Soft Computing (2019) 23:693–706
https://doi.org/10.1007/s00500-018-3018-z

METHODOLOGIES AND APPLICATION

# Multi-constraint QoS routing using a customized lightweight evolutionary strategy

Samaneh Torkzadeh[1] · Hadi Soltanizadeh[1] · Ali Asghar Orouji[1]

Published online: 18 January 2018


**Abstract**
The ever-increasing transmitting real-time multimedia applications such as VoIP and video conference through the Internet require developing routings methods which guarantee quality of service (QoS) according to the needs of these applications. For these types of applications, a fundamental issue is how to find a feasible path that satisfies multiple constraints. This problem which is known as multi-constraint QoS routing is an NP-complete one, and many research has been devoted to solving it. However, there are still many gaps especially in terms of complexity and speed of the algorithms that must be bridged in order for these methods to be practical. In this regard, in this paper, a novel multi-constraint QoS routing algorithm based on evolutionary strategies is proposed which is lightweight and finds feasible solutions in a very short time. The main reason behind these features is due to the proposed inventive gene decoding mechanism that makes the algorithm needless of any complex evolutionary operators and validation phases. Moreover, a cost function is developed for evaluating the fitness of the potential solutions, which is so flexible for using for as many constraints as required. Therefore, the algorithm is able to search the solution space, no matter how big they are, efficiently and quickly. Simulation results on different network topologies and packet traffics show that our method outperforms competitor algorithms in terms of run time and success ratio, and it is more reliable in different network and traffic scenarios.

**Keywords** Evolutionary strategy · Genetic algorithm · Multi-constraint · QoS routing · Computer networks

## 1 Introduction

Since the Internet is an interwoven network of interconnected networks all around the world, data packets of an application may follow different paths to reach their destination. Nevertheless, due to the fair sharing of Internet resources such as bandwidth and switch buffers between the packets of different sessions, the mentioned issue does not violate the requirements of some applications such as email and file transfer. However, this shared network cannot sustain real-time multimedia applications such as video conference, video on demand (VoD), Internet telephone and audio on demand (AoD) which need satisfying multi-constraints like jitter, cost and packet loss (Roy and Das 2004; Feng et al. 2014; Martín et al. 2014; Jarosalv at al. 2015). There are two main reasons to prove this claim; first, the end-to-end delivery performance is not ensured because Internet does not support the resource reservation; second, data packets may experience unexpected delays and get to the destination untidily, which is undesirable for real-time applications (Karthikeyan and Baskar 2015; Ahn and Ramakrishna 2002). Therefore, it is necessary for network service providers to ensure QoS routing. It is worth noting that QoS refers to a set of performance metrics that must be satisfied for a given requested service, and QoS routing is finding a path, which fulfills the specified QoS requirements (Ni 2011). Those mentioned complexities make the multi-constraint routing problem, an NP-complete one (Karthikeyan et al. 2013; Alejandro et al. 2015). In this regard in recent years, many studies have addressed this problem, among which those using evolutionary algorithms (EAs) are in the center of attention (Yin et al. 2014; Benlai and Yu 2015; Vasilakos et al. 1998; Li et al. 2012; Zeng et al. 2013;

Communicated by V. Loia.

✉ Hadi Soltanizadeh
h_soltanizadeh@semnan.ac.ir

Samaneh Torkzadeh
samaneh_torkzadeh@semnan.ac.ir

Ali Asghar Orouji
aliaorouji@semnan.ac.ir

[1] Electrical and Computer Engineering Department, Semnan University, Semnan, Iran

Jain and Sharma 2012; Barolli et al. 2002). Generally, there are too many EAs which can be used to solve NP-complete problems like ours (Xue et al. 2017; Liu et al. 2016; Gu and Sheng 2016; Yuan et al. 2017; Rong et al. 2017; Gu et al. 2016). For example, SABC-GB, an intelligent swarm algorithm for global optimization problems, is an alternative of ant bee colony (ABC) algorithm, which fixes its flaws such as insufficient solution search space and poor exploitation performance by using a novel self-adaptive mechanism (Xue et al. 2017). Another EA-based method is presented in Deng et al. (2017a, b), in which an improved adaptive particle swarm optimization (PSO) is used to solve a multi-objective optimization problem. This algorithm is based on making full use of the advantages of alpha-stable distribution and dynamic fractional calculus. The dynamic fractional calculus is used to reflect the trajectory information of particle updating in order to improve the convergence speed. On the other side, the alpha-stable distribution theory is used to replace the uniform distribution in order to escape from the local minima in a certain probability and improve the global search ability. Therefore, this method could improve the convergence time and search ability of PSO simultaneously and as a result could solve the desired problem more efficiently. Another interesting algorithm named MGACACO is proposed by Deng et al. in which the authors introduced the chaotic optimization method, multi-population collaborative strategy and adaptive control parameters into genetic algorithm (GA) and ant colony optimization (ACO) in order to overcome the deficiencies of weak local search ability in GA and slow global convergence speed in ACO in solving complex optimization problems (Deng et al. 2017b). The proposed algorithm makes use of the exploration capability of GA and stochastic capability of ACO algorithm to search the search spaces of optimization problems faster and more accurately, and without falling into the local extremum. Other examples are TS-NSGA-II, a time and space optimized NSGA-II algorithm (Liu et al. 2016), and PD-LBP, a decentralized belief propagation-based method which reduces the search space of the given optimization problem and enables belief propagation to be operated in parallel (Kong et al. 2016).

The most important step to solve the problem of this paper by using EAs is to determine which one is more appropriate and most fit for the problem. To this end, we should consider the requirements of our problem, i.e., routing or finding a feasible path from a given source to a determined destination in a large network. The main goal of all routing algorithms is selecting a path adaptively, intelligently and flexibly. Most existing EAs have fundamental shortcomings to solve this problem which can be mainly classified into the four following categories:

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

To tackle the mentioned challenges, among all existing EAs, evolutionary strategy (ES) is selected. ES, the approach of derivative-free hill climbing in complex search spaces to maximize a fitness function, has many advantages over its alternatives some of which are discussed below. While it is claimed by some researchers that linear convergence is the best one that can be achieved by EAs (Voigt et al. 1995), the authors in Back et al. (1993) have shown that ESs are able to achieve this convergence order. In addition to a linear order of convergence which assures a sufficiently large velocity, the property of global convergence is the second important requirement for any global optimization algorithm. By global optimization, we mean "given unlimited time, the algorithm has to find a globally optimal solution with probability one" (Zhang et al. 2016). In Rudolph (1994), it is demonstrated that ESs globally converge with probability one. Another advantage of ESs is that they certainly benefit from a larger degree of freedom by working with $n$ different self-adaptive standard deviations per individual in contrast to a single mutation rate in some of its alternatives such as GAs, which is predefined and is constant over the complete evolution process (Bäck 1995). The most important advantage of ESs that we exploited well in our approach is that in contrast to its other alternatives such as GAs, recombination in ESs is a more flexible operator, incorporating 2 to $\mu$ parents for the creation of a single descendant (Salimans et al. 2017; Bäck 1995).

The main feature of the proposed algorithm is its mode of gene decoding of chromosomes that makes it needless to use complex evolutionary operators and extra phases such as checking and repairing. These capabilities result in fast convergence of the algorithm and a reduction in the run time. The main contributions of the paper can be summarized as follows:

- Developing a routing algorithm based on EAs which satisfies multimedia application's constraints while preserving functionality and simplicity
- Presenting a novel gene decoding mechanism which avoids loop existence in the paths and markedly reduces the complexity of evolutionary operators
- 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
- 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 successful methods. The rest of this paper is organized as follows. Next section reviews some recent related contributions to the field. In Sect. 3, the proposed algorithm is described and an example is presented to illustrate it. The simulation results are offered in Sect. 4. Finally, Sect. 5 concludes the paper.

## 2 Related works

In recent years, researchers have proposed many QoS routing algorithms. In this section, we review some of them and discuss their advantages and drawbacks.

The authors in Jain and Sharma (2012) have proposed a multi-objective multicast routing GA which is based on topological assisted tree structured encoding. In order to form the initial chromosomes, this approach generates a combination of random routes from the multicast group destination nodes to the source node. This method has two main drawbacks which reduce its effectiveness to be considered as a favorable routing algorithm; first, to form its initial population, it uses a complex and costly process, and second, to guarantee the feasibility and validity of the chromosomes at the end of each generation, it uses complex genetic operators which impose extra time and computation overhead.

Another GA-based multi-constrained QoS multicast routing method is proposed in Yena et al. (2011). In order to decrease the search space, this method applies a mechanism named flooding-limited which eliminates the probability of selecting undesirable nodes and links. Although this approach saves time by the proposed search space shrinkage method, it still suffers from two shortcomings. The imprecise fitness function used in the algorithm deteriorates the quality of the generated routs. Moreover, its heavy reliance on the values of the genetic operators undermines its performance in different scenarios.

Chitra and Subbaraj (2012) proposed a method called NSGA, which uses an elitist multi-objective EA to solve the dynamic shortest path routing problem in computer networks. Since their proposed method uses a priority-based encoding scheme to produce initial population, it overcomes some of the challenges mentioned about the previous algorithms, but it still lasts too many generations to converge into a feasible solution. Munetomo et al. (1998) proposed a QoS routing algorithm based on GA, in which the genes order in chromosomes is the same as the order of their respective nodes in the path (Feng et al. 2014). 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 considers only a single constraint (which is delay time) to determine 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 unwanted chromosomes. Therefore in general, this algorithm is too costly.

In order to overcome the mentioned problems in the above algorithms, an algorithm called ARGA Barolli et al. (2004) is proposed that fixes the size of chromosomes by employing a novel gene coding method. In ARGA, network and chromosome genes are described as tree and tree junction, respectively, and as a result, the method guarantees loop-free paths. By eliminating 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 determine QoS routes.

While none of the above algorithms is able to handle more than one constraint, QoS routing requires considering multi-constraints. Therefore, another algorithm named ARGAQ Barolli et al. (2004) was proposed to upgrade the previous ones. To address more than one constraint, in this method, the relation of two constraints [delay time and transmission success ratio (TSR)] is used. 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, although it is still an approximation of QoS routing.

Leela et al. (2011) developed a GA-based algorithm, which directly considers constraints in calculating routs which satisfies given QoS. In fact, the authors proposed a method called Multi-constraint QoS Unicast Routing Using GA (MURUGA), 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. This issue results in a considerable time and cost overhead.

Ting and Zhu (2013) have resolved the previous method's drawback by proposing a gene structure (GS) for encoding the MCR problem and presenting GSA algorithm to generate the GS. By using this technique, they guarantee that there would be no invalid chromosome in the population. As far as known to the authors, this method is the most efficient algorithm presented to solve our problem in terms of quality of the solutions and run time. Therefore, its results are taken as the most important criterion to evaluate the efficiency of our proposed method in the experiments.

In Liu et al. (2015), the authors proposed a multi-constrained routing scheme named MFA-RA using mean field annealing which exploits a set of deterministic equations to replace the stochastic process in simulated annealing (SA). The simulation results show that their proposed algo-

rithm has a fast convergence rate and can effectively escape the local optimum in comparison with some evolutionary-based competitors. Although MFA-RA reaches equilibrium at each temperature much faster than SA, it still needs many computations to calculate MFA equations and its updating. Moreover, according to the results presented in the paper the success ratio of the proposed algorithm is not well satisfying.

Bilbao et al. have applied the coral reefs optimization (CRO) and the Firey algorithm, two of the latest bio-inspired meta-heuristic techniques, to solve multiply-constraint network routing. Results obtained from Monte Carlo simulations over synthetic network instances show that the performance of these two algorithms is comparable in terms of convergence speed and statistical significance. However, their performance against other competitors is under question.

In Abdullah et al. (2017), another GA-based multi-constraint routing algorithm named IRAGA is proposed in which by applying a novel crossover-refining operation genetic variety of the chromosomes is increased and the ability of searching is improved. Although the rate of convergence between the chromosomes is increased, due to its the weak exploitation, the quality of the generated solutions by this algorithm is not satisfying.

Recently, energy is raised as an important constraint for QoS routing in the networks, especially in wireless ones. As an example in this regard, an ant colony optimization (ACO)-based solution for QoS-aware energy efficient routing is presented in Zaheeruddin et al. (2017). More clearly, the authors have extended two algorithms based on metaphor of swarm intelligence for finding an energy efficient routing tree satisfying the QoS guarantees. The main feature of their algorithm is that it does not increase the number of forwarding nodes in the routing tree which negatively impacts the QoS in terms of propagation delay, delay jitter, packet loss, etc. However, the power optimization does not exactly lie in the scope of this paper and it is presented to introduce the future trend of the subject.

Having discussed the above methods, the current paper intends to resolve some of their shortcomings, dealing with multi-constraints, removing additional stages, accelerating the convergence and 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 is described. First of all, it is necessary to explain the underlying network. In order to model the network, an undirected graph is used. This graph is denoted by $G(N, E, e)$ in which $N$, $E$ and $e$ represent the set of nodes (vertices), set of links (edges) and constraints' matrix of the network, respectively.

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

- Chromosome is a loop-free path from a given source to a given destination each gene of a chromosome represents a node of graph
- Population consists of a determined number of different chromosomes, and this number is highly dependent on the graph structure
- Fitness function evaluates the feasibility of a chromosome to see whether it satisfies the QoS constraints as well as determining its rank among other members of the population

The steps of the proposed algorithm are as follows:

- Generate the initial population, take some random paths from the source to the destination
- Evaluate the fitness of each chromosome using the proposed fitness function
- Sort the population individuals based on the values obtained in the previous steps
- Stop the algorithm whenever the first rank individual's fitness becomes less than 1, otherwise go to the next step
- Generate a new population using ES operator
- Continue the algorithm until the stopping criterion is met

### 3.1 General overview of the proposed algorithm

The proposed algorithm uses an inventive 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 be converged into a feasible path in a very short time. Our proposed fitness function decides whether a path is feasible or not. If it is 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 algorithm, unlike GA, no crossover operator is applied and only mutation operator is used to evolve the population. More specifically, $\mu$ parents will be selected based on their fitness value and then they produce $\lambda$ offspring. Only the top $\mu$ of $\lambda$ offspring and $\mu$ parents are permitted to stay alive and enter the next generation. Using this kind of evolutionary

operation, i.e., mutation, efficiently helps in substituting the unfavorable paths with more new appropriate paths which satisfy the desired requirements of a given application. The main feature of the proposed method, which is due to its particular gene decoding mechanism, is verified through some simulation experiments. The flowchart of the proposed algorithm is shown in Fig. 1.

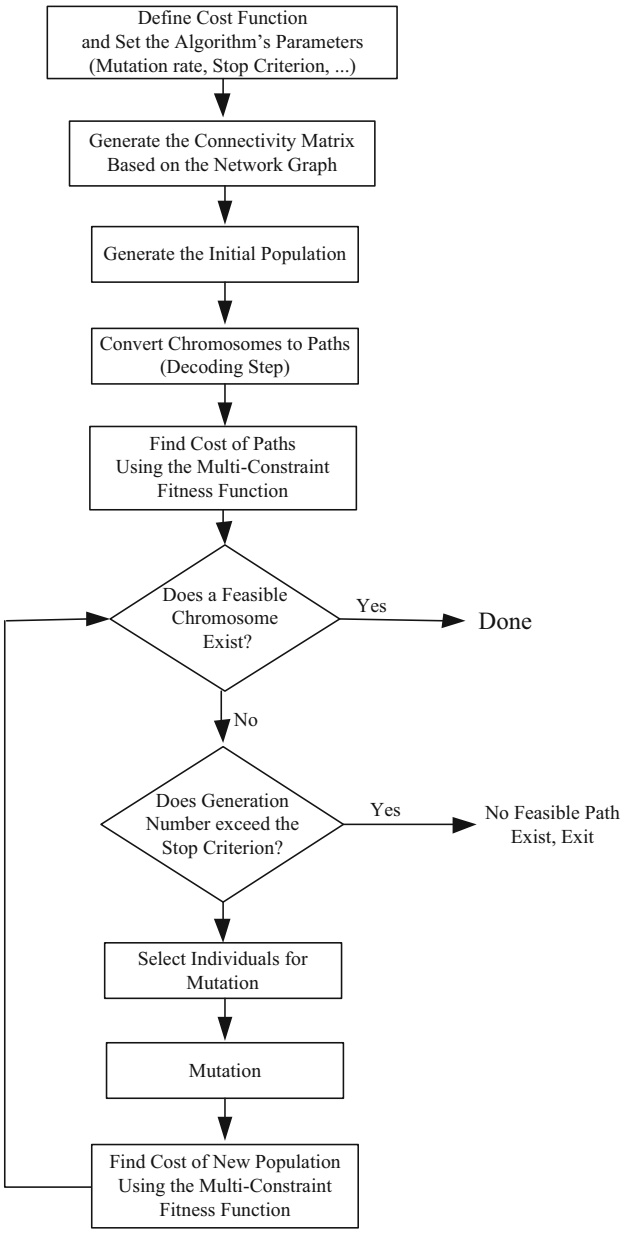


**Fig. 1** Flowchart of the proposed algorithm

## 3.2 The algorithm's procedure

In this subsection, we will explain the steps of the proposed algorithm in detail, which is also briefly presented in Algorithm 1. Generally, it has 7 steps described separately 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 length of all chromosomes is fixed and equal. This property simplifies the algorithm and its implementation. Each gene includes a number between 1 and $N-1$. These numbers do not have any direct relation to real paths from the source to the 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 generation of any invalid individuals in the population. Therefore, there is no need to use extra phases for validation and repairing unacceptable solutions.

*Step 2: Generating the Initial Population;* as mentioned in the previous step, chromosome genes can get a number between 1 and the number of nodes $-1$. To produce the initial population, a number in the mentioned range will be assigned to each chromosome. In 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 which is explained in the next step.

*Step 3: Chromosome to Path Conversion:* the main step in every evolutionary algorithm is how to design a feasible solu-

Algorithm 1 **The proposed algorithm**

1: Set chromosomes length equal to ($N$-1).
2:Generate chromosomes ($C$) with random genes value between [$i$ = 1 to $N$-1].
3: **for** $j$ = 1 to $C$ **do**
4: Set the source node as the first node.
5: Determine selected node's neighbors (*nm*)
6: Determine next node index by [(gene value) mod (*nm*)]
7: Repeat 5 & 6 until reaching into the destination node.
8: **if** Didn't touch the destination node **then**
9: Starting from the last gene.
10: **if** There would be no path to the destination **then**
11: Decrement i and Select the next node.
12: **end if**
13: **end if**
14: **end for**
15: Evaluate the decoded paths using the fittness function.
16: Generate a new population by mutating.
17: Repeat 15 for the new population.
18:**if** The algorithm reached into a feasible solution **then**
19: Return the feasible path as the solution.
20:**else if** Generation number exceed the stop criterion **then**
21: Exit and return no feasible solution.
22: **end if**

tion structure (in our case, chromosome structure) so that it contains all parameters of the given problem. To this end, two different approaches can be exploited: direct approach and indirect approach. In the former which is used by most algorithms, all optimization parameters of the problem are directly included in the potential solutions. The main drawback of these types of representation is that the order of genes may be disturbed in the evolving phases, and as a result, infeasible solutions would be generated. In this situation, using heuristic genetic operator mechanisms which ensure the feasibility of solutions is inevitable. These types of mechanisms not only complicate the algorithm's components, but also add an additional stage for non-randomly generating the initial population. In indirect approach, a string of genes which do not have any direct information about the problem's parameters forms the chromosome structure and then turns into a feasible solution using a decoding function. Therefore, there is no need for any repair phases to correct invalid chromosomes, and crossover and mutation operators would be simplified. In this paper, the indirect approach is adopted; thus, a special decoding function which is customized for our problem is presented. By adopting this approach, actually we try to separate the search space and the solution space in our problem to improve the performance. The motivation for applying this approach is that our QoS routing problem is a constrained optimization problem—not only our proposed ES algorithm should generate one or more real paths satisfying the QoS parameters in a short time, it is also forced to allow only such paths to be generated which are feasible. This step decodes the individuals into some potential paths from the source to the destination and has six stages, which are explained bellow:

1. The source node is always the first node, and the following algorithm is run for the $i$th node.
2. The adjacent nodes of the current node which have a way to the destination and are not traversed yet are determined, and their numbers are 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 all genes are investigated, or until there is no adjacent node to check or until the destination node is obtained.
5. If the algorithm does not reach the destination node by the end of the previous stage, this chromosome goes 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.

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. However, in this paper, we investigated 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 $C_{\text{Link}\,ij}$ (for the cost of a link between nodes $i$ and $j$) and $D_{\text{Link}\,ij}$ (for the delay imposed by a link between nodes $i$ and $j$). The delay and cost of a path from source to destination are the sum of costs and delays of the constitutive links of that path which is presented mathematically in Eq. (1):

$$C_{\text{path}} = \sum_{i=1}^{N} C\,(\text{Link}_i) \tag{1}$$

$C_{\text{path}}$ is the sum of the values of a constraint through the path; $\text{Link}_i$ and $N$ represent the $i$th link and total number of links in the path, respectively. The fitness function is defined so that 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 are 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, that path is not feasible for that application. In this regard, Eq. (2) is calculated for each attribute:

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

$\text{Per}\,(C)$ and $\text{Ratio}\,(C)$ are maximum permissible value and the relation of path's attribute to 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 will be checked whether the generation number exceeds the allowable limit or not. If so, the algorithm terminates and notifies that no feasible path is found. Otherwise, the algorithm sorts the chromosomes based on their fitness. The product of the mentioned ratios for all constraints should fit the chromosomes and therefore their order in the sorted population. The justification for 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 Eq. (3):

$$\mathrm{FF} = \prod_{i=1}^{m} \mathrm{Ratio}\,(C_{i_{\mathrm{path}}}) \tag{3}$$

FF is the chromosome's total fitness, $m$ is the number of constraints (which is 2 here), and $C_{i_{\mathrm{path}}}$ 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, and the reason is that this algorithm does not well benefit from 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. More precisely, $\mu$ parents will be selected based on their fitness value and then they produce $\lambda$ offspring. Only the top $\mu$ of $\lambda$ offspring and $\mu$ parents are permitted to live and enter the next generation.

*Step 6: Re-computing the Fitness Values;* indeed, this step is the repetition of step 4, to generate 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 exemplified using a typical graph with 8 nodes which is depicted in Fig. 2. As mentioned before, two constraints are used to fulfill the QoS requirements: delay and cost. The links' attributes are also shown in Fig. 2. The evaluation of the feasibility of source to destination paths is carried out by using these attributes. In this example, nodes 1 and 8 are supposed as source and destination nodes, respectively.

In addition, a multimedia application is intended to be transferred which tolerates uttermost 50 units for both delay and cost constraints. Now, the connectivity matrix of the graph must be generated. If there exists a link between two nodes, the corresponding row and column element is set to 1, otherwise to 0. The corresponding connectivity matrix of Fig. 2 is depicted in Fig. 3.

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

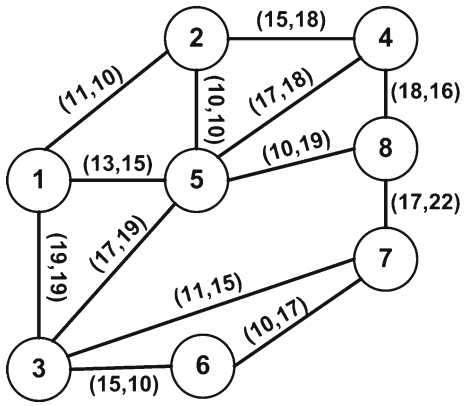


**Fig. 2** An 8 node network graph with links' attributes

| | 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 |

**Fig. 3** Connectivity matrix of the figure

**Table 1** Randomly generated chromosomes for the sample graph of Fig. 2

| I | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Value | 3 | 4 | 2 | 2 | 5 | 4 | 1 |

Our sample graph has 8 nodes, so each chromosome will have 7 genes. The individuals of the initial population are generated randomly. We suppose one of them is as Table 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 (i.e., 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 will be obtained. The steps of this computation are summarized in Table 2.

$$1 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 3 \rightarrow 7 \rightarrow 6 \rightarrow X \tag{4}$$

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

Now, we have the following acceptable path:

$$1 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 3 \rightarrow 7 \rightarrow 8 \tag{5}$$

**Table 2** Chromosome to path conversion for our typical example

| $i$ | Gene value | Selected node's neighbors | Path index | Chosen path |
|---|---|---|---|---|
| 1 | 3 | {2,3,5} | 3mod3 = 0 | 2 |
| 2 | 4 | {4,5} | 4mod2 = 0 | 4 |
| 3 | 2 | {5,8} | 2mod2 = 0 | 5 |
| 4 | 2 | {3,8} | 2mod2 = 0 | 3 |
| 5 | 5 | {6,7} | 5mod2 = 1 | 7 |
| 6 | 4 | {6,8} | 4mod2 = 0 | 6 |
| 7 | 1 | | | X |

**Table 3** Chromosome to path conversion for our typical example after correction

| $i$ | Gene value | Selected node's neighbors | Path index | Chosen path |
|---|---|---|---|---|
| 1 | 3 | {2,3,5} | 3mod3 = 0 | 2 |
| 2 | 4 | {4,5} | 4mod2 = 0 | 4 |
| 3 | 2 | {5,8} | 2mod2 = 0 | 5 |
| 4 | 2 | {3,8} | 2mod2 = 0 | 3 |
| 5 | 5 | {6,7} | 5mod2 = 1 | 7 |
| 6 | 4 | {6,8} | Next | 8 |
| 7 | 1 | | Exit | |

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

$$\begin{aligned} C_{\text{delay}} = 86 &\Longrightarrow \text{Ratio (delay)} = 1.72 \\ C_{\text{cost}} = 102 &\Longrightarrow \text{Ration (cost)} = 2.04 \end{aligned} \tag{6}$$

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

$$\text{gene (3)} = 2 \overset{\text{Mutation}}{\Longrightarrow} 3 \tag{7}$$

Now, the chromosome is converted to another one which directs the packets to the destination through a shorter path. The procedure of converting this chromosome into a solution is presented in Table 4.

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

$$\begin{aligned} C_{\text{delay}} = 44 &\Longrightarrow \text{Ratio (delay)} = 0.88 \\ C_{\text{cost}} = 44 &\Longrightarrow \text{Ration (cost)} = 0.88 \end{aligned} \tag{8}$$

**Table 4** Chromosome to path conversion after the mutation

| $i$ | Gene value | Selected node's neighbors | Path index | Chosen path |
|---|---|---|---|---|
| 1 | 3 | {2,3,5} | 3mod3 = 0 | 2 |
| 2 | 4 | {4,5} | 4mod2 = 0 | 4 |
| 3 | 3 | {5,8} | 3mod2 = 1 | 8 |
| 4 | 2 | | Exit | |
| 5 | 5 | | | |
| 6 | 4 | | | |
| 7 | 1 | | | |

Both constraints' values are less than 1 which means that this path is feasible for the given application. The mutation process is depicted in Fig. 4.

### 3.4 The proposed method's properties

In the ES that we have developed in our proposed method, no crossover operator is applied and only mutation operator is used to evolve the population. This helped us to skip implementing time-consuming procedure of recombination whose consequence is nothing but another way of mutation. In other words, since chromosomes' genes contain integers that do not have any direct relationship with real path nodes' numbers, and they will be converted into real path nodes' numbers using our customized decoding mechanism, recombining a chromosome with another one results in producing two new chromosomes which are actually the mutated version of their parents. In fact, they do not inherit the traits of their parents because their changed genes are not really changed path nodes' numbers but they may or not be converted into another corresponding nodes' numbers in the decoding phase. Therefore, by choosing ES, we have benefited the mentioned features and consequently decreased the complexity of the algorithm and increased the convergence speed.

The main properties of the proposed method can be summarized as the following:

(a) The time complexity of the algorithm is not considerable. It is due to our simple fitness function and using one evolutionary operator (i.e., mutation).
(b) The proposed fitness function is so flexible that can tackle any number of constraints, without any major change in the algorithm.
(c) The chromosome's lengths are fixed and equal.
(d) There is no need for any validation phase to check the existence of loops or invalid chromosomes.

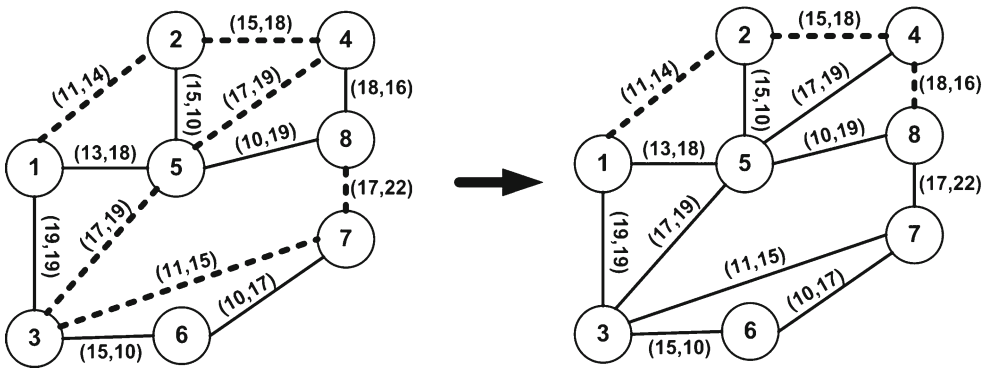


**Fig. 4** Offspring generated after applying mutation operator

## 4 Simulation results

In order to compare the proposed algorithm with five other successful algorithms, ARGA (Barolli et al. 2003), ARGAQ (Barolli et al. 2004), MURUGA (Leela et al. 2011) and the state of the art TING (Ting and Zhu 2013) and MFA-RA (Liu et al. 2015), some simulations were done and their results are presented in this section. The simulations of all experiment scenarios were implemented in MATLAB environment on a personal computer with 2 GB of RAM and 2.60 GHz Core 2 Duo Intel CPU.

In our experiments, we generated all graphs using Waxman random network (Waxman 1998). The main feature of this network generator is that it distributes the nodes of the network uniformly and generates links based on some 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 Waxman (1998). In our simulations, we assumed the Waxman parameters as $\alpha = 0.2$ and $\beta = 0.1$. Our multi-constraint routing problems are subjected to two QoS metrics: TD (transmission delay) and TSR (transmission success ratio), which hereafter will be referred to as by $d$ and $r$, respectively. For being consistent with one of our main references, $d$ and $r$ are randomly chosen in a uniform distribution [0, 100] and [0, 1], respectively (Ting and Zhu 2013). Moreover, 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 taken 20 nodes. In order to perform fair comparison between the mentioned algorithms, according to TING, the population size of all algorithms was taken 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 among other population members. The chromosome with the lowest cost is considered as 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 (Kormza and Krunz 2001), 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 results of this comparison are shown in Fig. 5.

According to this figure, 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 out of the range. Therefore, it can be concluded that the proposed algorithm has better convergence performance in comparison with others, especially ARGA and ARGAQ. Regarding the simulation results, the number of generations needed to reach a feasible solution in our algorithm is averagely 3.65 and 9.94 times less than those of ARGAQ and ARGA, respectively. It should be noted that since MFA-RA is a single solution algorithm it could not take part in this experiment.

In order to prove the reliability and significance of our proposed method in all simulation runs, we repeated the previous experiment for 20 runs; each run generates a Waxman network with 20 nodes and the fitness of the chromosome with rank 1 in all algorithms is returned. The box-plot of the results is illustrated in Fig. 6. It should be noted that in order to perform a fair comparison, all algorithms are implemented in the same programming language and run on the same computer, and finally the same scenarios in terms of network size, topology, link attributes and traffic patterns are considered for all algorithms comparison.

As can be seen from the figure, it can be inferred that not only the results generated by the proposed method are always

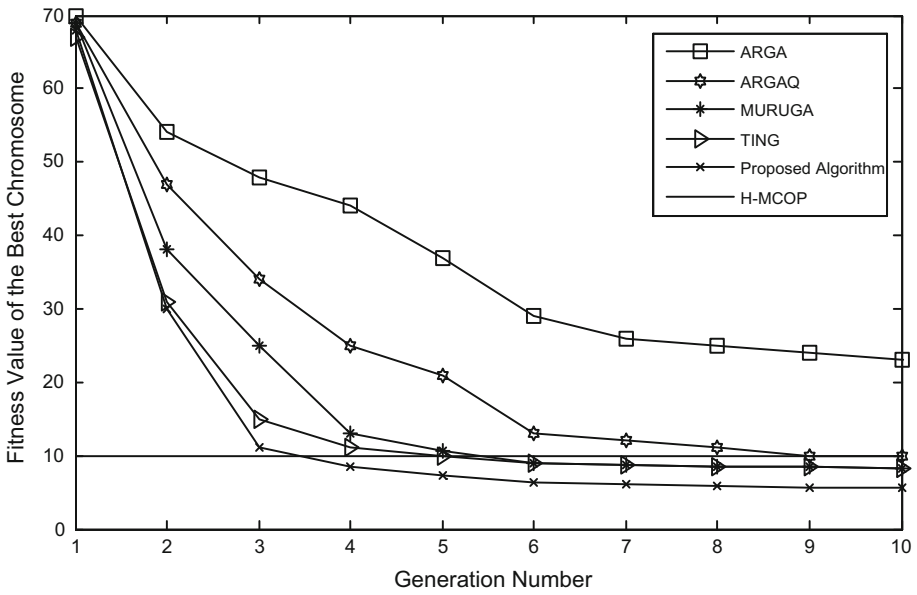


**Fig. 5** Fitness value of the chromosome with rank 1 versus generation number

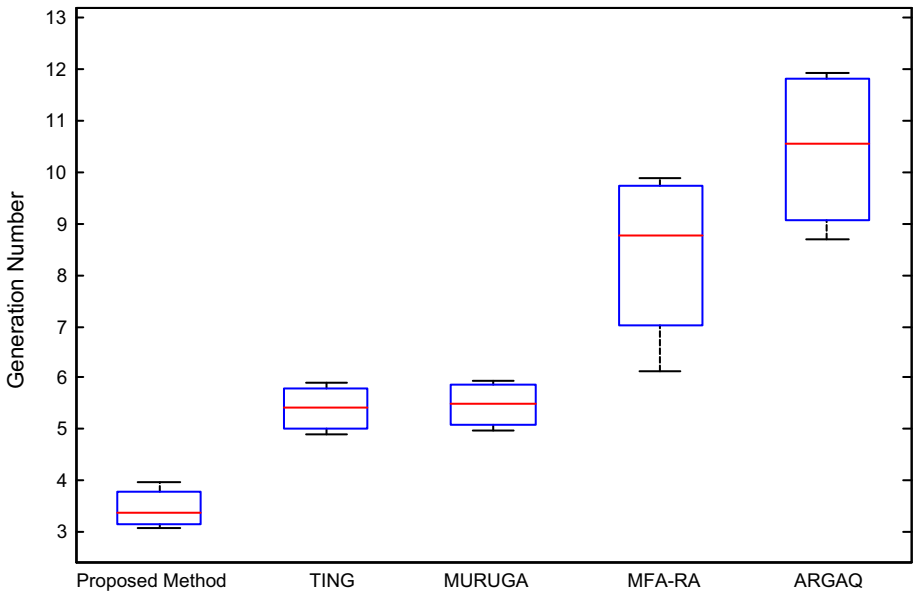


**Fig. 6** Box-plot of the fitness value of the chromosome with rank 1 versus generation number for 20 runs.

superior to others, but also its reliability, i.e., the difference between the worst and the best results, is situated closely in a narrow box. However, those of ARGAQ are disturbed in a wide range from about 9 to 12. The results produced by ARGA are not presented in the figure since they are highly out of range.

Having set the maximum number of generations to 10 and run the simulation 20 times, we examined the success ratio (SR) of our algorithm to find a feasible path in a 20-node network. In fact, we measured the success ratio of our proposed method for a number of routing demands. SR is defined as dividing the number of founded feasible paths by the number of all routing demands. Here again, the number of population individuals is 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 Ting and Zhu (2013) and are shown as a diagram in Fig. 7.

This figure shows that our algorithm outperforms others, especially it has much better success ratio than ARGA, ARGAQ and MFA-RA. Regarding the simulation results, the success ratio of the proposed algorithm is averagely 1.56% better than those of MURUGA and TING, and it is 1.02, 1.07 and 2.15 times more successful than MFA-RA, 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 ran-

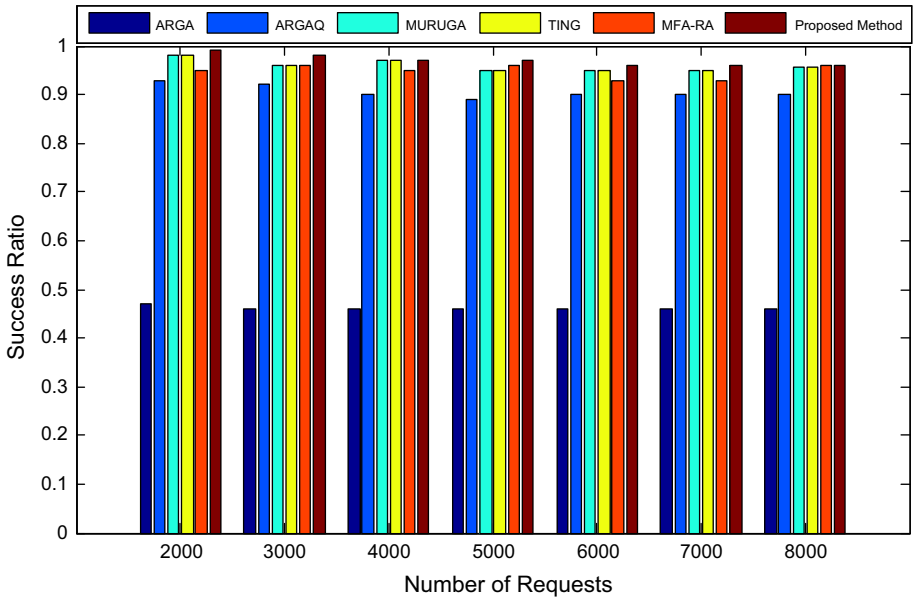


**Fig. 7** Percentage of successful routing versus the number of routing demands

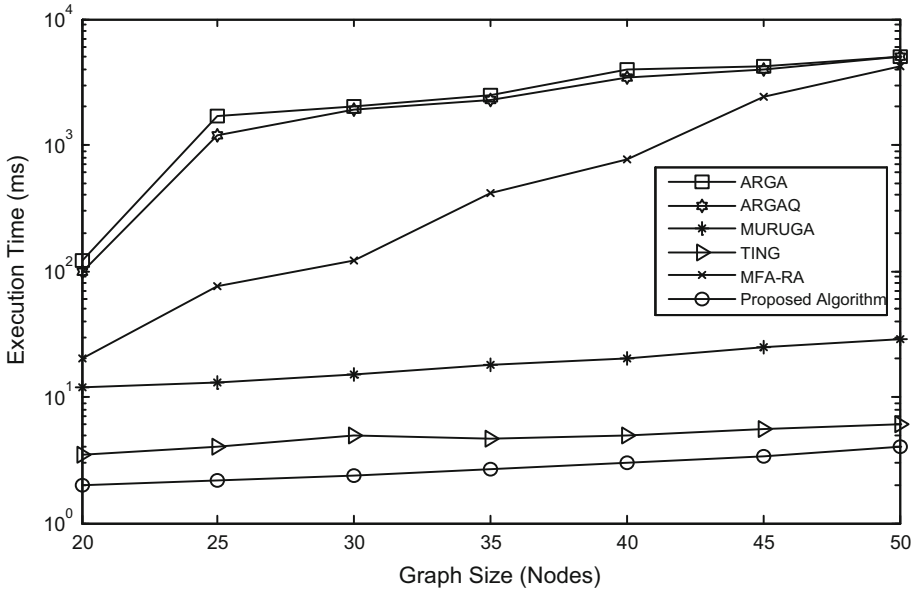


**Fig. 8** Execution time of the algorithms to find a feasible path in a fixed number of generations

dom networks. Finally, the maximum number of generations was considered 10. Since the number of generations does not necessarily indicate high speed of an algorithm, another criterion must be observed. A given algorithm in comparison with another 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 time needed to find a feasible path by the algorithms. The result of this experiment is shown in Fig. 8. It should be noted that in order to perform a fair comparison, all algorithms are implemented in the same programming language and run on the same computer, and finally the same scenarios in terms of network size, topology, link attributes and traffic patterns are considered for all algorithms comparison.

As can be seen from this figure, 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. Nevertheless, the other algorithms, except TING and MFA-RA, check the validity of chromosomes' genes at the end of each generation which imposes a considerable time overhead. The main reason which makes the MFA-RA algorithm a time-consuming one is that it needs many computations to calculate MFA equa-

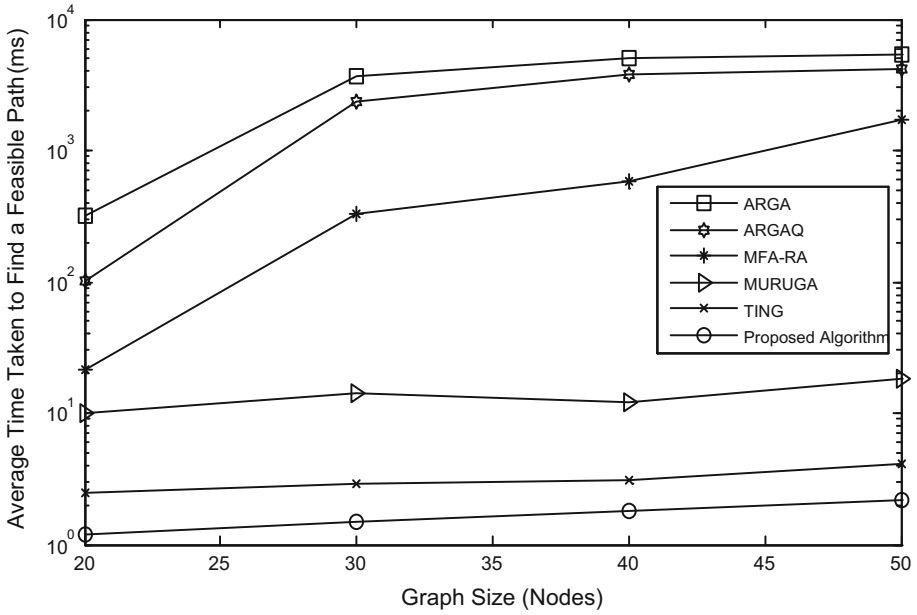


**Fig. 9** Average run time to find a feasible solution by the algorithms

**Table 5** Mann–Whitney *U* test

| Routing algorithms | Mann–Whiney (*p* value) | | |
|---|---|---|---|
| | Generation number | Success ratio | Average execution time |
| Proposed algorithm versus ARGA | 0.0265 | 0.0234 | 0.0155 |
| Proposed algorithm versus ARGAQ | 0.0389 | 0.0006 | 0.0264 |
| Proposed algorithm versus MURUGA | 0.0466 | 0.0168 | 0.0083 |
| Proposed algorithm versus TING | 0.0065 | 0.0081 | 0.0300 |
| Proposed algorithm versus MFA-RA | 0.0284 | 0.0396 | 0.0131 |

tions and its updating in each temperature. Moreover, by increasing the size of graph, the time needed to perform their calculations actually grows. Numerically, ours is 2.6, 11.3, 390, 868 and 922 times faster than TING, MURUGA, MFA-RA, ARGAQ and ARGA, respectively.

Since one might claim that comparing the execution time of the algorithms for a fixed number of generations is not fair, another experiment has been designed to evaluate the run time of the algorithms to find a feasible path in the network. The scenario of this experiment is the same as the previous one. The results of this experiment are presented in Fig. 9.

From Fig. 9, it can be seen that the proposed method always takes the least time to find a feasible path when compared to its competitor algorithms. The following reasons justify the trend in this figure: ARGA takes much more time because the check condition is invoked each time to validate the genes (partial routes). When the number of nodes increases, the connectivity matrix increases which results in an increase in search time and validation time. ARGAQ takes more time to generate the tree with all possible paths. As the network size in ARGAQ increases, the solution space increases which in turn results in an increase in search time. However, in MURUGA, the gene structure is generated without identifying all possible paths, resulting in reduced search time. The proposed method which is needles of any repair phases and also uses lightweight decoding mechanism as well as exploiting only the mutation operator has the lowest run time. Thus, simulation results show that the proposed method is strongly faster than the compared methods.

In order to make sure the differences in performance of the compared algorithms observed in the results are not occurred due to random chance with the samples available, Mann–Whitney–Wilcoxon test is applied. This nonparametric test also known as rank-sum test (Wilcoxon 1945) is chosen because it does not make any particular assumptions about the distribution of the result, avoiding the need to verify the data conform to the test assumption. Another benefit of using the rank-sum test is that the statistics generated from this test can be used to perform scientific significant test. Since in Mann–Whitney test, two samples are compared at one time, all algorithms are compared pairwise with their results summarized in Table 5. According to this table, it can be inferred that the proposed approach is statistically significant at the 95% confidence level.

Another important point which should be addressed is that by increasing the size of the network, i.e., the number

of nodes and links, it is obvious that the average number of every node's immediate neighbors will be increased too. As a result, numerous neighbor selection options increases the complexity of deciding which node should be the subsequent hop of the route. Such a complexity varies from one network topology to another. For example, for a structured peer-to-peer network every node has at least $O(\log N)$ outgoing nodes or links (Chun et al. 2005). Complexity analysis of the proposed algorithm without having those of other comparative algorithms is not effective. However, it should be noted that since most of those algorithms are GA-based, none of them exploits any direct neighbor selection mechanism and the sequence of the path's nodes is changed only based on evolutionary operators. Therefore, the main reason, which increases the time and space complexity of these algorithms, is their checking and repairing phases as well as genetic operators.

## 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. Recently, EAs are widely used to tackle the problem. However, existing evolutionary methods have three main drawbacks, since they mostly consider a single constraint instead of multi-constraints, they in a way use complex evolutionary operators which increases the algorithm run time and require extra steps to correct the invalid individuals at every stage of the algorithm. In this paper, a multi-constraints QoS routing algorithm is developed based on ES, which finds 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 successful methods. The results show that our algorithm outperforms the other two, in terms of algorithm run time and success ratio. More precisely, it is averagely 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. In the future, we intend to extend the comparison metrics, such as dynamic adaptability degree (DAD), to prove the proposed method's efficiency as much as possible.

### Compliance with ethical standards

**Conflict of interest** The Authors declare that they have no conflict of interest.

## References


Abdullah N, Al-wesabi OA, Baklizi M, Kadhum MM (2017) Intelligent routing algorithm using genetic algorithm (IRAGA). In: International conference of reliable information and communication technology. Springer, pp 255–263

Ahn CW, Ramakrishna RS (2002) A genetic algorithm for shortest path routing problem and the sizing of populations. IEEE Trans Evolut Comput 6:566–579

Alejandro RR, Chin KW, Soh S, Raad R (2015) On the performance of online and offline green path establishment techniques. EURASIP J Wirel Commun Netw 1:1–17

Bäck T (1995) Evolution strategies: an alternative evolutionary algorithm. In: European conference on artificial evolution. Springer, Berlin, Heidelberg, pp 1–20

Back T, Rudolph G, Schwefel H (1993) Evolutionary programming and evolution strategies: similarities and differences. In: Proceedings of the second annual conference on evolutionary programming, pp 11–22

Barolli L, Koyama A, Sawada H, Suganuma T, Shiratori N (2002) A new QoS routing approach for multimedia applications based on genetic algorithms. In: Proceedings of the international IEEE conference on cyber worlds, pp 289–295

Barolli L, Koyama A, Suganuma T, Shiratori N (2003) A genetic algorithm based QoS routing method for multimedia communications over high-speed networks. IPSJ J 44(2):544–552

Barolli L, Koyama A, Matsumoto K, Apduhan BO (2004) A GA-based Multi-purpose optimization algorithm for QoS routing. In: Proceedings of the international IEEE conference on advanced information networking and applications, pp 23–28

Benlai L, Yu J (2015) One multi-constraint QoS routing algorithm CGEA based on ant colony system. In: Information science and control engineering (ICISCE), 2nd international conference on IEEE, pp 848–851

Chitra C, Subbaraj P (2012) A non-dominated sorting genetic algorithm solution for shortest path routing problem in computer networks. Expert Syst Appl J 39:1518–1525

Chun B, Zhao B, Kubiatowicz J (2005) Impact of neighbor selection on performance and resilience of structured P2P networks. In: International workshop on peer-to-peer systems, pp 264–274

Deng W, Zhao HM, Yang XH, Xiong JX, Sun M, Li B (2017a) Study on an improved adaptive PSO algorithm for solving multi-objective gate assignment. Appl Soft Comput 59:288–302

Deng W, Zhao HM, Zou L, Li GY, Yang XH, Wu DQ (2017b) A novel collaborative optimization algorithm in solving complex optimization problems. Soft Comput 21(15):4387–4398

Feng G, Curry E, Intizar Ali M, Bhiri S, Mileo A (2014) Qos-aware complex event service composition and optimization using genetic algorithms. In: Service-oriented computing. Springer, pp 386–393

Gu B, Sheng VS (2016) A robust regularization path algorithm for $\nu$-support vector classification. IEEE Trans Neural Netw Learn Syst. https://doi.org/10.1109/TNNLS.2016.2527796

Gu B, Sun X, Sheng V (2016) Structural minimax probability machine. IEEE Trans Neural Netw Learn Syst. https://doi.org/10.1109/TNNLS.2016.2544779

Jain S, Sharma JD (2012) Tree structured encoding based multi-objective multicast routing algorithm. Int J Phys Sci 7:1622–1632

Karthikeyan P, Baskar S (2015) Genetic algorithm with ensemble of immigrant strategies for multicast routing in Ad hoc networks. Soft Comput 19:489–498

Karthikeyan P, Baskar S, Alphones A (2013) Improved genetic algorithm using different genetic operator combinations (GOCs) for multicast routing in ad hoc networks. Soft Comput 17:1563–1572

Kong Y, Zhang MJ, Ye DY (2016) A belief propagation-based method for task allocation in open and dynamic cloud environments. Knowl Based Syst 115:123–132

Kormza T, Krunz M (2001) Multi-constrained optimal path selection. In: Proceedings of the twentieth annual joint conference of the IEEE computer and communications societies, pp 834–843

Leela R, Thanulekshmi N, Selvakumar S (2011) Multi-constraint QoS unicast routing using genetic algorithm (MURUGA). Appl Soft Comput J 11:1753–1761

Li P, Guo S, Yu S, Vasilakos AV (2012) CodePipe: an opportunistic feeding and routing protocol for reliable multicast with pipelined network coding. In: Proceedings of the IEEE conference on INFOCOM, pp 100–108

Liu L, Peng Y, Xu W (2015) To converge more quickly and effectively—mean field annealing based optimal path selection in WMN. Inf Sci J 294:216–226

Liu Q, Cai WD, Shen J, Fu ZJ, Liu XD, Linge N (2016) A speculative approach to spatial–temporal efficiency with multi-objective optimization in a heterogeneous cloud environment. Secur Commun Netw 9(17):4002–4012

Martín V, Skorin-Kapov L, Ebrahimi T (2014) Quality of service versus quality of experience. Quality of experience. Springer International Publishing, Berlin, pp 85–96

Munetomo M, Takai Y, Sato Y (1998) An adaptive routing algorithm with load balancing by a genetic algorithm. IPSJ J 39(2):219–227

Ni M (2011) A simple and fast algorithm for restricted shortest path problem. In: Proceedings of the international IEEE conference on computational sciences and optimization. pp 99–102

Rong H, Ma T, Tang M, Cao J (2017) A novel subgraph $K^+$-isomorphism method in social network based on graph similarity detection. Soft Comput. https://doi.org/10.1007/s00500-017-2513-y

Roy A, Das SK (2004) A QoS-based mobile multicast routing protocol using multi-objective genetic algorithm. Wirel Netw J 10:271–286

Rudolph G (1994) Convergence of non-elitist strategies. In: Proceedings of the first IEEE conference on evolutionary computation, pp 63–66

Salimans T, Ho J, Chen X, Sutskever I (2017) Evolution strategies as a scalable alternative to reinforcement learning. arXiv preprint arXiv:1703.03864

Ting L, Zhu J (2013) A genetic algorithm for finding a path subject to two constraints. Appl Soft Comput J 13:891–898

Vasilakos A, Ricudis C, Anagnostakis K, Pedryca W, Pitsillides A (1998) Evolutionary-fuzzy prediction for strategic QoS routing in broadband networks. In: Proceddings of the IEEE world congress on computational intelligence and fuzzy systems, pp 1488–1493

Voigt H, Mfihlenbein H, Cvetkovid D (1995) Fuzzy recombination for the breeder genetic algorithm. In: Proceedings of the 6th international conference, pp 104–111

Waxman BM (1998) Routing of multipoint connection. IEEE J Sel Areas Commun 6:1617–1622

Wilcoxon F (1945) Individual comparisons by ranking methods. Biom Bull 6:80–83

Xue Y, Jiang JM, Zhao BP, Ma TH (2017) A self-adaptive artificial bee colony algorithm based on global best for global optimization. Soft Comput. https://doi.org/10.1007/s00500-017-2547-1

Yena YS, Chao HC, Changd RS, Vasilakos A (2011) Flooding-limited and multi-constrained QoS multicast routing based on the genetic algorithm for MANETs. Math Comput Model J 53:2238–2250

Yin PY, Chang RI, Chao CC, Chu YT (2014) Niched ant colony optimization with colony guides for QoS multicast routing. J Netw Comput Appl 40:61–72

Yuan C, Xia Z, Sun X (2017) Coverless image steganography based on SIFT and BOF. J Internet Technol 18(2):209–216

Zaheeruddin, Lobiyal DK, Prasad S (2017) Ant based Pareto optimal solution for QoS aware energy efficient multicast in wireless networks. Appl Soft Comput 55:72–81. https://doi.org/10.1016/j.asoc.2017.01.029

Zeng Y, Xiang K, Li D, Vasilakos A (2013) Directional routing and scheduling for green vehicular delay tolerant networks. Wirel Netw 19(2):161–173

Zhang Y, Huang H, Lin Z, Hao Z, Hu G (2016) Running-time analysis of evolutionary programming based on Lebesgue measure of searching space. Neural Comput Appl, pp 1–10