World Academy of Science, Engineering and Technology
International Journal of Electronics and Communication Engineering
Vol:4, No:1, 2010

Multiobjective Optimization Solution for Shortest Path Routing Problem

C. Chitra and P. Subbaraj

***Abstract*—**The shortest path routing problem is a multiobjective nonlinear optimization problem with constraints. This problem has been addressed by considering Quality of service parameters, delay and cost objectives separately or as a weighted sum of both objectives. Multiobjective evolutionary algorithms can find multiple pareto-optimal solutions in one single run and this ability makes them attractive for solving problems with multiple and conflicting objectives. This paper uses an elitist multiobjective evolutionary algorithm based on the Non-dominated Sorting Genetic Algorithm (NSGA), for solving the dynamic shortest path routing problem in computer networks. A priority-based encoding scheme is proposed for population initialization. Elitism ensures that the best solution does not deteriorate in the next generations. Results for a sample test network have been presented to demonstrate the capabilities of the proposed approach to generate well-distributed pareto-optimal solutions of dynamic routing problem in one single run. The results obtained by NSGA are compared with single objective weighting factor method for which Genetic Algorithm (GA) was applied.

***Keywords*—**Multiobjective optimization, Non-dominated Sorting Genetic Algorithm, Routing, Weighted sum.

I. Introduction

THE ability of multiobjective evolutionary algorithms to find multiple pareto-optimal solutions in one single run have made them attractive for solving problems with multiple and conflicting objectives. During the last decade, several multiobjective evolutionary algorithms have been proposed which are aimed at finding the pareto-optimal front and also achieve diversity in the obtained pareto-optimal front.

A computer network is an interconnected group of computers with the ability to exchange data. Today, computer networks are the core of modern communication. Routing is one of the most important issues that have a significant impact on the network's performance. An ideal routing algorithm should strive to find an optimum path for packet transmission within a specified time so as to satisfy the Quality of Service (QoS). Current routing protocols use a simple metric and shortest path algorithm so as to work out the routes [1]. In QoS routing, routes must be determined by requirements based on features of the data flows, such as cost, delay, bandwidth etc. There are two main goals that need to be achieved by the QoS routing algorithm. The first goal is to find a path that satisfies the QoS requirements [2]. The second goal is to optimize the global network resource utilization. Many applications, such as audio, video conferencing or collaborative environments and distributed interactive simulations have multiple QoS requirements such as bandwidth, packet delay, packet loss, cost etc. [3].

The simple multiobjective method is to form a composite objective function as the weighted sum of the objectives, where a weight for an objective is proportional to the preference factor assigned to that particular objective. This method of scalarizing an objective vector into a single composite objective function converts the multiobjective optimization problem into a single objective optimization problem [4], [5]. In an ideal multiobjective optimization procedure, multiple trade-off solutions are found. Higher level information is used to choose one of the trade-off solutions. It is easy to realize that single objective optimization is a degenerate case of multiobjective optimization.

In this paper, NSGA based approach is proposed for solving the dynamic routing optimization problem. The problem is formulated as a nonlinear constrained multiobjective optimization problem, where cost and delay are treated as competing objectives. A diversity-preserving mechanism is developed and superimposed on the search algorithm to find widely different pareto-optimal solutions. Several runs are carried out on a sample network and the results are compared to the single objective optimization by weighted sum method.

This paper is organized as follows: Section 2 describes the routing problem formulation, Section 3 describes the principles of multiobjective optimization, Section 4 describes the implementation of NSGA into dynamic routing and Section 5 discusses the results followed by conclusion in Section 6.

II. Problem Formulation

The routing problem is formulated as a multiobjective mathematical programming problem which attempts to minimize both delay and cost simultaneously, while satisfying the constraints.

The topology of a multihop network is specified by an undirected graph, where the set of nodes is V, and the set of its link is E. There is a cost C_{ij} associated with each link. The costs are specified by the cost matrix $C = [C_{ij}]$, where C_{ij} denotes a cost of transmitting a packet on link (i, j). The

C. Chitra is with the Department of Electronics & Communication Engineering, Arulmigu Kalasalingam College of Engineering, Krishnankoil, 626190, Tamilnadu, INDIA (phone: +91-4563-289042; e-mail: mahadevchitra@yahoo.co.in).

P. Subbaraj is with the Theni Kammavar Sangam College of Technology, Theni, 625534, Tamilnadu, INDIA (phone: +91-4546-291994; e-mail: potti_subbu@yahoo.co.in).

delay is specified by the delay matrix $D = [d_{ij}]$, where d_{ij} denotes the propagation delay of transmitting a packet on link (i,j). Source and destination nodes are denoted by S and D, respectively. Each link has the link connection indicator denoted by X_{ij}, which plays the role of a chromosome map providing information on whether the link from node i to node j is included in a routing path or not. If the link is used then the binary variable is 1 else it is 0. A path from node V_i to node V_j is a sequence of nodes from V in which no node appears more than once. A path can also be equivalently represented as a sequence of nodes (V_i, V_l, . . . , V_k, V_j). For the example given in Fig. 1, (1, 2), (2, 3), (3, 8) (8, 14) and (14, 20) is a path from node 1 to node 20. The path representation is (1, 2, 3, 8, 14, 20). The problem is to find a path between the source and destination nodes having minimum total cost and minimum end to end delay [6],[7].

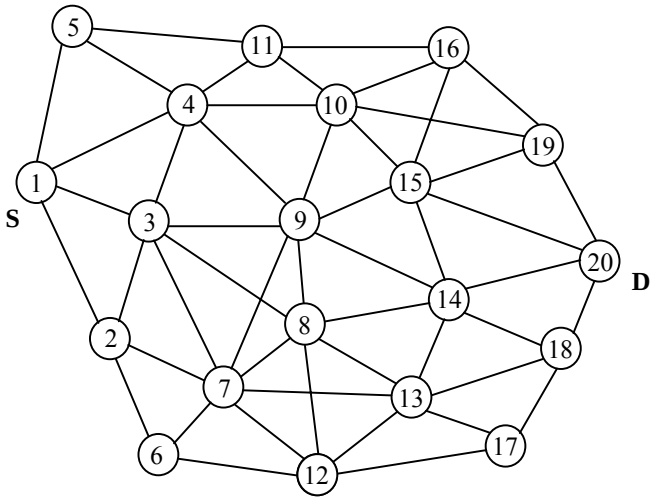

Fig. 1 A simple undirected graph with 20 nodes and 48 edges

Objectives:

a) *Cost*:
The total cost function is the sum of cost of link along the path from the source to the destination. The cost can be expressed as

$$f_1 = \sum_{(i,j)\in E} C_{ij} X_{ij} \tag{1}$$

b) *Delay*:
The total delay function is the sum of delay of the link along the path from the source to the destination. There are three basic concepts of delay, *viz*. switching delay, queuing delay and propagation delay.

The delay can be expressed as

$$f_2 = \sum_{(i,j)\in E} d_{ij} X_{ij} \tag{2}$$

Subject to the constraints

$$\sum_{(i,j)\in E} X_{ij} - \sum_{(i,j)\in E} X_{ji} = 1; \quad i = S \tag{3}$$

$$\sum_{(i,j)\in E} X_{ij} - \sum_{(i,j)\in E} X_{ji} = -1; \quad i = D \tag{4}$$

$$\sum_{(i,j)\in E} X_{ij} - \sum_{(i,j)\in E} X_{ji} = 0; \quad i \neq S, i \neq D \tag{5}$$

$$X_{ij} = 0 \text{ or } 1 \tag{6}$$

Constraints (3), (4) and (5) are flow conservation constraints. Constraint (3) ensures that the total flow emerging from ingress node to egress node should be 1. Constraint (4) ensures that the total flow coming towards an egress node should 1. Constraint (5) ensures that for any intermediate node different from the ingress node and egress node, the sum of their output flows to the egress node D minus the input flows with destination egress node D should be zero. The variable X_{ij} in (6) takes values 0 or 1, to show whether or not the link (i, j) is used to carry information to the egress node D.

III. Principles of Multiobjective Optimization

Most real-world problems involve simultaneous optimization of several objective functions. Generally, these functions are often competing and conflicting objectives. Multiobjective optimization having such conflicting objective functions gives rise to a set of optimal solutions, instead of one optimal solution. Here no solution can be considered to be better than any other with respect to all objectives. These optimal solutions are known as pareto-optimal solutions. Classical optimization methods can at the best find one solution in one simulation run. Therefore these methods are inconvenient to solve multiobjective optimization problems. Evolutionary Algorithms, on the other hand, can find multiple optimal solutions in one single simulation run due to their population based approach.

Generally, multiobjective optimization problem consisting of a number of objectives and several constraints can be formulated as follows:

Minimize/maximize

$$f_i(x) \qquad i = 1,2,3, \ldots N_{\text{objectives}}$$

Subject to

$$g_k(x) = 0,\ k = 1,2,3,\ldots K$$
$$h_l(x) \leq 0,\ l = 1,2,3,\ldots L$$

where f_i is the i^{th} objective function, x is a decision vector that represents a solution and $N_{\text{objectives}}$ is the number of objectives. K and L are the number of equality and inequality constraints respectively. In many real-life problems, objectives under consideration conflict with each other. Hence, optimizing x with respect to a single objective often results in unacceptable results with respect to the other objectives. Therefore, a perfect multiobjective solution that simultaneously optimizes each objective function is almost impossible. A reasonable solution to a multiobjective problem is to investigate a set of solutions, each of which satisfies the objectives at an acceptable level without being dominated by any other solution [8].

For a multiobjective optimization problem, any two solutions x^1 and x^2 can have one of two possibilities: one dominates the other or none dominates the other. In a

minimization problem, without loss of generality, a solution x^1 dominates x^2 if the following two conditions are satisfied:

$$\forall i \in \{1,2,..., N_{objective}\} : f_i(x^1) \leq f_i(x^2),$$
$$\exists j \in \{1,2,..., N_{objective}\} : f_j(x^1) < f_j(x^2).$$

If any of the above conditions is violated, the solution x^1 does not dominate the solution x^2. If x^1 dominates the solution x^2, x^1 is called the non-dominated solution within the set $\{x^1, x^2\}$. The solutions that are non-dominated within the entire search space are denoted as *pareto-optimal* and constitute the *pareto-optimal set* or *pareto-optimal front*. A solution is said to be pareto-optimal if it is not dominated by any other solution in the solution space. A pareto-optimal solution cannot be improved with respect to any objective without worsening at least other objective. The set of all feasible non-dominated solutions is referred to as the pareto-optimal set, and for a given pareto-optimal set, the corresponding objective function values in the objective space is called the pareto front. For many problems, the number of pareto-optimal solutions is enormous, may be infinite also. The ultimate goal of a multiobjective optimization algorithm is to identify solutions in the pareto-optimal set. However, identifying the entire pareto-optimal set for many multiobjective problems is practically impossible due to its size. In addition, for many problems, especially for combinatorial optimization problems, proof of solution optimality is computationally infeasible. Therefore, a practical approach to multiobjective optimization is to investigate a set of solutions that represent the pareto-optimal set as much as possible [9]. With these concerns in mind, a multiobjective optimization approach should achieve the following three conflicting goals:

1. The best known pareto-front should be as close as possible to the true pareto-front. Ideally, the best-known pareto-set should be a subset of the pareto-optimal set.
2. Solutions in the best-known pareto set should be uniformly distributed and diverse over the pareto-front in order to provide the decision maker a true picture of trade-offs.
3. In addition, the best-known pareto-front should capture the whole spectrum of the pareto-front. This requires investigating solutions at the extreme ends of the objective function space.

This paper presents common approaches used in multiobjective algorithms to attain these three conflicting goals while solving a multiobjective optimization routing problem.

IV. Implementation of NSGA into Dynamic Routing Problem

A description of the NSGA algorithm is given in this section. The difficulties of classical methods can be listed as: To find the multiple pareto-optimal solutions, the algorithm has to be applied many times. Most algorithms demand some knowledge about the problem being solved. The spread of pareto-optimal solutions depend on efficiency of the single objective optimizer. In general, the goal of a multiobjective optimization is to find the pareto-optimal front and also maintain population diversity in the set of the nondominated solutions.

NSGA uses ranking selection method to emphasize current nondominated solutions and a niching method to maintain diversity in the population. Two main steps are followed in the algorithm (i) fitness assignment which prefers nondominated solutions and (ii) fitness sharing strategy which preserves diversity among solutions of each nondominated front.

A. Fitness assignment

The basic idea of this approach is to find a set of solutions in the population that are nondominated by the rest of the population. The following approach describes a step-by-step procedure for finding the non-dominated set in a given set P of size N.

Step 1: Set solution counter, i =1 and create an empty non-dominated set P'.

Step 2: For a solution $j \in P, j \neq i$, check if solution j dominates solution i. If yes go to step 4.

Step 3: If more solutions are left in P, $j = j + 1$ and go to step 2; otherwise, set $P' = P' \in \{i\}$.

Step 4: $i = i + 1$. If $i \leq N$, go to step 2; otherwise stop and P' is the non-dominated set.

These solutions represent the first front P1 and are eliminated from further contention. This process continues until the population is properly ranked. After classification has been completed, all solutions in the first set are said to belong to the best non-dominated set in the population. The second best solutions in the population are those that belong to the second set, and so on. Since all solutions in the first non-dominated set P1 are equally important in terms of their closeness to the pareto-optimal front relative to the current population, the same fitness is assigned to all of them. Assigning more fitness to solutions belonging to a better non-dominated set ensures a selection pressure towards the pareto-optimal front. In order to achieve the second goal, diversity among solutions in a front must also be maintained. In NSGA, the sharing function method is used for this purpose.

B. Fitness sharing

The basic idea behind sharing is: the more individuals are located in the neighborhood of a certain individual, the more its fitness value is degraded [10]. The neighbourhood is defined in terms of a distance measured and specified by the niche radius σ_{share}. Given a set of n_k solutions in the k^{th} front each having a dummy fitness value f_k , the sharing procedure is performed in the following way for each solution $i = 1,. . ., n_k$:

Step 1: The sharing function is used front-wise [11], [12]. That is, for each solution i in the first front, the normalized Euclidean distance d_{ij} from another solution j in the same front is calculated as follows:

$$d_{ij} = \sqrt{\sum_{k=1}^{p1} \left(\frac{x_k^{(i)} - x_k^{(j)}}{x_k^{\max} - x_k^{\min}} \right)^2}$$

The parameters $x_k^{\max}$ and $x_k^{\min}$ are the upper and lower bounds of variable x_k.

Step 2: Then the sharing function value is calculated using the following equation

$$Sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{share}}\right)^2, & \text{if } d \le \sigma_{Share}; \\ 0 & \text{otherwise} \end{cases}$$

Step 3: Calculate niche count for the i^{th} solution as follows:

$$nC_i = \sum_{j=1}^{n_k} Sh(d_{ij})$$

Step 4: Calculate the shared fitness value as

$$f_i^{'} = f_i / nC_i$$

This procedure is continued for all population members and the corresponding $f_i^{'}$ is found. The smallest value $f_k^{\min}$ of all $f_i^{'}$ in that particular front is found for further processing. The dummy fitness of the next non-dominated front is assigned to be $f_{k+1} = f_k^{\min} - \in_k$, where $\in_k$ is a small positive number [13].

C. Initialization

A routing path is encoded by a string of positive integers that represent the IDs of nodes through which the path passes. Each locus of the string represents an order of a node that is indicated by the gene of the locus. The gene of the first locus is for the source node and the one at the last locus is for the destination node. The length of a routing path should not exceed the maximum length n, where n is the number of nodes in the network. Two encoding techniques are proposed as following.

i. Random Based Encoding (RBE)
ii. Priority Based Encoding (PBE)

D. Random Based Encoding

A chromosome or an individual consists of integer node IDs that form a path from the source node to a destination node. The chromosome is essentially a list of nodes along the constructed path, ($S \rightarrow N_1 \rightarrow N_{k-1} \rightarrow N_k \rightarrow D$) [3]. Each chromosome corresponds to a potential solution. The initial population is composed of a certain number of chromosomes. To explore the genetic diversity for each chromosome, the corresponding routing path is randomly generated.

A random path is searched starting from source node S to destination node D by randomly selecting a node N from the list of n nodes that is the neighborhood of S. Then another node N_k is randomly selected from the list of nodes. This process is repeated until the destination D is reached. Since the path should be loop-free, the nodes that are already included in the current path are excluded, thereby avoiding re-entry of the same node. The initial population is generated as follows:

Step 1: Start ($i = 0$).

Step 2: Generate chromosome *Chi*: search a random loop-free path $P(S, D)$;

Step 3: $i = i + 1$. If $i < q$, go to *Step 2*, otherwise, stop. Here $q = 20$.

Thus, the initial population $P_t = \{Ch_0, Ch_1, \ldots, Ch_{q-1}\}$ is obtained. An example of random based chromosome encoding from S to D is shown in Fig. 2.

Locus	1	2	3	…	$N-2$	$N-1$	N
Chromosome	S	N_1	N_2	…	N_{k-2}	N_{k-1}	N_k

Fig. 2 Example of random based encoding scheme

E. Priority Based Encoding

Special difficulties arise when a random sequence of edges usually does not correspond to a path. To overcome such difficulties, an indirect approach is adopted by encoding some guiding information to construct a path. The path is generated by sequential node appending procedure beginning from the specified node 1 and terminating at the specified node n, where $n = 20$. At each step, there are usually several nodes available for consideration. Each node is assigned a priority with a random mechanism and adds the one with the highest priority into path. A gene in a chromosome is characterized by two factors: locus, *i.e.,* the position of gene located within the structure of chromosome, and allele, *i.e.,* the value the gene takes. In the proposed priority-based encoding method, the position of a gene is used to represent node ID and its value is used to represent the priority of the node for constructing a path among candidates. A path can be uniquely determined from this encoding scheme.

An example of chromosome generated using priority based encoding scheme is shown in Fig. 3. To find a path from source node 1 to destination node 20, a node which is connected to node 1 is identified first. As seen from Fig. 1, the nodes 2, 3, 4 and 5 are such nodes to be considered. The priorities for them are 5, 7, 6 and 3 respectively. The node 3 has the highest priority and is put into the path. The possible nodes from node 3 are nodes 1, 2, 4, 7, 8 and 9. The priorities of these nodes are 2, 5, 6, 9, 4 and 10 respectively. Since node 9 has a larger value than the other nodes, it is taken as the next node while constructing the path. Then the set of nodes that are available for next consideration are chosen and the one with the highest priority among them is selected. The same procedure is repeated until a complete path from the source node1 to the destination node 20 is obtained (1, 3, 9, 15, 20).

F. Tournament selection

Selection plays an important role in improving the average quality of the population by passing the high quality chromosomes to the next generation. The individual with the

Node ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Priority	2	5	7	6	3	1	9	4	10	8	14	16	30	20	44	70	57	46	35	90

Fig. 3 Example of Priority-based encoding

lowest front number is selected if the two individuals are from different fronts. The individual with the highest crowding distance is selected if they are from the same front. A higher fitness is assigned to individuals located on a sparsely populated part of the front [14]. In each iteration the N existing individual parents generate N new individual offspring. Both parents and offspring compete with each other for inclusion in the next iteration.

G. Crossover and mutation

The first genetic operation done to the chromosomes in the mating pool is crossover. The idea behind crossover is to create an information exchange between two chromosomes. By doing so, the algorithm will explore new paths and hopefully be able to find better paths in the process [15]. Two crossover schemes that are proposed here are Node Based Crossover (NBX) and Partially Mapped Crossover (PMX).

H. Node Based Crossover

The cross over scheme is an adaptation of the one-point cross-over. For each pair of paths a locus is randomly selected from one of the chromosomes and the node ID of the locus is matched with the genes in the other chromosome. If there is a match then crossover is performed otherwise two new paths are selected for crossover until the mating pool is empty. Here the loci of both individuals need not be the same. That is, the crossover does not depend on the position of nodes in routing paths [16], [17]. Fig. 4 and Fig. 5 show an example of the crossover procedure for NBX.

$S \longrightarrow N_1 \longrightarrow N_2 \longrightarrow N_4 \longrightarrow N_5 \longrightarrow D$

$S \longrightarrow N_2 \longrightarrow N_3 \longrightarrow N_5 \longrightarrow N_6 \longrightarrow N_7 \longrightarrow D$

Fig. 4 Chromosome before crossover

$S \longrightarrow N_1 \longrightarrow N_2 \longrightarrow N_3 \longrightarrow N_5 \longrightarrow N_6 \longrightarrow N_7 \longrightarrow D$

$S \longrightarrow N_2 \longrightarrow N_4 \longrightarrow N_5 \longrightarrow D$

Fig. 5 Chromosome after crossover

Figures 4 and 5 show the chromosomes before and after crossover respectively. In this, in the first chromosome, the third locus is chosen for crossover. The node *ID* in the third locus is N_2. In the second chromosome, a match for N_2 is found in the second locus. Therefore crossover is performed and the new chromosome is shown in Fig. 5.

I. Partially Mapped Crossover

PMX is a crossover of permutations which guarantees all positions that will be found exactly once in each offspring, i.e. both offspring receive a full complement of genes, followed by the corresponding filling in of alleles from their parents. PMX proceeds as follows:

1) The two chromosomes are aligned.
2) Two crossing sites are selected uniformly at random along the strings, defining a matching section.
3) The matching section is used to cross through position-by-position exchange operation.
4) Alleles are moved to their new positions in the offspring.

The parents that are selected for PMX are shown in Fig. 6. Here the first two cut points are selected uniformly at random along the parent strings. The sub strings between the cut points are called the mapping sections. Now the mapping section of the first parent is copied into the second offspring and the mapping section of the second parent is copied into the first offspring which is shown in Fig. 7.

Then offspring1 is filled up by copying the first two elements N_1, N_2 of the first parent. In case a node is already present in the offspring it is replaced according to the mapping. Here the mapping is defined as $N_4 \leftrightarrow N_3$, $N_5 \leftrightarrow N_6 \leftrightarrow N_8$. For example the first two elements of parent1 N_1, N_2 are copied as the first two elements of the offspring1. The third element would be a N_3 like the first element of the mapped sections in offspring1. So there is already a N_3 present in offspring1. Hence, because of the mapping $N_4 \leftrightarrow N_3$ the third element of the offspring1 is chosen to be N_4. The seventh and ninth elements of offspring1 can be taken from the first parent. However, the eighth element of the offspring1 would be an N_8, which is already present. Because of the mapping $N_5 \leftrightarrow N_6 \leftrightarrow N_8$, it is chosen as N_5. Hence offspring1 and offspring2 are shown in Fig. 8.

Parent 1: $N_1 \longrightarrow N_2 \longrightarrow N_3 \longrightarrow N_4 \longrightarrow N_5 \longrightarrow N_6 \longrightarrow N_7 \longrightarrow N_8 \longrightarrow N_{20}$

Parent 2: $N_1 \longrightarrow N_7 \longrightarrow N_5 \longrightarrow N_3 \longrightarrow N_6 \longrightarrow N_8 \longrightarrow N_2 \longrightarrow N_4 \longrightarrow N_{20}$

Fig. 6 Parents for Partially Mapped Crossover

Offspring 1: $X \longrightarrow X \longrightarrow X \longrightarrow N_3 \longrightarrow N_6 \longrightarrow N_8 \longrightarrow X \longrightarrow X \longrightarrow X$

Offspring 2: $X \longrightarrow X \longrightarrow X \longrightarrow N_4 \longrightarrow N_5 \longrightarrow N_6 \longrightarrow X \longrightarrow X \longrightarrow X$

Fig. 7 Offsprings after mapping sections crossed

Offspring 1: $N_1 \longrightarrow N_2 \longrightarrow N_4 \longrightarrow N_3 \longrightarrow N_6 \longrightarrow N_8 \longrightarrow N_7 \longrightarrow N_5 \longrightarrow N_{20}$

Offspring 2: $N_1 \longrightarrow N_7 \longrightarrow N_8 \longrightarrow N_4 \longrightarrow N_5 \longrightarrow N_6 \longrightarrow N_2 \longrightarrow N_3 \longrightarrow N_{20}$

Fig. 8 Production of offsprings after crossover

Here, the source node and the destination nodes are fixed. Each partial route is exchanged and assembled and thus, two new routes are produced. In NBX, during crossover, there is a possibility of routes with loops. In order to avoid this, repair function is used as a countermeasure. If penalty functions are used it is not easy to come up with an appropriate penalty function. Therefore repair function finds and eliminates loops in a routing path without increasing computation cost. Loops in a chromosome can be repaired by performing a search along the chromosome to find repeated nodes. The nodes in between the repeated nodes are then eliminated. For example, assume that the chromosome shown in Fig. 9 contains a loop.

$$S \longrightarrow N_1 \longrightarrow N_2 \longrightarrow N_3 \longrightarrow N_2 \longrightarrow N_6 \longrightarrow N_7 \longrightarrow D$$

Fig. 9 Chromosome with loop formation

Here, there are two N_2 nodes in the chromosome which signifies that the path contains a loop. This chromosome can be fixed by eliminating one of the N_2 nodes and all the other nodes in between the two N_2 nodes. The repaired chromosome would be like the one given in Fig. 10.

$$S \longrightarrow N_1 \longrightarrow N_2 \longrightarrow N_6 \longrightarrow N_7 \longrightarrow D$$

Fig. 10 Repaired chromosome

But in PMX loop formation is avoided. Here there is no repetition of nodes. The repetition of nodes is avoided by a mapping function. Therefore PMX finds many new paths without increasing computational complexity. Here no repair function is needed. The objective of mutation is to create diversity in the population. The population undergoes mutation by an actual change or flipping of one of the genes of the candidate chromosomes, thereby keeping away from local optima [2].

V. Simulation Results

In order to test the capability of NSGA algorithm for the shortest path routing problem, an undirected network with randomly generated 20 nodes based on Waxman's model was considered. Each of the links in the network is associated with two additive Quality of Service parameters, cost and delay. The range of cost varies from 10 to 250 and the range of delay varies from 5 to 200. The simulation was carried out on an IBM PC with Pentium dual core processor and the coding was developed using MATLAB version 7.4, software package.

The problem is formulated as a multiobjective optimization problem, and NSGA is applied to minimize both the objectives simultaneously. Priority based encoding technique and partially mapped crossover methods are used. The simulation intends to show the behavior of a multiobjective evolutionary algorithm in terms of optimality of solutions and computational complexity. The algorithm was implemented and a series of simulation runs were conducted to test the effectiveness of the routing algorithm. For all the runs, the sender is always the first node and the receiver is the twentieth node since that would give the largest number of possible paths in the network. The population size and maximum number of generations have been selected as 20 and 100 respectively. The probability for crossover, P_c and mutation, P_m are 0.8 and 0.1 respectively.

The pareto-optimal front discovered by the proposed approach is shown in Fig. 11. The CPU execution time was found to be 5681 seconds.

For comparison purposes, the problem has been converted to a scalar optimization problem by linear combination of cost and delay as follows:

$$\text{Minimize} \quad w f_1 + (1 - w) f_2$$

where w is the weighting factor [6]. The problem is minimized using GA. To generate 20 nondominated solutions, the algorithm was run 20 times with varying w as a random number w = rand [0,1]. The solutions obtained by GA for the routing problem considered are plotted in Fig. 12. The execution time was found to be 21,206 seconds.

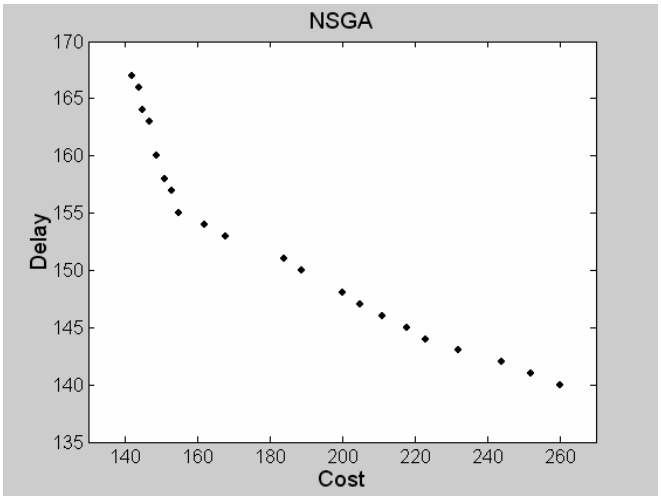

Fig.11 Pareto-optimal front of NSGA in last generation

Comparing the results shown in Fig. 11 and Fig. 12, it can be concluded that: (a) the 21 solutions that present the results of the proposed technique have been obtained in a single run while the 20 solutions shown in Fig. 12 has been obtained in 20 separate runs; (b) the number of solutions found by the proposed approach depends on the cost and delay data available which is generated randomly based on Waxman's model described in Section. 4.1.2. (c) the solutions of NSGA

approach shown in Fig. 11 have better diversity characteristics and well-distributed over the trade-off surface (d) there is no guarantee that the single objective optimizer will span over the entire trade-off surface while the proposed NSGA approach has an impeded diversity-preserving mechanism through fitness sharing procedure.

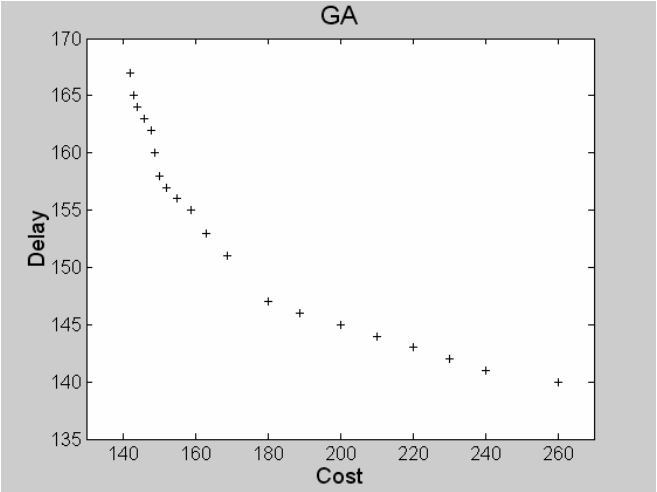

Fig. 12 Pareto-optimal front of linear combination in 20 separate runs

From Fig. 11, it could be observed that, NSGA has detected a large number of solutions on the pareto-optimal front in a single run. The diversity of the pareto-optimal set over the trade-off surface is shown in Fig. 11. Preferences for the minimum cost or delay need not be specified before the model run. With the solutions from NSGA, the decision-maker has the opportunity to visualize trade-offs and may be inclined to accept a very small violation of the delay requirement for a large cost saving. In a multiobjective problem the 'best' solution found would be totally up to the decision maker and not to the analyst. This is an advantage when compared to the case where MOOP is solved by normalizing and combining objectives into a single one. Another advantage of the multiobjective optimization problem is that the objective functions are simple to formulate and do not require complex mathematical tools to implement.

Convergence property of NSGA

In this section, the convergence property of NSGA algorithm is analyzed by taking the two objectives cost and delay, one at a time, adopting different encoding techniques and different cross-over techniques. Using the two different encoding techniques *viz*. random based encoding and priority based encoding and two cross-over techniques *viz*. node based crossover and partially mapped crossover four different combinations are formed as follows:

1) Random Based Encoding and Node Based Cross-over (RBE and NBX)
2) Random Based Encoding and Partially Mapped Cross-over (RBE and PMX)
3) Priority Based Encoding and Node Based Cross-over (PBE and NBX)
4) Priority Based Encoding and Partially Mapped Cross-over (PBE and PMX)

At first, the cost function alone is considered. The NSGA algorithm was applied with all the four combinations. The experiments were done on the same undirected network described in Section 4.1.2. The population size and maximum number of generations have been selected as 20 and 100 respectively. The probability for crossover, P_c and mutation, P_m are 0.8 and 0.1 respectively. Variation of cost against the number of generations for all the four combinations of encoding and cross-over techniques is shown in Fig. 13.

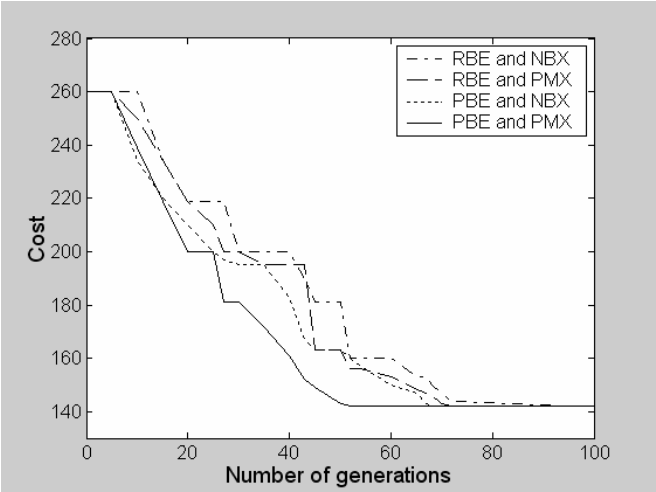

Fig. 13 Variation of average network cost against number of generations

With reference to Fig. 13, it is seen that, when random based encoding and node based crossover is used, the cost converges at 95th generation, and is the worst convergence characteristic obtained. Since the initial population is encoded at random, the possibility of a chromosome to be a feasible solution is less and most of the solutions are invalid (infeasible) solutions, and takes more generations to converge.

When the combination RBE and PMX is used, the cost function converges at 72nd generation. Again, as the initial population is encoded at random, the possibility of a chromosome to be a feasible solution is less. When PMX is used, the number of feasible solution increases and so the objective function converges at 72nd generation earlier than the previous case. If PBE is used for the generation of initial population, all the chromosomes present in the initial population are feasible solutions. When PBE and NBX are used the cost converges at the 68th generation, thus, further improvement in convergence is obtained. When PBE and PMX are used, the objective function converges at the 52nd generation. Since all the chromosomes in the initial population are feasible and also PMX avoids repetition of nodes and loop formation, it converges very fast, i.e., at 52nd generation. This combination of initial population generation and cross-over technique is found to be the best as far as the convergence characteristics are compared.

The same experiment was conducted for the next objective function of delay for the same network and with same control parameters. The convergence of NSGA with all the four combinations of initial population generation and cross-over are shown in Fig. 14.

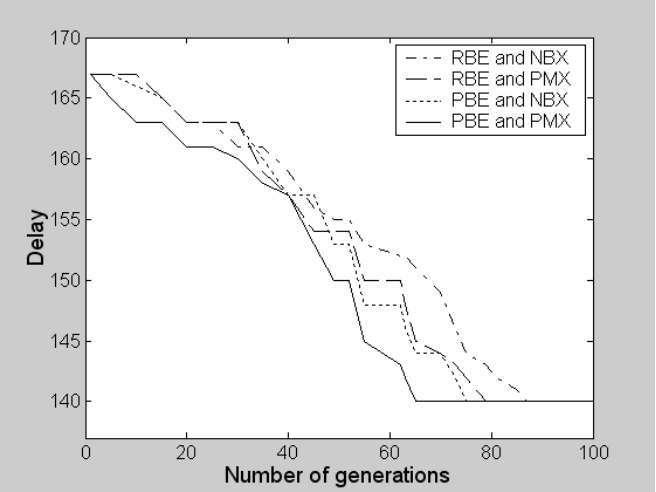

Fig. 14 Variation of average network delay against number of generations

It is observed from Fig. 14 that, when PBE and PMX methods are used the objective function converges fast. When RBE and NBX are used, the function converges at 87th generation, which seems to be the worst characteristics. As discussed before, the reason for such worst convergence is, the generation of initial population randomly. But when RBE technique and PMX is used the minimum delay was obtained at the 79th generation. When PBE technique and NBX methods were used the optimal result was obtained at the 75th generation. The combination of PBE and PMX techniques results in the proposed technique to converge at 65th generation. This demonstrates that NSGA algorithm with PBE and PMX techniques outperforms other three combinations.

However, since there are many elements of randomness in this algorithm, the number of generations required to find a solution is not the same every time the algorithm is executed even though the parameters used are exactly the same. This experiment is aimed to identify the number of generations required to find a feasible path with the same QoS parameters, cost and delay. The number of generations required to find a solution can vary. The fact that sometimes it takes a very large number of generations to find a solution is probably caused by the state of the initial population. This initial population may be of a very low quality that it takes many generations to find an acceptable solution. However, this does not seem to be a critical problem since most of the times the number of generations required to find a solution is quite low and acceptable. One parameter that can be modified to remedy this problem is the population size. The idea behind NSGA is to have the complete solutions provided by the population to slowly converge to optimal or exact solution.

VI. Conclusion

In this paper, a feasible multiobjective evolutionary algorithm, NSGA, was proposed and simulated to solve the routing problem in communication networks. This paper describes the implementation of NSGA algorithm to the network problem. The results obtained show that NSGA may be an efficient approach for multiobjective shortest path problem. It is particularly beneficial when intractability and memory issues become obstructions to find efficient solutions to the multiobjective shortest path problems. The experimental results obtained from the multiobjective solution revealed that the number of pareto points increase with the number of generations. The results show that the NSGA algorithm is efficient for solving multiobjective routing problem where multiple pareto-optimal solutions can be found in one simulation run. In addition, the non-dominated solutions obtained are well distributed and have satisfactory diversity characteristics. The approach is quite flexible so that other formulations using different objectives and/or a larger number of objectives are possible. Simulation experiments demonstrate the quality of solutions and computational efficiency of NSGA. Various combinations of encoding and cross over methods were used for the demonstration of the algorithm and it is found that PBE and PMX are best compared to the other combinations.

Acknowledgment

Authors thank the authorities of Arulmigu Kalasalingam College of Engineering, Krishnankoil, India and Theni Kammavar Sangam College of Technology, Theni, for the facilities provided to carry out this work.

References

[1] George N. Rouskas and Ilia Baldine, "Multicast Routing with End-to-End Delay and Delay Variation Constraints," *IEEE Journal on Selected Areas in Communications*, vol.15, No.3, pp.346-356, 1997.

[2] Jose Craveirinha, Rita Girao-Silva and Joao Climaco, "A meta-model for multiobjective routing in MPLS networks," *Central European Journal of Operation Research (Springer)*, vol.16, No.1, pp.79-105, 2008.

[3] Sriram, R., Manimaran, G., and Siva Ram Murthy, C., "Preferred link based delay-constrained least-cost routing in wide area networks," *Computer Communications*, vol. 21, pp. 1655–1669, 1998.

[4] Aluizio F. R. Araujo, Cicero Garrozi, Andre R. G. A. Leitao and Maury M. Gouvea Jr., "Multicast Routing Using Genetic Algorithm Seen as a Permutation Problem", 20th *International Conference on Advanced Information Networking and Applications* (AINA'06), Vienna, 2006, vol. 1, pp. 477-484.

[5] Chang Wook and Ramakrishna, R.S., "A genetic algorithm for shortest path routing problem and the sizing of populations", *IEEE Transactions on Evolutionary Computation*, vol.6, No.6, pp.566-579, 2002.

[6] Ha Chen and Baolin Sun, "Multicast Routing Optimization Algorithm with Bandwidth and Delay Constraints Based on GA," *Journal of Communication and Computer*, vol. 2, No.5, pp.63-67, 2005.

[7] Hayder A. Mukhef, Ekhlas M. Farhan and Mohammed R. Jassim (2008) "Generalized Shortest Path Problem in Uncertain Environment Based on PSO," *Journal of Computer Science*, vol. 4, No. 4, pp. 349-352.

[8] Abido, M. A., "A novel multiobjective evolutionary algorithm for environmental/economic power dispatch," *Electric Power Systems Research*, vol. 65, No. 1, pp 71-81, 2003.

[9] Kalyanmoy Deb and Santosh Tiwari, "Omni-optimizer: A generic evolutionary algorithm for single and multiobjective optimization," *European Journal of Operational Research*, vol. 185, No. 3, pp. 1062-1087, 2008.

[10] Srinivas, N. and Deb, K., "Multiobjective Optimization using Non-dominated Sorting in Genetic Algorithm," *Evolutionary Computation*, vol. 2, No. 3, pp. 221-248, 1994.
[11] Kalyanmoy Deb, *Multiobjective Optimization using Evolutionary Algorithms*. New York: John Wiley & Sons., 2001, ch. 5.
[12] Omar Al Jadaan, Lakishmi Rajamani, C. R. Rao, "Non-dominated ranked genetic algorithm for Solving multiobjective optimization Problems: NRGA", *Journal of Theoretical and Applied Information Technology*, pp. 60-67, 2008.
[13] K. Rath and S. N. Dehuri, "Non-dominated Sorting Genetic Algorithms for heterogeneous Embedded System Design", *Journal of Computer Science,* 2 (3), pp. 288-291, 2006.
[14] N. Srinivas and Kalyanmoy Deb, "Muiltiobjective Optimization Using nondominated Sorting in Genetic Algorithms," *Evolutionary computation*, vol. 2, No. 3, pp. 221-248, 2007.
[15] Jose Maria A. Pangilinan and Gerrit K. Janssens, "Evolutionary Algorithms for the Multiobjective Shortest Path Problem," *World Academy of Science*, *Engineering and Technology*, vol. 25, pp. 205-210, 2007.
[16] Salman Yussof and Ong Hang See, "Finding Multi-Constrained Path Using Genetic Algorithm," *Proc*. *IEEE International Conference on Telecommunications and Malaysia International Conference on Communications*, Penang, 2007, vol.1, pp.1-6.
[17] Salman Yussof and Ong Hang See, "QoS Routing for Multiple Additive QoS Parameters using Genetic Algorithm," *Proc*. *International Conference on Communication*, vol.1, pp.99-104, 2005.

Ms C. Chitra was born at Dindigul, India in 1974. She graduated in Electronics and Communication Engineering and post graduated in Digital Communication and Network Engineering from Madurai Kamaraj University, India in the year 1996 and 2002 respectively. Since 2000, she has been a faculty of Electronics and Communication Engineering at Arulmigu Kalasalingam College of Engineering, Krishnankoil, India. Her fields of interest include Computer Networks, Satellite Communication and Mobile Computing. She is working towards her Ph.D degree in Anna University, Chennai, India.

Dr. P. Subbaraj was born at Kalingapatti, Tamilnadu, India, in 1957. He received his B.E (Electronics and Communication Engineering) in 1979 and M.E (Control Systems) in 1981 from PSG College of Technology, Coimbatore, India. He received his Ph.D (Optimization) in 1991 from IIT-Madras, Chennai, India. He was with Department of Electrical and Electronics Engineering, Thiagarajar College of Engineering, Madurai, India from 1981 to 2001. Now he is the Principal of Theni Kammavar Sangam College of Technology, Theni. He is a member of many professional bodies. He has presented nearly 90 papers in national and international conferences. He has more than 40 publications in international journals to his credit. His fields of interest are Computer Control, Optimization, Signal Processing and Evolutionary Computation.