension: Problems, algorithms, and protocols", IEEE Network, Jan/Feb, 2000, pp. 22-36.

[2] Roch A Guerin and A. Orda, "QoS routing in networks with inaccurate information: Theory and algorithms", IEEE/ACM. Trans. On Networking, No.3, Vol.7. June. 1999, pp. 350-363.

[3] Li Layuan, "A formal specification technique for communication protocol", Proc of IEEE INFOCOM, April. 1989, pp. 74-81.

[4] X. Jia, "A distributed algorithm of delay-bounded multicast routing for multimedia applications in wide area networks", IEEE/ACM Transactions on Networking, No.6, Vol.6, Dec. 1998, pp. 828-837.

[5] Li Layuan and Li Chunlin, "The QoS-based routing algorithms for high- speed networks", Proc of WCC, Aug. 2000, pp. 623-1628.

[6] Dorigo M., "Optimization, learning and natural algorithms", Doctoral dissertation, Politecnico di Milano, Dipartimento di Elettronica, Italy, 1992.

[7] Dorigo M., Maniezzo V. and Colorni A., "The Ant System: Optimization by a colony of cooperating agents", IEEE Transactions on Systems, Man, and Cybernetics, Part B, 1996, pp. 29–41.

[8] Corne D., Dorigo M. and Glover F., "New ideas in optimization", Maidenhead, UK: McGraw-Hill, 1999.

[9] Bonabeau E., Dorigo M. and Theraulaz G., "From natural to artificial swarm Intelligence", New York: Oxford University Press, 1999.

[10] Dorigo M. and Gambardella L. M., "Ant Colony System: A cooperative learning approach to the traveling salesman problem", IEEE Transactions on Evolutionary Computation, 1997, pp. 53–66.

[11] Gambardella L. M. and Dorigo M., "HAS-SOP: An hybrid ant system for the sequential ordering problem", (Tech. Rep. 11-97), Switzerland: IDSIA., 1997.

[12] St¨utzle T. and Hoos H., "The MAX °MIN Ant System. and local search for the traveling salesman problem", Proceedings of IEEE-ICEC-EPS '97, IEEE International Conference on Evolutionary Computation and Evolutionary Programming Conference Piscataway, NJ: IEEE Press. 1997, pp. 309–314.

[13] Di Caro G. and Dorigo M., "AntNet: A mobile agents approach to adaptive routing", Tech. Rep. 97-12, Universit Libre de Bruxelles, IRIDIA.,1997.

[14] G. Lu, Z. Liu and Z. Zhou, "Multicast Routing Based on Ant Algorithm for Delay-Bounded and Load-Balancing Traffic", 25th Annual IEEE Conference on Local Computer Networks, 2000, pp 362-368.

[15] G. Lu and Z. Liu, " Multicast Routing Based on Ant Algorithm with Delay and Delay Variation Constraints", 2000 IEEE Asia-pacific Conference on Circuits and Systems, 2000, pp 243-246.

[16] C.H. Chu, J.H. Gu, Xiang Dan Hou and Q.Gu, "A Heuristic Ant Algorithm for solving QoS Multicast Routing Problem", IEEE conference on Evolutionary Computation, 2002, pp 1630-1635.

[17] Colorni A., Dorigo M. and Maniezzo V., "Distributed optimization by ant colonies", Proceedings of the First European Conference on Artificial Life, Cambridge, MA, 1992, pp. 134–142.

[18] Di Caro G. And Dorigo M., "Ant colonies for Adaptive Routing in Packet-switched Communications Networks", Technical Report 98-34, Parallel Problem Solving from Nature, 1998.

[19] Dorigo M., Maniezzo V. and Colorni A., "Positive feedback as a search strategy", Tech. Rep. 91-016, Politecnico di Milano, Dipartimento di Elettronica, Italy, 1991.

[20] Dorigo M., Maniezzo V. and Colorni A., " The Ant System: Optimization by a colony of cooperating agents", IEEE Transactions on Systems, Man, and Cybernetics, Part B, 26 (1), 1996, pp. 29–41.

[21] D. E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning", Reading, MA: Addison-Wesley,1999.

[22] X. Hue, "Genetic algorithms for optimization: Background and applications," Technical Report, Edinburgh Parallel Computing Centre, Univ. Edinburgh, Dinburgh, Scotland, Ver 1.0, Feb. 1997.

[23] G. Syswerda, "Uniform crossover in genetic algorithms," Proc. Of 3rd Int. Conf. Genetic Algorithms, San Mateo, CA: Morgan Kaufmann, 1989, pp. 2–9.

[24] J. H. Holland, "Adaptation in Natural and Artificial Systems", Ann Arbor, MI: Univ. Michigan Press, 1975.

[25] K. A. DeJong, "An analysis of the behavior of a class of genetic adaptive systems", Ph.D. dissertation, Dept. Comput. Commun. Sci., Univ. Michigan, Ann Arbor, MI, 1975.

[26] D. E. Goldberg and M. Rundnick, "Genetic algorithms and the variance of fitness," Complex Syst., vol. 5, no. 3, 1991, pp. 265–278.

[27] D. E. Goldberg, K. Deb, and J. H. Clark, "Genetic algorithms, noise, and the sizing of populations", Complex Syst., vol. 6, 1992, pp. 333–362.

[28] C. A. C. Coello, "An Updated Survey of GA-Based Multiobjective Optimization Techniques", Technical Report, Laboratorio Nacional de Informatica Avanzada (Lania), Xalpa, Veracruz, Mexico, June 1998.

[29] K. Deb and D. E. Goldberg "An investigation of niches and species formation in genetic function optimization", Proceedings of the third International Conference on Genetic Algorithms, 1991, pp. 42-50.

[30] C. M. Fonesca and P. J. Fleming, "Genetic Algorithms for Multiobjective optimization: formulation, discussion and generalization", Proceedings of the Fifth International Conference on Genetic Algorithms, 1993, pp. 416-423.

[31] J. Horn, N. Nafpliotis and D. E. Goldberg, "A Niched Pareto Genetic Algorithm for Mutiobjective Optimization", Illinois Genetic Algorithms Laboratory Report No. 93005, University of Illinois at Urbana-Chamaign, New York, 1993.

[32] N. Srinivas and K. Deb, "Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms", Jour. of Evolutionary Computation, vol. 2, no. 3, pp. 221-248.

[33] G. Apostolopoulos and S. K. Tripathi, "On the effectiveness of path precomputation in reducing the processing cost of on-demand QoS path computation", Third IEEE Symposium on Computers and Communications, ISCC-98,.pp. 42-54.

[34] T. Back, J. M. Graaf, J. N. Kok and W. A. Kosters, "Theory of Genetic Algorithms",Technical Report, Leiden University, department of Computer Science, 1998.

[35] N. Banerjee and S. K. Das, "Fast Determination of QoS-based Multicast Routes in Wireless Networks using Genetic Algorithms", International Conference for Communication, ICC-2001, pp.48-68.

[36] L. Guo and I. Matta, "Search Space Reduction in QoS Routing", Proceedings of 19th IEEE International Conference on Distributed Computing Systems, 1999, pp. 142-149.

[37] D. H. Lorenz and A. Orda, "QoS Routing in networks with Uncertain Parameters: Theory and Algorithms", IEEE/ACM Transactions on Networking, vol. 6, no. 6, Dec. 1998, pp. 768-778.

[38] F. Xiang, L. junjhou, W. Jieyi and G. Guanqun, "QoS Routing based on Genetic Algorithms", Computer Communications, 1999, pp. 1392-1399.

[39] Zhang Q. and Lenug Y. W., "An orthogonal genetic algorithm for multimedia multicast routing", IEEE Tran. Of Evolutionary Computation, 1999, pp. 53-62.

[40] F. Xiang, L. Junzhou, W. Jieyi and G. Guanqun, "QoS routing based on genetic algorithm", Computer Communications, 22(1999), pp. 1392-1399.

[41] H. F. Salama, D.S Reeves and Y. Viniotis, "Evaluation of Multicast Routing Algorithms for Real Time Communication on High-Speed Networks", IEEE Journal on Selected Areas in Communications, Vol.15, 1997, pp 332-345.

[42] A.Benerjee , M.Faloutsos and R.Pankaj, " Designing QoSMIC: A Quality of Service Multicast Internet Protocol", IETF Draft, 1998.

[43] S.Deering , "A Multicast Extension to the Internet Protocol", RFC 966, IETF Draft, Dec 1985.

[44] S.Deering, "Host Extensions for IP Multicasting", RFC 1112, IETF Draft, Aug 1989.

[45] B.K.Haberman and G.Rouskas, "Cost,Delay and Delay Variation Conscious Multicast Routing ", Technical Report TR-97-03, North Carolina State University,1997.

[46] L. Kleinrock, "QUEUING SYSTEMS", Vol. 1: Theory, John Wiley & Sons,1975.

[47] V.P.Kompella, J.C.Pasquale and G.Polyzos, "Two Distributed Algorithms for Multicasting Multimedia Applications", Proceedings of ICCCN,1993, pp.343-349.

[48] V.P.Kompella, J.C.Pasquale and G.Polyzos, "Multicast Routing in Multimedia Communications", ACM/IEEE Transactions of Networking, Jun.1993, pp. 286-292.

[49] L.Kou, G. Markowski and L.Berman, "A Fast Algorithm for Steiner Trees", Acta Informatica, 1981, pp.141-145.

[50] R. Nelson, "Probability, Stochastic Processes and Queuing Theory", Springer Verlag, 1995.