Computing and Informatics, Vol. 36, 2017, 405–422, doi: 10.4149/cai_2017_2_405

FAST CONVERGING EVOLUTIONARY STRATEGY FOR MULTI-CONSTRAINT QOS ROUTING IN COMPUTER NETWORKS USING NEW DECODING MECHANISM

Hadi Shahriar Shahhoseini, Samaneh Torkzadeh

School of Electrical Engineering
Iran University of Science and Technology
Narmak, 16846-13114, Tehran, Iran
e-mail: hshsh@iust.ac.ir, samaneh_torkzadeh@elec.ac.ir

Abstract. In recent years, real-time multimedia applications' demands such as Voice-on-IP (VoIP) and video conference are extremely increased which require QoS routing. This type of routing has been considered as an NP-Complete problem since it requires satisfying multiple constraints. Many solutions have been proposed to solve it, but most of them are complex and time consuming. In this paper, a novel multi-constraints QoS routing algorithm is proposed based on Evolutionary Strategies (ES). The algorithm preserves simplicity and offers a feasible solution in a few numbers of generations. This is due to a novel gene decoding mechanism that is used in the algorithm; and consequently more simple evolutionary operators can be applied. The simulation results show that our method outperforms previous algorithms in terms of speed and performance, so that it is 2.6 and 11.3 times faster, and its success ratio is also better.

Keywords: Convergence time, evolutionary algorithm, genetic algorithm, multi-objective, population size, QoS routing

1 INTRODUCTION

Internet is a connectionless network in which data packets may follow different paths to reach the destination. Network resources such as bandwidth and switch buffers are shared fairly between the packets of different sessions and that makes the network suitable for the existing applications like email and file transfer. However,

it cannot sustain real-time multimedia applications such as video conference, Video-on-Demand (VoD), internet telephone and Audio-on-Demand (AoD) which need satisfying multiple constraints like jitter, cost and packet loss [1]. There are two main reasons to prove that multimedia applications are not supported. First, the end-to-end delivery performance is not ensured because it does not support the resource reservation. Second, data packets may experience unexpected delays and get to the destination untidily, what is undesirable for real-time applications [2]. Due to these necessities, promising QoS by the network service provider is inevitable. It is worth noting that QoS refers to a set of performance metrics that should be satisfied for the requested service. Hence, QoS routing finds a path, which fulfills the specified QoS requirements [3]. The mentioned complexities make the multi-constraint routing problem, NP-Complete. Therefore, in recent years many studies have been done to solve this problem [4, 5, 6, 7]. In order to find all feasible paths in large-scale networks, many researchers use Evolutionary Algorithms (EAs); a generic population based meta-heuristic optimization algorithm. The main goal of all routing algorithms is selecting a path adaptively, intelligently and flexibly. The main drawbacks of the existing EAs can be classified into some categories:

1. considering only a single constraint [3] instead of real multiple QoS constraints
2. requiring to generate some sorts of tree graphs of the network which is not practical for real-time large networks
3. exploiting complex evolutionary operators which imposes a major time overhead [8].
4. requiring extra steps to validate and correct invalid individuals produced at the end of each stage [9].

In this paper, a routing algorithm is developed based on ES, which efficiently satisfies multimedia applications' constraints. The main feature of the proposed algorithm is its mode of gene decoding chromosomes that make it needless to use complex evolutionary operators and extra phases such as checking and repairing. These capabilities result in the fast convergence of the algorithm and reduce the run time. The main contributions of the paper can be summarized as follows:

1. Developing a routing algorithm based on EAs which satisfies multimedia applications' constraints while preserving functionality and simplicity.
2. Presenting a novel gene decoding mechanism which avoids loop existence in the paths, and markedly reduces the complexity of evolutionary operators.
3. The flexible fitness function of the algorithm is designed so that the QoS constraints can be simply extended to any number of parameters and be used to optimize larger targets and problems.
4. The algorithm converges into a feasible path in a very short time.

The performance of the algorithm is analyzed and compared with those of some other methods. The rest of paper is as follows: next section reviews some recent

related contributions to the field. In Section 3, the proposed algorithm is described and an example is presented to clarify it. The simulation results are offered in Section 4. Finally, Section 5 concludes the paper.

2 RELATED WORK

In recent years, researchers have proposed many QoS routing algorithms. In the following parts, we will review some of them and we will illustrate their advantages and drawbacks.

Considering [10] multi-constrained QoS multicast routing method proposed, using the GA. This method has been flooding-limited by using the minimum computation time and available resources in a dynamic environment. While trying to obtain the appropriate values for genetic operators, the algorithm tries to improve and optimize the routes based on GA. The main drawbacks of this method are its imprecise fitness function and long convergence time.

Considering [8] multi-objective multicast routing GA which is based on topological assisted tree structured encoding we find out that a combination of random routes from the multicast group destination nodes to the source node are used to form the chromosomes. The complex process for chromosome generating besides complex genetic operators alleviate the preferences of this method.

Chitra and Subbaraj [11] proposed a method called NSGA, which uses an elitist multi-objective EA to solve the dynamic shortest path routing problem in computer networks. They proposed a priority-based encoding scheme to initial population. This method has some advantages but lasts too many generations to converge into a feasible solution.

Monetomo et al. proposed a QoS routing algorithm based on GA, in which the genes order in chromosomes are the same as the order of their respective nodes in the path [12]. This algorithm like most other GAs starts with a random initial population and generates next generations by applying crossover and mutation. It uses ordinary single point crossover but a different mutation mechanism. This algorithm is not appropriate for QoS routing, since it employs only a single constraint, delay time, for computing the feasible routes. Moreover, the network may be overloaded due to many delay query packets generated in the algorithm. The main weakness of this method is its complex crossover operator that sometimes generates paths with loops, which in turn necessitate an extra step to check and repair the unwanted chromosomes. Therefore, one can claim that this algorithm is generally too costly.

In order to overcome the mentioned problems in the above algorithms, an algorithm called ARGA [13] is proposed that fixes the size of chromosomes by employing a novel gene coding method. In ARGA, the network and chromosome genes are described as tree and tree junction respectively and as a result the method guarantees the loop-free paths. By eliminating the extra check and repair phases for invalid chromosomes, this algorithm performs better than the previous ones, but still uses a single constraint, delay time, to compute the QoS routes.

In order for QoS routing, it is inevitable to consider multi constraints. Another algorithm named ARGAQ [14] was proposed to improve the previous one. In this method, the relation of two constraints is used: delay time and TSR (Transmission Success Ratio). Both ARGA and ARGAQ are the same in all steps but in fitness function computing. However, the path-selecting criterion adopted in the latter is much superior to the previous ones, but it is yet an approximation of QoS routing.

Leela et al. developed an algorithm, which satisfies multi-constraints QoS routing problem based on GA. They proposed a finding called Multi-constraint QoS Unicast Routing Using GA (MURUGA) [15], which satisfies the requirements of multimedia applications. The main drawback of this algorithm is that it requires validation and repair phases for the generated individuals at the end of generations.

Ting and Zhu [16] proposed a Gene Structure (GS) for encoding the MCR problem, and presented GSA algorithm to generate the GS. By using this technique, they guarantee that there will be no invalid chromosomes in the population thus their algorithm seems to have a good performance.

According to the discussed methods, the current paper intends to resolve some of their shortcomings; dealing with multi-constraints, removing additional stages, accelerating the convergence, being useful for a wide range of network sizes without losing the appropriateness of performance.

3 THE PROPOSED METHOD

In this section, our proposed algorithm will be described. First, it is necessary to explain the underlying network. In order to model the network, an undirected graph is used. It represents the set of nodes (vertices), the set of links (edges) and the constraints matrix of the network respectively.

Before describing the proposed algorithm, clarifying the applied parameters and considerations is essential:

- Each chromosome is a loop-free path from source to destination.
- Each gene represents a node.
- Population is a determined number of different chromosomes which completely depends on the graph structure.
- Fitness function evaluates the feasibility of a chromosome whether satisfying the QoS constraints, and determining its rank. between other population members.

The steps of our algorithm are as follows:

1. To generate the initial population, take some random paths from source to destination.
2. Evaluate the fitness of each chromosome using the fitness function.
3. Sort the population individuals based on the values obtained in previous steps.
4. Stop the algorithm whenever the first rank individual's fitness being less than 1, otherwise go to the next step.

5. Generate a new population using an ES operator.
6. Continue the algorithm until the stopping criterion would met.

3.1 The General Overview of the Proposed Algorithm

The proposed algorithm uses a heuristic mechanism for gene coding. This method not only avoids loop existence in the paths, but also remarkably reduces the complexity of evolutionary operators. These advantages let the algorithm to converge into a feasible path in a very short time. Our proposed fitness function decides whether a path is feasible or not. If not, it assigns a cost to its corresponding chromosome. In order to produce the next generation and reach into more fit individuals, an ES is used. In this method, unlike in Genetic Algorithm (GA), no crossover operator is applied and only mutation operator is used to evolve the population. In detail, parents will be selected based on their fitness value and then they produce offsprings. Only the top of offsprings and parents are permitted to stay alive and enter the next generation. Using this kind of evolutionary operation, i.e. mutation, helps efficiently in substituting the unfavorable paths with new more appropriate paths, which satisfy the desired requirements of a given application. The main feature of the proposed method – the particular gene decoding mechanism – has been verified through the simulation experiments. The flowchart of the proposed algorithm is shown in Figure 1.

3.2 The Algorithm's Procedure

In this subsection, we are going to explain the steps of the proposed algorithm in detail; its pseudo-code is shown in Algorithm 1. Generally, it has 7 steps which are described independently in the following:

Step 1: Gene Coding and Chromosome Structure. Supposing that the number of network's nodes is N, the length of chromosomes would be $N-1$. This means that the lengths of all chromosomes are fixed and equal. This property simplifies the algorithm and its implementation. Each gene includes a number between 1 to $N-1$. These numbers do not have any direct relation to real paths from source to destination. In fact, these numbers will be mapped into a feasible path using a heuristic mapping function. This mechanism avoids loops (i.e. chromosomes with duplicate genes) or any invalid individuals in the population. Therefore, there is no need to use extra phases for validation and repairing the unacceptable solutions.

Step 2: Generating the Initial Population. As mentioned above, the chromosome genes are numbers between 1 to the number of network's nodes minus 1. To produce the initial population, these numbers are generated randomly for all chromosomes. In the other words, the initial population consists of a number of chromosomes with equal number of genes, which are generated randomly.

These chromosomes do not represent real routes and must be translated into some potential paths using the decoding mechanism of the next step.

Step 3: Chromosome to Path Conversion. This step decodes the individuals into some potential paths from source to destination and has six stages, which are:

1. The source node is always the first node and the following algorithm is run for the i^{th} node.
2. The nodes which have a way to the destination and are not traversed yet are determined and their number is saved in nm.
3. The remainder resulted from dividing the i^{th} gene's content by nm specifies the next elected node.
4. Stages 2 and 3 are repeated until the destination node is obtained.
5. If the algorithm reaches into the last gene but the path did not touch the destination node, this chromosome is handed to the correction stage.
6. In the correction stage, the procedure is started from the last gene and if there is no path to the destination, i is decremented and the next node will be selected. Then, the algorithm will be repeated from stage 3.

Algorithm 1: code for the proposed algorithm

1. Set chromosomes length equal to the number of network graph's nodes (N) minus 1
2. Generate some chromosomes with random genes containing an integer number between 1 to $N - 1$
3. Convert the unfeasible chromosomes into real routes from the specified source to destination nodes using the following decoding mechanism
4. Set the source node as the first node and run the following algorithm for the i^{th} node
5. The nodes which have a way to the destination and are not traversed yet are determined and their number is saved in nm
6. The remainder resulted from dividing the i^{th} gene's content by nm specifies the next elected node
7. Repeat Stages 5 and 6 until reaching into the destination node
8. If the algorithm reaches the last gene but the path did not touch the destination node
9. Then hand over the chromosome to the correction stage:
10. Starting from the last gene;
11. If there is no path to the destination
12. Then decrement i and select the next node
13. Repeat the algorithm from stage 6
14. Evaluate the decoded paths using the fitness function

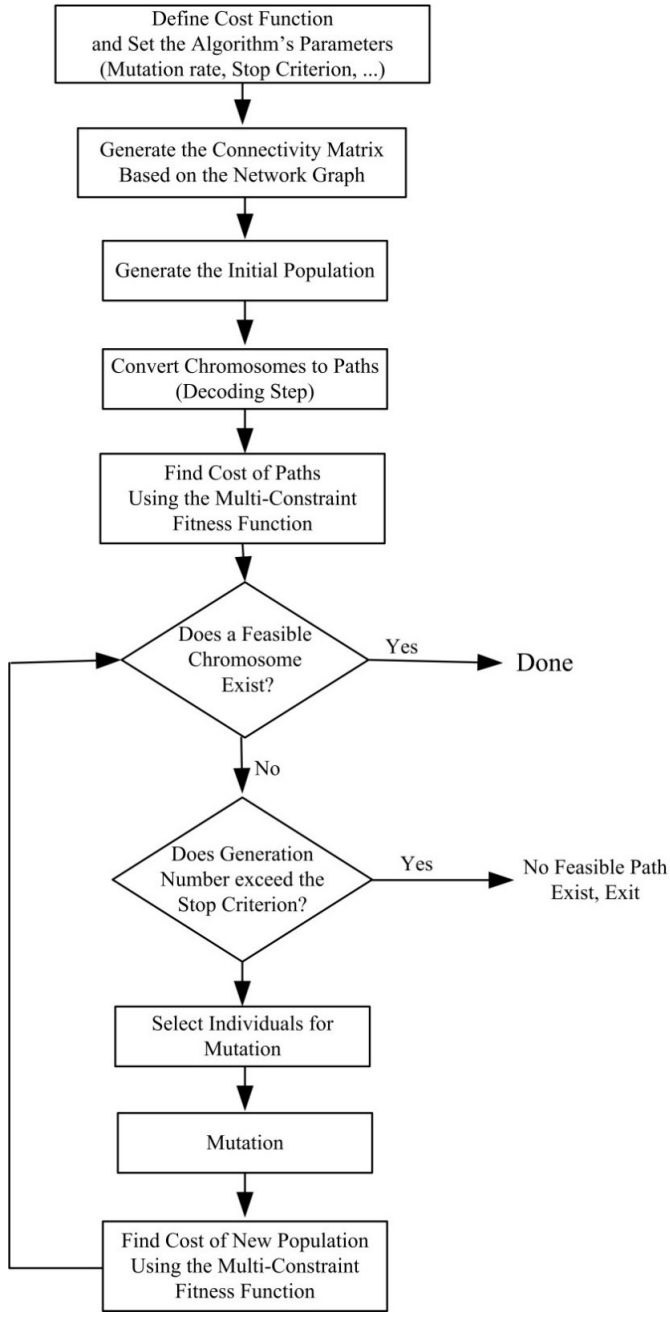

Figure 1. The proposed algorithm

15. Generate a new population by mutating some randomly selected genes of individuals
16. Repeat Stage 13 for the new generated population
17. Continue the algorithm until the termination conditions are met:
18. If the algorithm reached into a feasible solution
19. Then return the feasible path as the solution
20. Else, if the algorithm exceeds from the maximum number of generations
21. Then exit and return no feasible solution

By the end of this step, we obtained real potential paths, which can be evaluated using the fitness function.

Step 4: Computing Fitness Values. The proposed algorithm can tackle problems with graph links having two or more attributes but in this paper, we investigate a two-attribute one. Clearly, in order to satisfy the QoS requirements, two constraints are considered; cost and delay. Due to the flexible fitness function of the algorithm, they can be simply extended to more constraints and be used to optimize larger targets and problems. The associated attributes to each link are represented by C_{linkij} (for the cost of a link between nodes i and j) and D_{linkij} (for delay imposed by a link between nodes i and j). The delay and cost of a path from source to destination is the sum of costs and delays of the constitutive links of that path presented mathematically by the following equation:

$$C_{path} = \sum_{i=1}^{N} C(link_i) \tag{1}$$

where C_{path} is the sum of the values of a constraint through the path, $link_i$ and N represent the i^{th} link and total number of links in the path, respectively. The fitness function is defined so that it can fulfill the parameters of a QoS application for every path and distinguish whether the path has the capability of transmitting the application's packets or not. For instance, if the maximum permissible delay for voice transmitting application is 150 ms and the sum of links' delays is more than 150 ms, it means that the path is not appropriate for the application. Therefore, we can simply conclude that if the relation of path's attribute and its maximum permissible value for an application is more than 1, the path is not feasible for that application, i.e.:

$$Ratio(C) = \frac{C_{path}}{Per(C)} \tag{2}$$

where $Per(C)$ and $Ratio(C)$ are maximum permissible value and the relation of path's attribute with that, respectively. If the number of QoS constraints is more than 1, this calculation must be repeated for each of them. If all ratios are less than 1, the path is feasible and acceptable for the respected application. In this condition, the algorithm terminates and presents the feasible chromosome as the solution.

If there is no feasible chromosome in a generation, it would be checked whether the generation number exceeds the allowable limit or not. If so, the algorithm terminates and notifies that no feasible path has been found. Otherwise, the algorithm sorts the chromosomes based on their fitness. The product of mentioned ratios for all constraints would fit the chromosomes and hence their order in the sorted population. The justification of multiplication is that the importance of all constraints is identical and their impact on the total fitness must also be the same. The lesser the amount of product, the better the chromosome. This fact is represented mathematically in the following equation:

$$FF = \prod_{i=1}^{m} C_{i_{path}} \tag{3}$$

where FF is the chromosome's total fitness, m is the number of constraints (which is 2 here), and $C_{i_{path}}$ represents the i^{th} constraints value.

Step 5: Evolving the Population. As mentioned before, in this paper, producing the next generation is done using ES. In this method, the main phase of GAs i.e. crossover is omitted. This was done since this algorithm does not benefit well from the crossover operator. Since the genes are some random numbers which are not the real graph nodes, crossing over the chromosomes is nothing but mutating them. Therefore, implementing the crossover operator only imposes an overhead to the algorithm and mutation operator can thoroughly cover its function. Hence, to generate a new population some randomly selected genes of individuals are mutated. In detail, μ parents will be selected based on their fitness value and then they produce λ offspring. Only the top μ of λ offspring and μ parents are permitted to stay alive and enter the next generation.

Step 6: Re-computing the Fitness Values. Indeed this step is the repetition of step 4, for new population.

Step 7: Checking the Stopping Criterion. Two terms are considered as the algorithm's termination condition, reaching into a feasible solution or exceeding from the maximum number of generations. In the latter, the algorithm exits and returns no feasible solution.

3.3 Sample Graph

In this subsection, the proposed algorithm is illustrated using a typical graph with 8 nodes which is depicted in Figure 2. As mentioned before, two constraints are used to fulfill the QoS requirements; delay and cost. The links' attributes are also shown in Figure 2. Evaluation of the feasibility of source to destination paths are carried out by using these attributes. In this example, nodes 1 and 8 are supposed as source and destination nodes, respectively. Also, a multimedia application is intended to be transferred which tolerates utmost 50 units for both delay and cost constraints. Now we need the connectivity matrix of the graph. If there exists a link between

two nodes, the corresponding row and column element is set by 1, otherwise by 0. The connectivity matrix of Figure 2 is depicted in Figure 3.

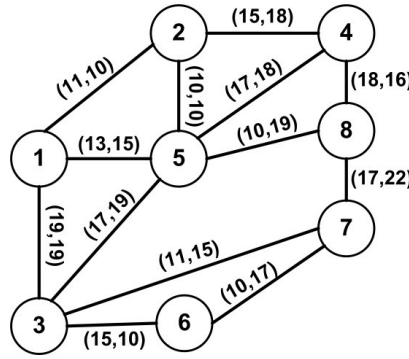

Figure 2. An 8 node network graph with links' attributes

	1	2	3	4	5	6	7	8
1	0	1	1	0	1	0	0	0
2	1	0	0	1	1	0	0	0
3	1	0	0	0	1	1	1	0
4	0	1	0	0	1	0	0	1
5	1	1	1	1	0	0	0	1
6	0	0	1	0	0	0	1	0
7	0	0	1	0	0	1	0	1
8	0	0	0	1	1	0	1	0

Figure 3. The connectivity matrix of Figure 2

This matrix will be used for step 3 of the algorithm, acquiring all possible paths from a node to the destination.

Our sample graph has 8 nodes, so the chromosomes will have 7 genes. The initial population individuals are generated randomly. Suppose one of them is as bellow:

i	1	2	3	4	5	6	7
value	3	4	2	2	5	4	1

For the first gene, i.e. $i = 1$, the gene value is equal to 3 and the adjacent nodes are $\{2, 3, 5\}$. So the variable nm is set to 3 (the number of adjacent nodes). The modulus resulted from dividing 3 by 3 is 0. It means that the node index is 0 and node 2 will be chosen. If one continues this procedure for other nodes, the following path would be obtained. The steps of this computation are summarized in Table 1.

$$1 \to 2 \to 4 \to 5 \to 3 \to 7 \to 6 \to X$$

This chromosome does not bring us to the destination, so the correction stage of step 3 should be run. This chromosome must be recovered from 6^{th} gene ($i = 6$). The formerly mentioned procedure is repeated and the results are presented in Table 2.

i	Adjacent Nodes	Gene Value	Chosen Path	
1	$\{2, 3, 5\}$	3	$34 \,\%\, 3 = 0$	2
2	$\{4, 5\}$	4	$4 \,\%\, 2 = 0$	4
3	$\{5, 8\}$	2	$24 \,\%\, 2 = 0$	5
4	$\{3, 8\}$	2	$24 \,\%\, 2 = 0$	3
5	$\{6, 7\}$	5	$54 \,\%\, 2 = 1$	7
6	$\{6, 8\}$	4	$44 \,\%\, 2 = 0$	6
7		1		X

Table 1. Chromosome to path conversion for our typical example

i	Adjacent Nodes	Gene Value	Chosen Path	
1	$\{2, 3, 5\}$	3	$34 \,\%\, 3 = 0$	2
2	$\{4, 5\}$	4	$44 \,\%\, 2 = 0$	4
3	$\{5, 8\}$	2	$24 \,\%\, 2 = 0$	5
4	$\{3, 8\}$	2	$24 \,\%\, 2 = 0$	3
5	$\{6, 7\}$	5	$54 \,\%\, 2 = 1$	7
6	$\{6, 8\}$	4	Next	8
7		1	Exit	

Table 2. Chromosome to path conversion for our typical example after correction

Now we have the following acceptable path:

$$1 \to 2 \to 4 \to 5 \to 3 \to 7 \to 8$$

Repeating this procedure for all chromosomes of the population leads the algorithm to some real routes from source to destination. Fitness of these paths can be obtained by running step 4. Here, we compute the fitness value for our example path:

$$C_{delay} = 86 \Rightarrow Ratio(delay) = 1.72$$
$$C_{cost} = 102 \Rightarrow Ration(cost) = 2.04$$

These numbers indicate that neither delay nor cost of the application are satisfied by this path. Let this chromosome being selected for the next generation and suppose a mutation occurred on its third gene as bellow:

$$gene(3) = 2 \Rightarrow 3$$

Now, the chromosome is converted to another one which directs the packets to the destination in a shorter path. The procedure of converting this chromosome into a solution presented in Table 3.

Computing the fitness function for the new path i.e. $1 \rightarrow 2 \rightarrow 4 \rightarrow 8$ results in the following numbers:

$$C_{delay} = 44 \Rightarrow Ratio(delay) = 0.88$$
$$C_{cost} = 44 \Rightarrow Ration(cost) = 0.88$$

i	Adjacent Nodes	Gene Value	Chosen Path	
1	$\{2,3,5\}$	3	$3\,\%\,3 = 0$	2
2	$\{4,5\}$	4	$4\,\%\,2 = 0$	4
3	$\{5,8\}$	3	$3\,\%\,2 = 1$	8
4		2	Exit	
5		5		
6		4		
7		1		

Table 3. Chromosome to path conversion after the mutation

Both constraints' values are lesser than 1 and it means that this path is feasible for the given application. The mutation process is depicted in Figure 4.

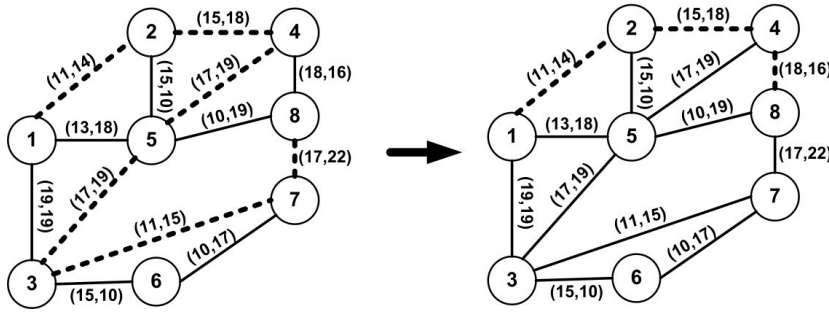

Figure 4. Offspring generated after applying mutation operator

3.4 The Proposed Method's Properties

- The time complexity of the algorithm is inconsiderable. It is because of our simple fitness function and using one evolutionary operator (i.e. mutation).
- The fitness function is so flexible that it can tackle any number of constraints, without any major change in the algorithm.
- The chromosome's lengths are fixed and equal.

- There is no need of any validation phase to check the existence of loops or invalid chromosomes.

4 SIMULATION RESULTS

In order to compare the proposed algorithm with four other algorithms, ARGA [13], ARGAQ [14], MURUGA [15] and TING [16], some simulations were done and their results are presented in this section.

In our experiments, we generated all graphs using Waxman random network [17]. The main feature of this network generator is that it distributes the nodes of the network uniformly and puts links based on probabilities. The distances between nodes determine these probabilities. Description about this network generator is out of the scope of this section and one can find the detailed information about it at [17]. In our simulations we assumed the Waxman parameters as $\alpha = 0.2$ and $\beta = 0.4$. Our multi-constraint routing problems are subjected to two QoS metrics; TD (Transmission Delay) and TSR (Transmission Success Ratio), which here after will be shown by d and r, respectively. For similarity with one of our main references, d and r are randomly chosen in a uniform distribution $[0, 100]$ and $[0, 1]$, respectively [16]. Also, the source and destination nodes are taken randomly, and the constraints (d and r) of each routing demand are selected in a uniform distribution $[200, 400]$ and $[0.5, 1]$, respectively.

In this section, three different experiments are presented to prove the efficiency of the proposed method.

The size of all network graphs in our simulations was 20 nodes. In order to perform fair comparison between the mentioned algorithms, according to TING the population size was equal to the number of graph size.

In this experiment, the elitism of the chromosome with rank 1 in the population is evaluated. Based on the value returned by fitness function, each chromosome obtains a grade, which will be used to rank it in the population. The chromosome with lowest cost will be the best individual and seizes the first rank. The second finest one holds the second rank and so on. The solutions of all algorithms are compared with the results obtained by H_MCOP algorithm [18], which is taken as a criterion. It must be noted that H_MCOP is an approximation algorithm that offers a non-optimal solution to the MCP problems which are otherwise NP-complete for an optimal solution algorithm. H_MCOP's major aspect is that it shortcuts some of complexities using heuristics. The result of this comparison is shown in Figure 5.

According to Figure 5 the proposed algorithm outperforms the others so that ours reached a feasible path at generation 4, whereas TING and MURUGA both found a solution by getting to generation 5, ARGAQ took 9 generations to reach a feasible route and ARGA needs too many generations. Therefore, it can be concluded that the proposed algorithm has better convergence performance in comparison with others, especially ARGA and ARGAQ. Regarding the simulation re-

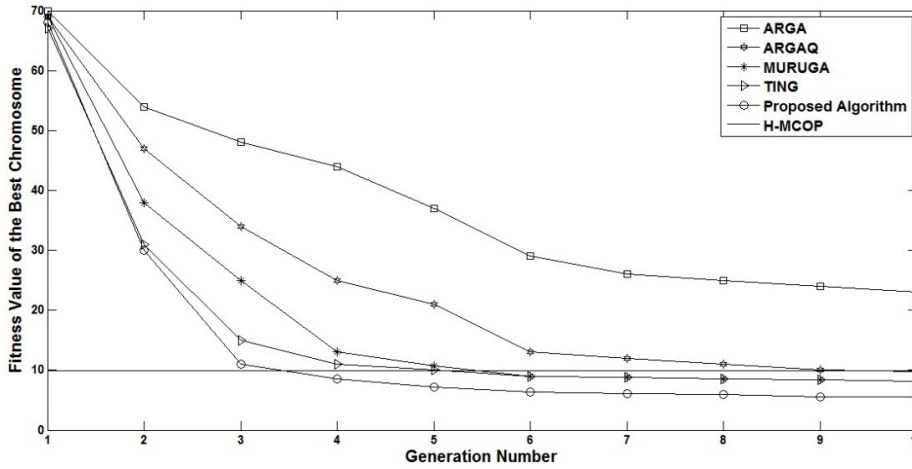

Figure 5. Fitness value of the chromosome with rank 1 versus generation number

sults, the number of generations needed to reach a feasible solution in our algorithm is averagely 3.65 and 9.94 times lesser than ARGAQ and ARGA, respectively.

Setting the maximum number of generations to 10 and running the simulation 20 times, we examined the Success Ratio (SR) of our algorithm to find a feasible path. In fact, we measured the success ratio of our proposed method for a number of routing demands. SR is dividing the number of founded feasible paths by the number of all routing demands. Here again, the number of population individuals are taken as many times as the number of graph nodes. It is clear that an algorithm is said to have better performance when having a higher SR. The results of this experiment were compared with that of [16] and are shown as a diagram in Figure 6.

Figure 6 shows that our algorithm outperforms others, especially it has much better success ratio than ARGA and, ARGAQ. Regarding to the simulation results, the success ratio of the proposed algorithm is better than MURUGA and TING by 1.56 % in average, and it is 1.07 and 2.15 times more successful than ARGA and ARGAQ, respectively.

For the next experiment, different sizes of networks were studied. The number of nodes was changed from 20 to 50. Again, the number of population individuals was taken as equal as the number of the graph nodes. The simulations repeated 500 times for each graph size with different random networks. Finally, the maximum number of generations was considered 10. Since the number of generations does not necessarily indicate the fastness of the algorithm, another criterion must be observed. A given algorithm in comparison with another one might require more genetic steps and operations which are complex and time consuming. Therefore, measuring the execution time of the algorithms is a more reliable criterion for comparing their performance. The first objective of this experiment was evaluating the computation

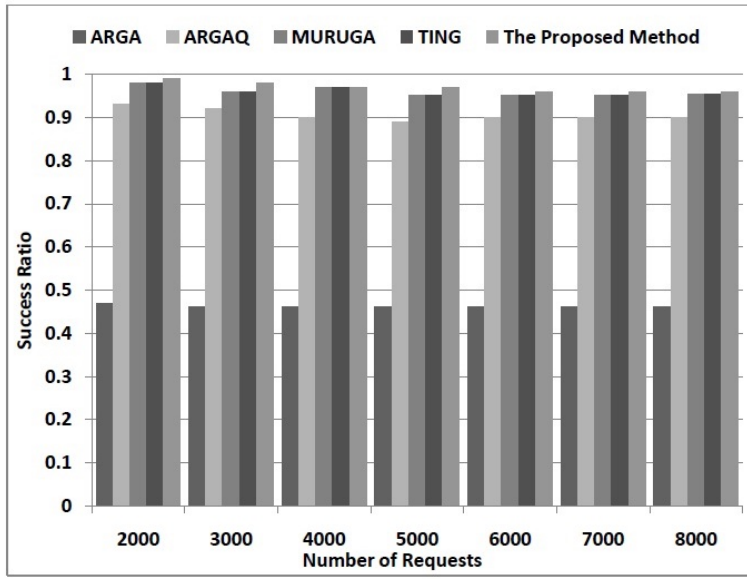

Figure 6. Percentage of successful routing versus the number of routing demands

time needed to find a feasible path by the algorithms. The result of this experiment is shown in Figure 7.

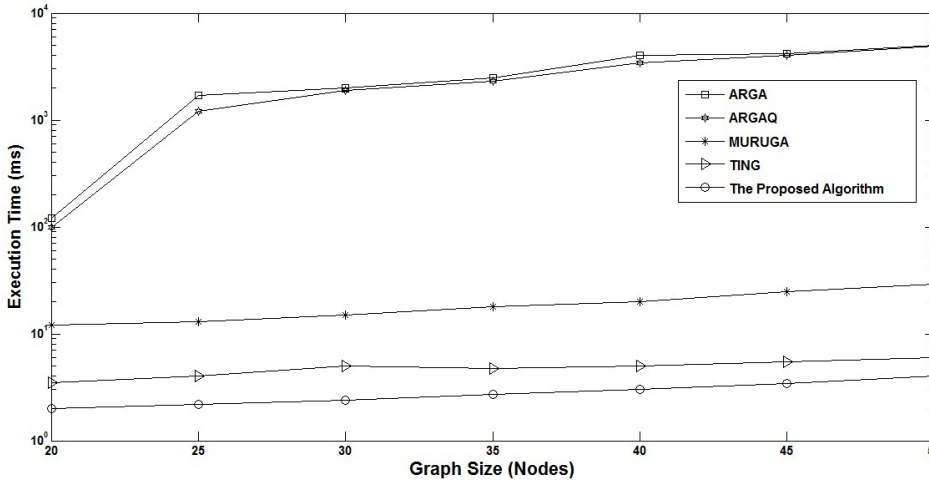

Figure 7. Execution time of the algorithms

As can be seen from Figure 7, the proposed algorithm performs much better in terms of execution time. Moreover, our algorithm shows a scalable behavior in various graph sizes. The main reason for the superiority of the proposed method is that it does not use time-consuming evolutionary operators and furthermore it does not include any repairing stages. But the other algorithms, except TING, check the validity of chromosomes' genes at the end of each generation which imposes a considerable time overhead. Moreover, by increasing the size of graph, the time needed to perform their calculations actually grows. Numerically, ours is 2.6, 11.3, 868 and 922 times faster than TING, MURUGA, ARGAQ and ARGA, respectively.

5 CONCLUSION

Real-time multimedia applications usage is growing widely. However, solving the problem of finding a path in real networks, which satisfies the QoS requirements of the application, is so challenging that has motivated many researchers to develop new routing algorithms. Lately, EAs are used vastly to tackle the issue. The existing evolutionary methods have three main drawbacks; considering single constraint instead of multi-constraint; using complex evolutionary operators that increases the algorithm run time; and requiring extra steps to correct the invalid individuals at every stage of the algorithm. In this paper, a multi-constraint QoS routing algorithm is developed based on ES, which finds the feasible solutions very fast. The main reason behind this convergence speed is its novel gene decoding mechanism. To prove the efficiency of the proposed algorithm, it has been simulated and compared with some of other existing methods. The results show that our algorithm outperforms the other two in terms of algorithm run time and success ratio. In fact, it is in average 2.6 and 11.3 times faster than our two-best compared algorithms and its success ratio is much better than all other compared algorithms. As a future work, one can extend the comparison metrics, such as Dynamic Adaptability Degree (DAD), to prove its efficiency as much as possible.

REFERENCES

[1] Roy, A.—Das, S. K.: QM2RP: A QoS-Based Mobile Multicast Routing Protocol Using Multi-Objective Genetic Algorithm. Wireless Network Journal, Vol. 10, 2004, pp. 271–286, doi: 10.1023/b:wine.0000023861.10684.f1.

[2] Ahn, C. W.—Ramakrishna, R. S.: A Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Populations. IEEE Transactions on Evolutionary Computation, Vol. 6, 2002, pp. 566–579, doi: 10.1109/TEVC.2002.804323.

[3] Ni, M.: A Simple and Fast Algorithm for Restricted Shortest Path Problem. Proceedings of the International IEEE Conference on Computational Sciences and Optimization, 2011, pp. 99–102, doi: 10.1109/cso.2011.57.

[4] Vasilakos, A.—Ricudis, C.—Anagnostakis, K.—Pedryca, W.—Pitsillides, A.: Evolutionary-Fuzzy Prediction for Strategic QoS Routing in Broadband

Networks. Proceedings of the IEEE World Congress on Computational Intelligence and Fuzzy Systems, Vol. 2, 1998, pp. 1488–1493, 10.1109/fuzzy.1998.686339.

[5] Li, P.—Guo, S.—Yu, S.—Vasilakos, A. V.: CodePipe: An Opportunistic Feeding and Routing Protocol for Reliable Multicast with Pipelined Network Coding. Proceedings of the IEEE Conference on INFOCOM, 2012, pp. 100–108.

[6] Zeng, Y.—Kai, X.—Deshl, L.—Athanasios, V.: Directional Routing and Scheduling for Green Vehicular Delay Tolerant Networks. Wireless Networks, Vol. 19, 2013, No. 2, pp. 161–173, doi: 10.1007/s11276-012-0457-9.

[7] Atlasis, A. F.—Loukas, N. H.—Vasilakos, A. V.: The Use of Learning Algorithms in ATM Networks Call Admission Control Problem: A Methodology. Journal of Computer Networks, Vol. 34, 2000, No. 3, pp. 341–353, doi: 10.1016/s1389-1286(00)00090-6.

[8] Jain, S.—Sharma, J. D.: Tree Structured Encoding Based Multi-Objective Multicast Routing Algorithm. International Journal of Physical Sciences, Vol. 7, 2012, pp. 1622–1632.

[9] Barolli, L.—Koyama, A.—Sawada, H.—Suganuma, T.—Shiratori, N.: A New QoS Routing Approach for Multimedia Applications Based on Genetic Algorithms. Proceedings of the International IEEE Conference on Cyber Worlds, 2002, pp. 289–295, doi: 10.1109/cw.2002.1180892.

[10] Yena, Y. S.—Chao, H. C.—Changd, R. S.—Vasiakos, A.: Flooding-Limited and Multi-Constrained QoS Multicast Routing Based on the Genetic Algorithm for MANETs. Mathematical and Computer Modeling Journal, Vol. 53, 2011, pp. 2238–2250, doi: 10.1016/j.mcm.2010.10.008.

[11] Chitra, C.—Subbaraj, P.: A Non-Dominated Sorting Genetic Algorithm Solution for Shortest Path Routing Problem in Computer Networks. Expert Systems with Applications Journal, Vol. 39, 2012, pp. 1518–1525, doi: 10.1016/j.eswa.2011.08.044.

[12] Munetomo, M.—Takay, Y.—Sato, Y.: An Adaptive Routing Algorithm with Load Balancing by a Genetic Algorithm. IPSJ Journal, 1998, pp. 219–227.

[13] Barolli, L.—Koyama, A.—Suganuma, T.—Shiratori, N.: A Genetic Algorithm Based QoS Routing Method for Multimedia Communications over High-Speed Networks. IPSJ Journal, 2003, pp. 544–552.

[14] Barolli, L.—Koyama, A.—Matsumoto, K.—Apduhan, B. O.: A GA-Based Multi-Purpose Optimization Algorithm for QoS Routing. Proceedings of the International IEEE Conference on Advanced Information Networking and Applications, 2004, pp. 23–28.

[15] Leela, R.—Thanulekshmi, N.—Selvakumar, S.: Multi-Constraint QoS Unicast Routing Using Genetic Algorithm (MURUGA). Applied Soft Computing Journal, Vol. 11, 2011, pp. 1753–1761, doi: 10.1016/j.asoc.2010.05.018.

[16] Ting, L.—Zhu, J.: A Genetic Algorithm for Finding a Path Subject to Two Constraints. Applied Soft Computing Journal, Vol. 13, 2013, pp. 891–898, doi: 10.1016/j.asoc.2012.10.018.

[17] Waxman, B. M.: Routing of Multipoint Connection. IEEE Journal on Selected Areas in Communications, Vol. 6, 1998, pp. 1617–1622, doi: 10.1109/49.12889.

[18] Korkmaz, T.—Krunz, M.: Multi-Constrained Optimal Path Selection. Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, 2001, pp. 834–843, doi: 10.1109/infcom.2001.916274.

Hadi Shahriar Shahhoseini received his B.Sc. degree in electrical engineering from University of Tehran, in 1990, M.Sc. degree in electrical engineering from Azad University of Tehran in 1994, and Ph.D. degree in electrical engineering from Iran University of Science and Technology, in 1999. He is Associate Professor in the Electrical Engineering Department at Iran University of Science and Technology. His areas of research include networking, supercomputing and reconfigurable computing. He has published more than 150 papers from his research works in scientific journals and conference proceedings. He is an executive committee member of IEEE TCSC and serves IEEE TCSC as regional coordinator in Middle-East Countries.

Samaneh Torkzadeh received her B.Sc. degree in electrical engineering from the Department of Electrical Engineering, Shahed University, Tehran, Iran in 2010. In 2010, she entered Iran University of Science and Technology to pursue her M.Sc. in electrical engineering. She also cooperated with IUST Super Computing and Networking (SCaN) research laboratory during her M.Sc. studies. Her main research interests include networking and optimization by developing intelligent algorithms.