Satya Prakash Sahoo
Manas Ranjan Kabat

MULTI-CONSTRAINED MULTICAST ROUTING IMPROVED BY HYBRID BACTERIA FORAGING/PARTICLE SWARM OPTIMIZATION

Abstract *Many network applications like IPTV, news distribution, online chatting, video conferencing, and online games require multi-constrained multicasting. To solve multicast routing under multiple constraints, it is necessary to generate a multicast tree structure that ranges from a source to the destinations with a minimum cost and subject to several constraints. In this paper, PSOhas been embedded with BFO to improve the convergence speed and avoid premature convergence to be used for solving the QoS multicast routing problem. The algorithm proposed here generates a set of delay compelled links to each destination present in a multicast group. Then, the bacteria foraging algorithm (BFA) selects the paths to all destinations sensibly from the set of least delay paths to construct a multicast tree. To maintain a fair balance among the intensification and diversification of the algorithm being proposed, we have dynamically tuned the parameters of PSO to meet the global search and BFO, which uses the global search technique of PSO to minimize the delay to generate an optimal solution. The robustness of the algorithm being proposed had been established through a simulation. The efficiency and effectiveness of the algorithm being proposed was validated through a comparison study with other existing meta-heuristic algorithms. This shows that our proposed IBF/PSO algorithm outperforms its competitive algorithms.*

Keywords QoS routing, Multicasting, Particle Swarm Optimization, Bacteria Foraging Optimization

Citation Computer Science 20(2) 2019: 245–270

1. Introduction

There are different techniques for data communication that work over IP networks for real-time and multimedia applications such as IPTV, video/audio conferencing, online games, news distribution, and distance learning, and multicast routing is one of them. These applications require data transmission from one to many users or many to many users under strict quality of service (QoS) metrics such as jitter, end-to-end delay, loss rate, and bandwidth. The increasing demand of multimedia applications that requires sending a packet to more than one destination host over the Internet has motivated the researchers for developing a multicast. The most critical problem in QoS multicast is to support point-to-multipoint communication respecting the QoS constraints for these applications that ensures sufficient bandwidth as well as low latency, jitter, and packet loss. The problem of finding an optimal solution of the multicast routing problem subject to cost minimization is known as the NP-hard Steiner tree problem [29]. Moreover, the cost-optimal multicast tree construction under multiple constraints is known as a constrained multicast tree, which is also NP-hard [16]. The optimization of resource utilization and guaranteeing multiple constraints in multicast routing are two conflicting interests [5]. It is possible to achieve a tradeoff – we aim to find the optimal solution under multiple constraints. Therefore, our aim is to find a cost-sensitive tree structure that spans to all destinations from the source where the end-to-end links to the destinations from the source satisfy the QoS constraints.

As proposed by K.M. Passino [17], the bacterial foraging optimization (BFO) algorithm is one of the many recent and deterministic meta-heuristic methods. BFO oriented by PSO(BF/PSO) is a meta-heuristic method that has a high convergence speed that has been used very effectively to solve a large variety of optimization problems. We propose an IBF/PSO-based multicasting algorithm here that can be used to solve the QoS-based multicasting problem. In this article, we have used an improved BFO-based algorithm that implements the PSO parameters and generates a multi-constrained multicast tree with the lowest cost. First, we compute k-delay constrained links to each destination using the k-Bellman-Ford algorithm. Then, the links to each destination are randomly combined to construct a multicast tree. The BFO algorithm represents each multicast tree as bacteria. The PSO parameters are used during the chemotactic process of the BF algorithm to find the direction of tumbling. Here, we exploit the speed of PSO and the searching power of the BF algorithm simultaneously to generate a cost-sensitive multicast tree under multiple constraints in less time than that of the other available meta-heuristic algorithms.

The motivation for applying a hybrid BF/PSO for the QoS multicast routing problem is that it uses PSO for global searching and BFO for local searching. Thus, there is a fair chance that the algorithm will not be trapped into the local minima and the algorithm will converge into a feasible multicast tree in a shorter amount of time.

The main contribution of the paper is summarized as follows:

1. The multicast routing algorithm based on a hybrid BF/PSO algorithm is developed to generate a cost-sensitive multicast tree satisfying multiple constraints of multimedia applications over the Internet.
2. The flexible fitness function defined in this paper can be extended to any number of parameters and used to optimize multicast routing problems for other types of networks.
3. An efficient loop-detection algorithm is developed to delete the edges that have greater costs. Therefore, cost-sensitive multicast trees are constructed in each iteration.

The remaining part of this paper is structured as follows. The problem definition of multicast routing is presented in the next sub-section. Section 2 describes the meta-heuristic approaches used to create a QoS multicast tree. An overview of BFO and PSO is presented in Section 3. Section 4 describes the hybrid IBF/PSO algorithm for QoS multicast tree generation. The results obtained from the simulation are represented in Section 5, and the summary and conclusion are given in Section 6.

1.1. Problem Formulation for QoS Multicast

Let us consider that $G = (V, E)$ is an undirected weighted graph that symbolizes the communication between the vertices of a network, where V is the set of vertices that symbolizes the switches or routers and E is the set of links that symbolizes the connections between the vertices (in either a logical or physical manner). Let $s \in V$ represent the source vertex, $M \subseteq V$ be the set of target vertices such that $M = \{d_j \in V, d_j \neq s, j = 1, 2, \ldots m\}$, and m be the number of target vertices. Each edge is assigned as a set of positive real numbers for representing QoS metrics such as delay, jitter bandwidth, loss rate, etc. Let the QoS parameters for each link $e \in E$ be defined as $bandwidth(e)$, $delay(e)$, $jitter(e)$, $andlossrate(e)$ as well as the set value assigned to each edge $E \in R^+$, where R^+ is a set of positive real numbers.

The QoS metrics consist of loss rate that is a multiplicative delay (which is additive) and a bottleneck metric (which represents bandwidth). The bottleneck metric can deal with the available bandwidth by pruning the links from the groups that are not satisfying the QoS constraint. Let $T(s, m)$ be the multicast tree spanning from source s to set of destinations M. The end-to-end delay of path $P_T(s, d_j)$ from source s to destination d_j of the multicasting group is the sum of the delay of the links present in the path.

$$dealy\left(P_T\left(s, d_j\right)\right) = \sum_{e \in P_T(s,d_j)} delay(e), \forall d_j \in M \tag{1}$$

The loss rate of the path $P_T\left(s, d_j\right)$, cost, delay, jitter, and bandwidth of the tree are defined as follows:

$$lossrate\left(P_T\left(s, d_j\right)\right) = 1 - \left\{ \prod_{e \in P_T(s,d_j)} \left(1 - lossrate(e)\right) \right\}, \forall d_j \in M \tag{2}$$

$$delay\left(T(s,M)\right) = \max\left\{delay\left(P_T\left(s,d_j\right)\right)\right\}, \forall d_j \in M \tag{3}$$

$$cost\left(T(s,M)\right) = \sum_{e \in T(s,M)} cost(e) \tag{4}$$

$$jitter\left(T(s,M)\right) = \sum_{e \in T(s,M), d_j \in M} \sqrt{\left(delay\left(s,d_j\right) - delay\_avg\right)^2} \tag{5}$$

$$bandwidth\left(T(s,M)\right) = \min\left\{bandwidth(e) | e \in T(s,M)\right\} \tag{6}$$

, where the average value of the delay of the paths from source to destination is denoted by *delay_avg*. The objective of the QoS multicast tree problem is to construct the lowest-cost multicast tree under multiple QoS constraints. The QoS constraints are roughly defined as link constraint (*bandwidth*), path constraints (*delay*, *lossrate*), and tree constraints (*delay*, *jitter*). The lowest-cost multi-constrained multicast tree problem is defined as follows:

Minimize $cost\left(T(s,M)\right)$ subject to:

$$dealy\left(P_T\left(s,d_j\right)\right) \le DC, \forall d_j \in M \tag{7}$$

$$lossrate\left(P_T\left(s,d_j\right)\right) \le LC, \forall d_j \in M \tag{8}$$

$$jitter\left(T(s,M)\right) \le DJC \tag{9}$$

$$bandwidth(e) \le BC, \forall e \in P_T\left(s,d_j\right) \text{ and } d_j \in M \tag{10}$$

, where, DC, LC, DJC, and BC represent the delay constraints, loss constraint, delay jitter constraint, and bandwidth constraints, respectively.

2. Related Works

In a related work, we present various meta-heuristic algorithms developed to solve multi-constrained QoS multicast routing. Hwang et al. [8] and Haghighat et al. [7] proposed methods for optimizing the QoS multicast problem using a genetic algorithm (GA). These algorithms first compute all of the possible paths to all destinations using the *k*-shortest path algorithm [33]. Each path is considered to be a gene, and the paths to each destination are randomly combined to form a chromosome. Then, various GA operations such as crossover, mutation, and eliminations are applied to generate the best-fit chromosome as the final solution.

However, the above algorithms suffered from a poor local search problem. To overcome this problem, a new method called GSA (Genetic Simulated Annealing) was proposed by Zhang et al. [4]. GSA is a tree-based approach that combines simulated annealing (SA) with a genetic algorithm (GA) to avoid the early convergence of the GA. SA has the capability of escaping from the local optimum proposed by Kun et al. in 2009 [35]. Subsequently, Peng et al. [19] presented a simulated annealing algorithm to solve the QoS multicast routing problem based on an adaptive genetic algorithm.

In this approach, a multicast tree is created as it is in GSA and use the roulette wheel method for selection.

Gong et al. [35], Wang et al. [27], Shi et al. [24], and Younes [36] proposed the ant colony optimization (ACO) algorithm for solving QoS multicast routing under multiple constraints. They used a path-based approach in which the path set is combined intelligently to generate a multicast tree. The fitness function is used to select the fittest multicast tree among the candidate trees as the desired solution. Furthermore, Wang et al. [28] proposed a tree-based ant colony optimization algorithm for the QoS multicast routing problem in which a set of multicast trees are generated randomly. Then, ACO is used to generate the best multicast tree. Yin et al. [33] proposed a niched ant colony-based optimization (NACO) algorithm to solve the QoS multicast routing problem. The NACO algorithm is basically a hybridization of the path-based and tree-based approaches. Sharma et al. [23] proposed an optimal routing algorithm for a wireless mesh network (WMN) based on the foraging behavior of ants to maximize the network performance and optimize the utilization of network resources simultaneously. The tree-based PSO algorithm (PSOTREE) for QoS multicast routing is proposed by Wang et al. [26]. The PSOTREE algorithm constructs the candidate multicast trees by calculating the fitness of the link by considering the cost, delay, and bandwidth of the links. The generated multicast tree is represented by a particle, and the particle with the best-known direction interacts with another to produce a new particle. The new particle fitness is computed, and the global best solution is updated according to the fitness.

Qu et al. [20] proposed the jumping PSO (JPSO) algorithm, which is a tree-based approach that generates a set of multicast trees randomly. These multicast trees are represented as a swarm of random particles. A local search is applied after a particle jumps to a new position. Then, the particle and swarm are both updated with their best positions. The QPSO algorithm was proposed by Sun et al. [25] for the QoS multicast routing problem used a path-based approach. In each iteration, the QPSO algorithm checks the fitness value of each multicast tree and updates both its current and global positions. R.F. Abdel-Kader [1] proposed a hybrid algorithm that combines both the power of PSO and GA to solve the multicast routing problem. This hybrid algorithm uses the position velocity update rules of PSO along with the crossover, mutation, and selection operators of GA.Represented as a particle, a multicast tree is constructed by choosing the path that satisfies the QoS constraints and removing the loops from the combination of paths.

Wei et al. [15] considered multicast routing with multiple QoS constraints in mobile ad-hoc networks (MANETs). They proposed an EA/ACA/QMRA algorithm (evolutionary algorithm/ant colony algorithm/QoS multicast routing algorithm) that uses the global exploration capability of an evolutionary algorithm to search the global solutions and the ant colony algorithm to optimize the solution. A hybrid swarming agent-based ACO/PSO algorithm has been proposed by Patel et al. [18]. A tree-based approach in which fixed pattern agents are created (called particles) to associate with each tree. The particles can move from pattern to pattern to interact with other particlesdynamically to generate candidate solutions.

Sangeetha et al. [22] proposes a solution for QoS support in real-time applications of WiMAX. They used a gene expression programming (GEP) technique in WiMAX and obtain the mathematical expression for throughput in terms of bandwidth, average end-to-end delay, and average jitter. To employ the GEP technique, they used relative mean square error (rMSE) as the fitness function. Felber et al. [4], Ahlgren et al. [2], and Xylomenos et al. [30] presented the design choices and features of an information-centric network (ICN)as well as its applications, scalability, cost efficiency, persistency, security, mobility, and tolerance. Yadav et al. [31] proposed a reliable QoS multicast routing protocol for MANET using reliable node selection schemes. The proposed protocol (QMRPRNS) calculated the reliability pair factor based on the energy level among the nodes involved in communication. Then, the multicast routes are assigned based on the maximum value of the reliability pair factor of a route.

Jiang et al. [10] proposed an innovation in the selection of an access network among the available diversified networks based on multiple decision factors. Here, how an application in a vehicle will select an available infrastructure network among a diversified network based on energy efficiency is proposed. To effectively describe the users' preference and network performance, they develop a utility function based on energy efficiency, signal intensity, network cost, delay, and bandwidth. Then the network selection process under multiple constraints was modeled as a multi-constraint optimization problem. Finally, they use a multi-criteria access selection algorithm to solve the built model.

Jiang et al. [11] proposed a topology control and cooperation-based energy-efficient multicast routing (TCEM) approach to achieve better network performance for multi-hop wireless networks. They made use of topology control along with the periodical and alternative sleep of network nodes to reduce network consumption and construct an energy-efficient optimal multicast route for smart medical applications.

Jiang et al. [14] proposed an energy-efficient minimum criticality routing algorithm (EEMCRA) and energy-efficient multi-constraint rerouting algorithm (E2MR2) for energy-efficient load balancing in smart city applications.I hope my interpretation is OK here They used a minimum criticality routing algorithm (MCRA) to find the energy-efficient paths. The paths generated through the MCRA are used with the EEMCRA algorithm to generate an energy-efficient and load-balanced path from source to destination. They further used the E2MR2 algorithm to finalize the energy-efficient paths that satisfy the constraints.

Jiang et al. [12] proposed an energy-efficient multicast routing model using network coding for data transmission in a multi-hop wireless network. They used the energy metric and energy efficiency metric in the proposed models and network coding to improve the network throughput. The distributions of the nodes in a wireless network are random in nature. To construct a network structure, the authors proposed three basic structures of network coding to overcome this problem.

Jiang et al. [13] proposed a next-generation energy-efficient multicast routing algorithm for multi-hop wireless networks. Their target was to optimize the energy efficiency by maximizing the network throughput and minimizing the energy consumption by the network using a network coding and sleeping scheme. To make the network coding effective, they used two cluster structures in a multi-hop wireless network. Their sleeping scheme allows the network node to stay in sleep mode whenever possible, considering the receiving and transmitting energy consumption of the nodes.

Jiang et al. [9] studied how to estimate and recover the end-to-end network traffic in a fine time granularity from sampled traffic in high-speed backbone networks for SDN applications. They used fractal interpolation to reconstruct the finer-granularity network traffic. Subsequently, they used cubic spline interpolation to reconstruct the end-to-end network traffic in a finer time granularity.

3. Hybrid BF/PSO Algorithm

The hybridization of the BFA and PSO algorithms (IBF/PSO) is emphasized in this work to improve the quality of the solutions of the QoS multicast routing problem. The idea of the BFA is to eliminate the bacteria with meager foraging strategies and favor the bacteria that have good foraging strategies [17]. The BFA searches a wide region to find the near-optimal solution efficiently; however, it suffers from low convergence. The ability of PSO to exchange the social information can be combined with the BFA to improve the exploration capability of the BFA and exploitation capability of PSO simultaneously. In this section, the basic concepts of the BFA, PSO, improved BFA, and hybrid IBF/PSO methods are presented.

3.1. Classical BFO Algorithm

The classical bacteria foraging algorithm mainly involves four steps: (i) chemotaxis; (ii)swarming; (iii) reproduction; and (iv) elimination and dispersal.

Chemotaxis: The simulation of the chemotaxis process is accomplished through the swimming and tumbling movements of E. coli bacteria. When a bacterium finds favorable surroundings (i.e., nutrient-rich and noxious free), it continues to swim in the nutrient direction; however, when it finds an unfavorable environment, it changes direction (i.e. it tumbles). An E. coli bacterium can alternate between swimming and tumbling operations during its entire lifespan. Tumbling can be presented as a unit length in a random direction – say, $Delta(m, i)$. The movement of an E. coli bacterium may be represented as follows:

$$\theta_i(m, j + 1, k, l) = \theta_i(m, j, k, l) + C(i) \times Delta(m, i) \tag{11}$$

, where m is the dimension of the search space, and $i = 1, 2, 3. \ldots B$ is the chemotactic index. $\theta_i(j, k, l) = P(j, k, l)$ is the location of each element in the population of B at the j^{th} chemotactic step, k^{th} reproduction step, and l^{th} elimination. $C(i)$ shows the step size taken in the random direction as given by the tumble. If the bacterium is better at position $\theta_i(m, j+1, k, l)$ than at position $\theta_i(m, j, k, l)$, then another value of the step size is considered in the same direction; otherwise, the bacterium tumbles in a random direction with another step size.

Swarming: Bacteria like E. coli display interesting group behaviors such as (i) sensing, (ii) actuation, and (iii) decision-making. With the movement of the bacterium, attractant aspartate is released; this signals others to move in a group and swarm towards it. If any one of them finds a better place, the bacterium attracts other bacteria to reach the desired place rapidly. In the meantime, each of the bacteria releases a repellent that informs others to keep a minimum distance among one another. $J(i, j, k, l)$ is the cost at the location of the i^{th} bacterium located in the m dimension. This social behavior is simulated by combining cell-to-cell repelling and attraction effects and may be represented by the following function:

$$\begin{aligned} J_{cc}(\theta, P(j,k,l)) &= \sum_{i=1}^{s} J_{cc}^{i}\left(\theta, \theta_i(j,k,l)\right) \\ &= \sum_{i=1}^{s}\left[-d_{attract} exp\left(-w_{attract}\sum_{m=1}^{p}\left(\theta_m - \theta_m^i\right)^2\right)\right] \\ &+ \sum_{i=1}^{s}\left[h_{repellent} exp\left(-w_{repellent}\sum_{m=1}^{p}\left(\theta_m - \theta_m^i\right)^2\right)\right] \end{aligned} \tag{12}$$

. Here, $J_{cc}(\theta, P(j,k,l))$ is a function that depends on time and is added to $J(i,j,k,l)$ so that the cells try to find nutrients, avoid noxious substances, and move towards other cells at the same time while maintaining a minimum distance between each other. Let B represent the total bacteria in number, D represent the number of variables to be optimized, and $d_{attract}$, $w_{attract}$, $h_{repellent}$, and $w_{repellent}$ represent the coefficients that are to be carefully selected.

Reproduction: Once the chemotaxis process is completed, the remaining bacteria population will undergo the reproduction stage. During this stage, the malign bacteria will die, and each wholesome bacterium will split into two different parts to ensure that the bacteria population remains constant. For convenience, we have considered B to be a positive integer. Consider $B_r = B/2$ to be the size of the population that has adequate nutrients so that they will appear in the reproduction steps with no mutation.

Elimination and Dispersal (ED): When the local environment changes, the bacteria population also changes. This happens by either the utilization of nutrients or due to changes in some other local influences; i.e., rising temperatures may kill most of the bacteria that live in a high-nutrient-concentration area. Some events may disperse or kill all of the bacteria that live in a specific region. This phenomenon can be simulated in the BFA by removing those bacteria that have small probabilities while randomly initializing some new bacteria over the search space. These elimination and dispersal events protect the bacteria from being trapped in a local optimum.

3.2. Standard PSO

Introduced in 1995 by Kennedy et al. [3], standard PSO was a bio-inspired meta-heuristic algorithm used to provide an optimized solution to a given problem based on the population. In this method, each particle is considered to be a member of the population in an m-dimensional search area, and the swarm is formed considering the set of particles. The velocity parameter of standard PSO is refreshed by the particle's own and flying experience. The aim of updating the velocity and position of a particle is to find the best solution in the sub-problem space. The few parameters to be tuned and easy to implement are the main emphasis of considering PSO for this optimizing problem. Let N_p be the size of the population in the generation, and PSO updates the position and velocity of each particle. These are given in the following mathematical expression:

$$v_i^m(t+1) = v_i^m(t) + \phi_1 * rand_1\,(pbest_i^m(t) - p_i^m(t)) + \phi_2 * rand_2\,(gbest^m(t) - p_i^m(t)) \tag{13}$$

$$p_i^m(t+1) = p_i^m(t) + v_i^m(t+1) \tag{14}$$

, where $p_i^m = (p_i^1, p_i^2 \ldots, p_i^m)$, and $v_i^m = (v_i^1, v_i^2 \ldots, v_i^m)$ represents the current position and velocity vector of particle i ($1 \leq i \leq N_p$) in the m-dimensional search space, respectively. ϕ_1 and ϕ_2 are the social and cognite factors, respectively, and $rand_1$ and $rand_2$ denote two random numbers uniformly distributed among 0 and 1. $pbest_i^m$ denotes the previous best position of particle i in iteration t, and $gbest^m$ denotes the previous global best position among all particles in iteration t.

3.3. Improved PSO

Parameter conversion is one of main weaknesses of standard PSO, which fails to provide a convergence in all cases to generate an optimal solution and also suffers from external parameters. Therefore, it does not provide a balance between intensification and diversification. To enhance the performance of PSO by maintaining a balance between its intensification and diversification properties, the inertia of weight has been enhanced dynamically in the PSO equation to update the position and velocity of each particle in the population. This also shows that a larger inertia value provides the ability to perform a global search, while a smaller inertia value provides the ability to perform a local search. Therefore, it is necessary to tune the parameters of PSO dynamically to improve the searching capabilities in a dynamic environment. Therefore, many schemes have been proposed to improve the PSO algorithm by updating the parameters in an adaptive manner in each generation. In the present context, PSO has been improved by adaptively adjusting the inertia value and acceleration coefficient to enhance the rate of convergence to the optimum value. The basic equation of standard PSO has been modified and presented mathematically as follows:

$$\begin{aligned} v_i^m(t+1) = w_i v_i^m(t) + \phi_1 * rand_1 \left(pbest_i^m(t) - p_i^m(t)\right) \\ + \phi_2 * rand_2 \left(gbest^m(t) - p_i^m(t)\right) + \phi_3 * rand_3 \left(\theta_i(t) - p_i^m(t)\right) \end{aligned} \tag{15}$$

$$.p_i^m(t+1) = p_i^m(t) + v_i^m(t+1) \tag{16}$$

In Eq.(15), $\theta_i(t)$ denotes the position of the neighbor node selected based on the fitness value. ϕ_1 and ϕ_2 are two random numbers uniformly distributed among 0 and 1. PSO updates its local best value mathematically as follows:

$$pbest_i^m(t+1) = \begin{cases} pbest_i^m(t) & \text{if } fit\left(p_i^m(t+1)\right) \geq fit\left(pbest_i^m(t)\right) \\ p_i^m(t+1) & \text{otherwise} \end{cases} \tag{17}$$

, where $fit(\ )$ denotes the fitness function of the particle in the population and updates its global best position as follows:

$$gbest^m(t) = \min\left\{fit\left(pbest_1^m(t)\right), fit\left(pbest_2^m(t)\right), ..., fit\left(pbest_{N_p}^m(t)\right)\right\} \tag{18}$$

. PSO improved its convergence rate by fine tuning the parameters in an adaptive manner without changing the internal structure of the algorithm. Here, PSO's inertia weight is changed dynamically to enhance the convergence rate. The empirical experiment has been performed in the part with a weight of inertia that lies between 0.9 and 0.4.

$$w_i = z_i + (w_{int} - \beta k_i + \lambda s) \tag{19}$$

, where k_i is the speed factor for each particle, w_{int} denotes the initial value of the inertia weight, and β and λ are the random values lying in between 0 and 1. To generate a better result in k_i conduct of experiment, we have set $\beta > \lambda$. z_i is the inertia weight coefficient. These parameters provide a better adaption in the multicast routing presented in the following equation:

$$k_i(t) = 1 - \left[\frac{\min\left(fit\left(pbest_i^m(t-1)\right), fit\left(pbest_i^m(t)\right)\right)}{\max\left(fit\left(pbest_i^m(t-1)\right), fit\left(pbest_i^m(t)\right)\right)}\right] \tag{20}$$

. In Eq. (20), $fit\left(pbest_i^m(t)\right)$ represents the best fitness value of the $pbest_i^m(t)$ at t number of iterations for the i^{th} particle. k_i varies between 0 and 1. A higher value of k_i is faster in evolution speed. If $k_i = 0$, then there is no change in $fit\left(pbest_i^m(t)\right)$, which indicates that a particle does not participate in the evolution process after its current iteration.

$$s = 1 - \left[\frac{\min\left(fit_{tbest}, \hat{fit}_t\right)}{\max\left(F_{tbest}, \hat{fit}_t\right)}\right] \tag{21}$$

, where $\hat{fit}_t$ is the mean fitness of all particles. F_{tbest} denotes the best fitness value at t iterations. Parameter s has values that lie between 0 and 1.

$$z_i = \begin{cases} U - \left[(U-L)\frac{t}{max\_iter}\right] & \text{If it is the best position} \\ 1 & \text{otherwise} \end{cases} \tag{22}$$

, where U denotes the upper bound and L the lower bound for z_i, having its values as 2 and 0.6, respectively, max_iter denotes the maximum allowable number of iterations. Likewise, the fixed value intended for the acceleration coefficient does not provide convergence every time. Therefore, it has been modified dynamically as follows:

$$\phi_1 = \phi_{1i} - \left(\frac{\phi_{1i} - \phi_{1f}}{max\_iter}\right) * iter \tag{23}$$

$$\phi_2 = \phi_{2i} - \left(\frac{\phi_{2i} - \phi_{2f}}{max\_iter}\right) * iter \tag{24}$$

, where $\phi_{1i}, \phi_{1f}, \phi_{2i}$, and ϕ_{2f} are the initial and final values of the cognitive and social component acceleration factors, respectively.

3.4. Necessities and Process of Improvement of BFO through PSO

In BFO, the local search capability in the search space has been incorporated through chemotaxis behavior. Premature convergence has been put an end to through elimination and dispersal, while the convergence speed has been enhanced through the reproduction process. The step size of BFO generates the accuracy and convergence of the global best optima. The fixed step size in classical BFO suffers the following problems: 1) If the size of the step is significantly less, then it takes a greater number of iterations to reach the optimum solution; 2) Again, if the size of the step is very high, then the bacterium is quickly damped at the optimum solution (but the accuracy of the value is low). 3) It takes an extended computational time to achieve convergence. This drawback fails to maintain the good balance between exploration and exploitation capability of the algorithm. Therefore, it is necessary to enhance the convergence speed and accuracy as well as to avoid the premature convergence of a global search algorithm. Such an improved version of PSO has been hybridized with BFO; this proposed algorithm is called IBF/PSO.

4. Multicast Routing Improved by IBF/PSO Algorithm

In this segment, we have shown the use of the IBF/PSO algorithm to solve the multi-constraint multicast routing problem. First, we present the strategy to initialize the bacterium. After initialization, each bacterium is represented as a multicast tree. Then, we use the IBF/PSO algorithm to generate new multicast trees by combining the paths to a different destination and removing the loops.

4.1. Initialization and Representation of Bacteria

To produce a QoS-based multicasting tree starting from a random network, the link delay is set to infinite, which denotes an available bandwidth that is lower than that of the requisite bandwidth. Subsequently, k numbers of links having minimum delay are generated by using the $k-Bellman-Ford$ algorithm. Then, the delay constraint links for each multicasting group destination are chosen out of the k-minimum delay links. For the destination nodes of a multicasting group having $M = \{d_1, d_2, ..., d_m\}$ and a given source node s, a bacterium is represented as $b = \{x_1, x_2, ..., x_m\}$ in our suggested BFO algorithm. In this algorithm, we represent each bacterium as a multicast tree having m components; i.e., $\{x_1, x_2, ..., x_m\}$, where the path to the i^{th} destination is represented by the i^{th} component in a multicast tree. Each component of a bacterium selects a feasible path out of k delay-constrained paths to each of the destinations. Each bacterium is a candidate solution of the QoS multicast tree from the source node to each of the destinations. A bacterium represented as a multicast tree after randomly selecting paths and combining them from the generated k delay constraint paths and removing the loops present, if any. After the random generation of multicasting trees and that chosen by the bacteria, the loop-deletion procedure and fitness function calculation is performed. The flowchart of the BF-oriented PSO is presented in Figure 1.

4.2. Movement of Bacteria and Fitness Function Calculation

The bacteria movement is characterized by tumbling and swimming amid the procedure of chemotaxis. During this process, the j^{th} bacterium fitness, $1 \le j \le B$ in the k^{th} chemotactic step, and $1 \le k \le N_c$ is calculated as $Fitness(P(j,k))$. N_c is the maximum number of iterations. $P(j,k)$ represents the present position of the j^{th} bacterium at the k^{th} chemotactic step. The bacterium position at $P(j,k)$ represents a multicasting tree's $T(s,M)$. The fitness function evaluated stores in J_{last}. $Fitness(P(j,k))$ is defined as follows:

$$\begin{aligned} fit(T(s,M)) = {} & cost(T(s,M)) + \tau_1 \min((DC - Delay(T(s,M))), 0) \\ & + \tau_2 \min((JC - Jitter(T(s,M))), 0) \\ & + \tau_3 \min((BC - Bandwidth(T(s,M))), 0) \end{aligned} \tag{25}$$

.

Predefined weights τ_1 represent the delay, τ_2 represents the delay-jitter, and τ_3 represents the bandwidth separately. Once the fitness function is calculated, the local position of the bacterium will be updated with regard to the next chemotactic step. A bacterium must go through one tumble behavior during its lifetime. The direction meant for the tumble behavior of the bacterium is decided by $\sigma(m,j)$. In the chemotactic phase, a bacterium undergoes a maximum number of swimming steps (N_s). If the bacterium updates, its local position during swimming is better as compared to earlier best position J_{last}, then $J(j,k+1)$ updates J_{last}. Then, $p_{current}$ is evaluated. If parameter $p_{current}$ shows the current location of the bacterium is better as compared to the earlier current location, then the current location of the bacterium is updated. The current location and the fitness function are both updated with each movement of the bacterium. With each chemotactic step, the bacterium moves towards the optimal QoS multicast tree. The*pbest* and *gbest* of each and every bacterium is evaluated after the maximum loop swimming for N_s consecutive steps. If more than three chemotactic phases, there is no change in global best position *pbest*; then, the chemotactic step is stopped and the next phase reproduction starts.

Algorithm 1 Procedure IBF-IPSO

```
Initialization:Population size of bacteria ($B$); steps of chemotactic ($N_c$); length of swimming
    ($N_s$); steps of reproduction ($N_{re}$); steps showing elimination-dispersal ($N_{ed}$); probability
    of elimination-dispersal ($P_{ed}$); step size ($C(j), where j = 1, 2, \ldots, B$); parameters of IPSO
    ($\phi_{1i}, \phi_{2i}, \phi_{1f}, \phi_{2f}, U, L, w_{int}, \phi_3$); bacteria position ($P_i(j,k,l)|i = 1, 2, ..., B$)
 1: Begin
 2: Generate vector $\sigma(p)$
 3: for $l = 1$ to $N_{ed}$ do
 4:     for $k = 1$ to $N_{re}$ do
 5:         for $j = 1$ to $N_c$ do
 6:             for $i = 1$ to $B$ do
 7:                 Calculate fitness function
 8:                 $J(i,j) = \text{Fitness}(P_i(j,k,l))$
 9:                 Memorize best fitness value in $J_{last} = J(i,j)$. Select each bacterium
                    based on best fitness value and used in local best
                    $J_{local}$. $J_{local}(i,j) = J_{last}(i,j)$
10:                 Update fitness value and position
                    $P_i(j+1,k,l) = P(j,k,l) + C(i) * Delta(m,i)$
11:                 $J(i,j+1) = Fitness(P_i(j+1,k,l))$
12:                 Swim: $n = 0$
13:                 while ($n < N_s$) do
14:                     if ($J(i,j+1) < J_{last}$) then
15:                         $J_{last} = J(i,j+1)$
16:                         Update position and fitness value
                            $P_i(j+1,k,l) = P(j,k,l) + C(i) * Delta(m,i)$
17:                         $J(i,j+1) = Fitness(P_i(j+1,k,l))$
18:                         $J(i,j+1) = Fitness(P_i(j+1,k,l))$
19:                         Evaluate local fitness and current position for each bacterium
                            $p_{current-i}(j+1,k,l) = P_i(j+1,k,l)$
20:                         $J_{local}(i,j+1) = J_{last}(i,j+1)$
21:                     else
22:                         $p_{current-i}(j+1,k,l) = P_i(j,k,l)$
23:                         $J_{local}(i,j+1) = J_{last}(i,j+1)$
24:                     end if
25:                     $n = n + 1$
26:                 end while
27:             end for
28:             Evaluate pbest and gbest for bacterium
29:             For each bacterium, update new direction. Calculate velocity $v$ through
                Eq. (15)
30:             $Delta = v$
31:         end for
32:         for $i = 1$ to $B$ do
33:             Reproduction: Health of the bacterium i is computed as follows:
                $J^i_{health} = \min_{1 \leq j \leq N_{c+1}} \{Fitness(P(j,k,l))\}$
34:         end for
35:     end for
36: end for
37: Remove bacteria $B_r$ having highest $J_{health}$ and remainder of bacteria $B_r$ having best value splits
38: Elimination-Dispersal: bacteria with probability $P_{ed}$ will be eliminated and dispersed
39: End
```

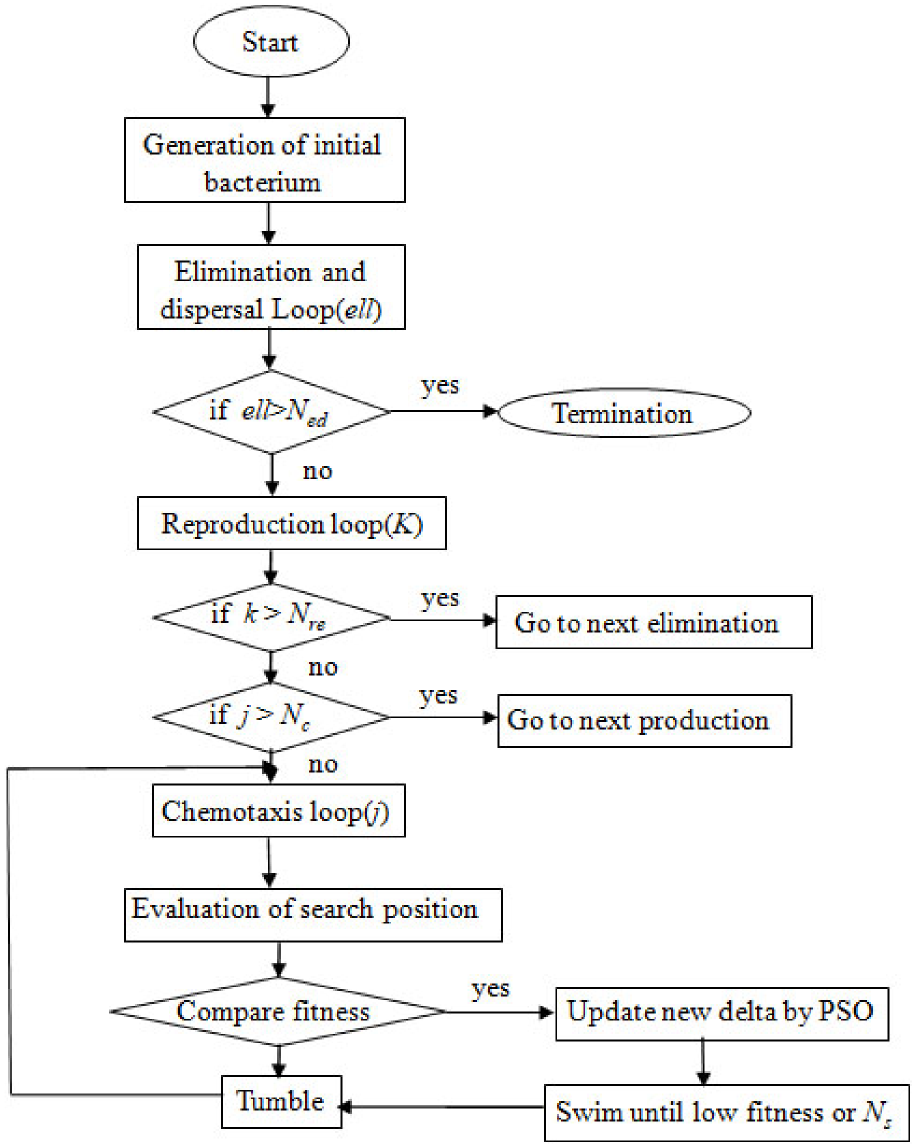


Figure 1. Flowchart above shows bacteria foraging-oriented PSO algorithm

Reproduction: In this phase, the multicast trees that are obtained in the chemotactic steps are arranged and sorted based on their fitness values J^i_{health}. The bacterium having the best fitness value in each chemotactic step is chosen and arranged in ascending order. The highest-fitness-valued bacteria dies, and the remaining bacteria with the best values remain and copy.

$$J_{health}^{i} = \min_{j=1}^{N_{c+1}} \{Fitness(P(i,j))\} \tag{26}$$

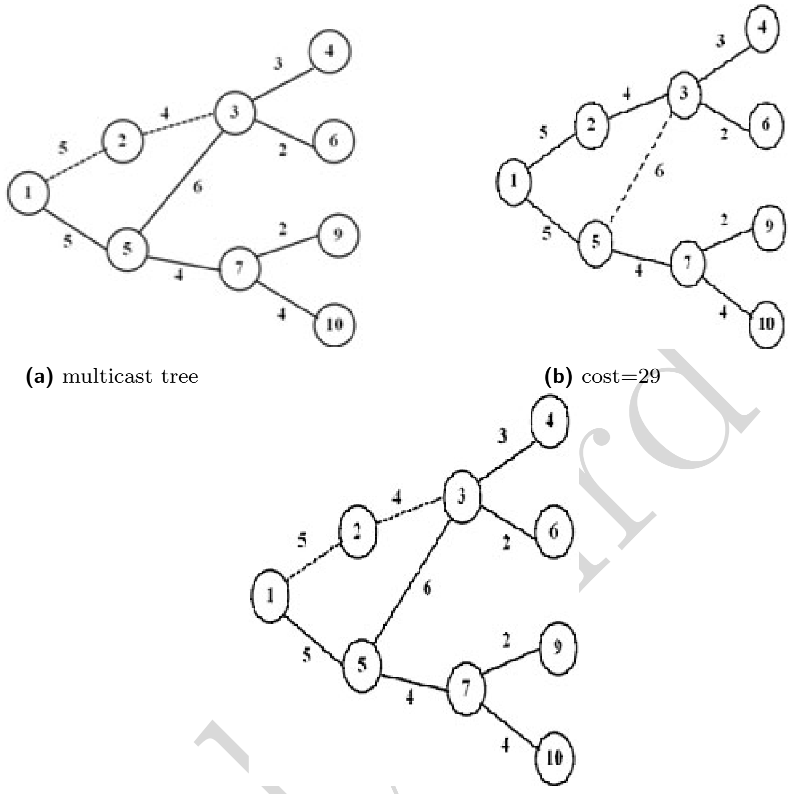
. **Elimination-Dispersal:** In the elimination-dispersal as given in Step 38, the poorest or weakest bacteria are dispersed with a probability of P_{ed}.

4.3. Loop-deletion Procedure

The problem of a routing loop occurs when the data is routed through the same router again and again in an endless circle. This consumes the precious bandwidth that is reserved for normal user traffic. There are several loop-avoidance algorithms such as hop count, split horizon, route poisoning, and hold time timers. In multicasting, intermediate routers forward the data away from the source to the destination over a multicast tree, avoiding routing loops. The reverse path forwarding (RPF) technique is used for loop avoidance by organizing the routing table on a reverse path. This technique is not scalable, as each router keeps the state information of the path. Therefore, the loop-deletion technique is used to avoid loops during the construction of a multicast tree. After the construction of the multicast tree, the multicaster implemented at each intermediate router only forwards the data to the children nodes by replication. This is more scalable than the loop-avoidance technique, as the intermediate routes are only involved in data forwarding and are well-suited to be integrated in the DiffServ framework for QoS provisioning over IP-based networks. In the QPSO algorithm, loop deletion is used after adding one path to the logical tree, whereas the loops are deleted after combining all paths to the tree in the proposed algorithm.

The multicast tree is constructed after combining the paths to each destination and removing the loops. The main motive in the loop-deletion operation is to minimize the cost and satisfy the delay constraint. While the paths from the source to the destination nodes are combined, the occurrence of each edge computed equal to the cost of that edge are divided by the number of occurrences of that edge. The loop occurs only if any node in this combination has an indegree of >1. Thus the cumulative weight of that node from all of its predecessors is computed, and all edges other than the minimum weighted edge are deleted. The loop-deletion method is shown in Figure 2.

For example, let the multicast tree constructed after combining paths 1-2-3-4, 1-5-3-6, 1-5-7-9, and 1-5-7-10 as given in Figure 2a. We get a loop at E {(1,2),(2,3),(3,5),(5,1)}. The destination nodes from source '1' are {4,6,9,10}. Here, three destination nodes {6,9,10} use the same edge (1,5) with weight {1,5}=5. The weight of the path is reduced thrice, and the path with the higher weight is eliminated. The ultimate multicast tree that was generated after our proposed loop-deletion procedure (as shown in Figure 2c) is better than the tree generated after the loop-deletion procedure of the QPSO algorithm (as shown in Figure 2b).



(a) multicast tree **(b)** cost=29

(c) cost=26

Figure 2. Illustration of loop-deletion procedure

5. Experimental Evaluation

Along with its NACO [33], QPSO-based [25], and GA/PSO-based [1] counterparts, the performance of the proposed algorithm is studied through a simulation using a multicast routing simulator (MRSIM) implemented in C++ based on Salama's generator [21]. The network topologies are randomly generated using Waxman's graph generator. The node positions in the network are randomly fixed inside a rectangular area of $(4000 \times 2400) sq.km$. The distance between each pair of nodes is then measured using the Euclidean method. The edges are added up among the (u, v) pair of nodes that has a probability dependent on the distance among the nodes. The likelihood of having an edge among the nodes is $p(u, v) = \beta exp(-l(u, v)/\alpha L$, where $l(u, v)$ denotes the distance between nodes u and v, and L denotes the greatest separation possible among any of the given two points as presented in the graph. The significance of α controls the short links that are presented in an arbitrarily created network topology. The lesser the significance of α decides the higher number of shorter links,

Algorithm 2 loopdeletion(s, M)

```
1: **Begin**
2: $weightsofar(s) = 0$
3: **for** Each $edge(i, j)$ in $P_T(s, d), \forall d \in M$ **do**
4:     $weightsofar(i) = weightsofar(i) + weight(i, j)$
5:     $visited(j) = 1$
6:     $pred(j) = 1$
7:     $indeg(j) = indeg(j) + 1$
8:     $outdeg(i) = outdeg(i) + 1$
9:     $weight(i, j) = weightsofar(i) + weight(i, j)$
10: **end for**
11: **if** $(indeg(j) > 1)$ **then**
12:     **if** $(weight(pred(j), j) < weight(i, j))$ **then**
13:         remove $edge(i, j)$
14:         $outdeg(i) = outdeg(i) - 1$
15:     **else**
16:         remove $edge(pred(j), j)$
17:         $outdeg(pred(j)) = outdeg(pred(j) - 1)$
18:     **end if**
19: **end if**
20: **End**
```

while β controls the links in the arbitrarily created network topology. The lower the significance of β, the higher the number of links, where β and α are set with the values of 0.25 and 0.4, respectively. The delay of the link is the sum of the queuing, propagation, and transmission delay, which is randomly set as a value from 0 to 30. The cost of the link is the current total bandwidth reserved on the link in the network, which is randomly set from 0 to 100. All of the experiments are performed on a machine with an Intel core i7 processor with 3.40 GHz. using a Windows 8.1-based operating system with 4 GB of RAM. We found that our proposed algorithm outperforms the other existing algorithms after being verified through an analysis of the simulation results obtained after executing the programs for 50 iterations. The average of the results is recorded to evaluate the performance comparisons. If there is no better multicast tree generated after two consecutive iterations, then the circulation terminates. We use Waxman's graph generation algorithm to generate random network topologies for our experiments.

The source node and set of destination nodes are randomly selected from the nodes in the network. We ran the simulation 50 times to record the result of our proposed algorithm along with other existing algorithms. During each run, a random network is generated, and the multicast group members are also randomly selected. The average of the results is recorded to evaluate the performance of the algorithms. First, the k-Bellman Ford algorithm is implemented to find the k-least delay paths from the source to each destination in the multicast group. The path information is

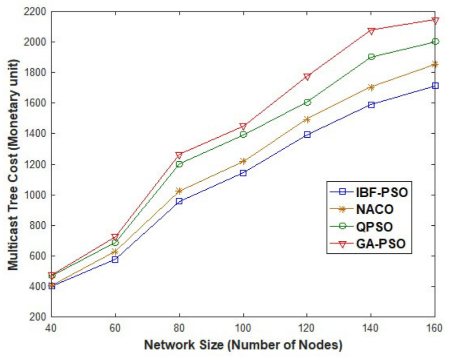


Figure 3. Multicast tree cost verses network size with 10% nodes as destinations

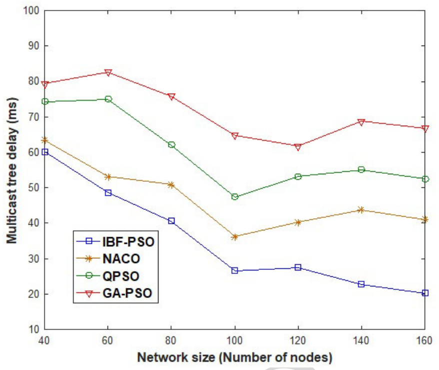


Figure 4. Multicast tree delay verses network size with 10% nodes as destinations

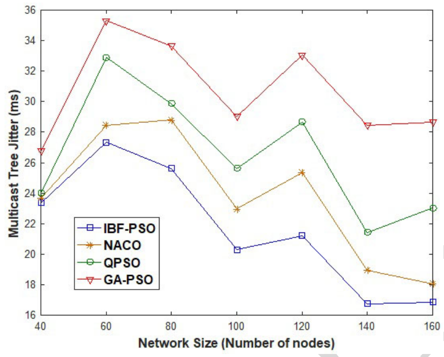


Figure 5. Multicast tree jitter verses network size with 10% nodes as destinations

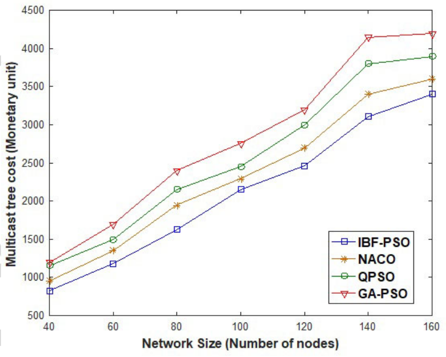


Figure 6. Multicast tree cost verses network size with 25% nodes as destinations

maintained in a table. The paths to the destinations are combined, and the loop-deletion procedure is applied for the construction of the multicast tree.

5.1. Performance with Network Size

In this segment, we present a performance comparison of the existing algorithms with our proposed algorithm by varying the network sizes. We select seven different network sizes in terms of varying the nodes from 40-160. The terminal nodes are specified as 10% and 25% of the total nodes. We ran the simulations for 50 iterations to provide a reasonable comparison among the algorithms. The significance of the source and destination nodes are considered for studying the performance shown by the various algorithms. In each iteration, we generate a new graph and randomly select the source and destination nodes.

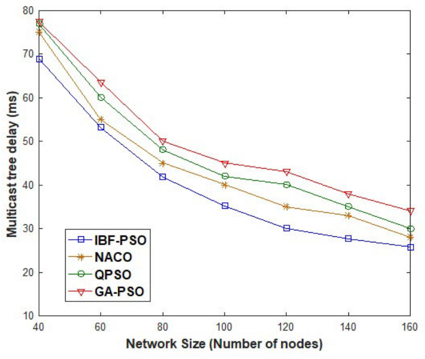


Figure 7. Multicast tree delay verses network size with 25% nodes as destinations

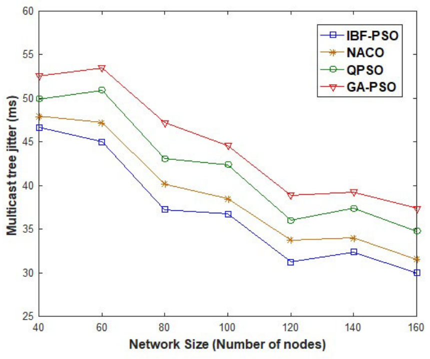


Figure 8. Multicast tree jitter verses network size with 25% nodes as destinations

Table 1
Multicast Tree having 10% nodes as destinations with different network size

Network Size	Multicast Tree Cost				Multicast Tree Delay				Multicast Tree Jitter			
	IBF/PSO	NACO	QPSO	GA/PSO	IBF/PSO	NACO	QPSO	GA/PSO	IBF/PSO	NACO	QPSO	GA/PSO
40	400	407	468	474	60	63	74	79	23	24	24	27
60	577	630	688	728	49	53	75	83	27	28	33	35
80	958	1027	1202	1264	41	51	62	76	26	29	30	34
100	1144	1220	1394	1450	26	36	47	65	20	23	26	29
120	1393	1495	1607	1777	28	40	53	62	21	25	29	33
140	1590	1706	1901	2079	23	44	55	69	17	19	21	28
160	1714	1856	2001	2145	20	41	52	67	18	18	23	29

Table 2
Multicast Tree having 25% nodes as destinations with different network size

Network Size	Multicast Tree Cost				Multicast Tree Delay				Multicast Tree Jitter			
	IBF/PSO	NACO	QPSO	GA/PSO	IBF/PSO	NACO	QPSO	GA/PSO	IBF/PSO	NACO	QPSO	GA/PSO
40	821	952	1156	1197	69	75	77	78	47	48	50	53
60	1179	1356	1495	1694	53	55	60	64	45	47	51	53
80	1629	1945	2152	2397	42	45	48	50	37	40	43	47
100	2152	2291	2449	2754	35	40	42	45	37	38	42	45
120	2461	2698	2995	3196	30	35	40	43	31	34	36	39
140	3107	3401	3799	4148	28	33	35	38	32	34	37	39
160	3399	3600	3895	4194	26	28	30	34	30	32	35	37

Table 3
Multicast Tree Cost Delay and Jitter with group size of 100

Network Size	Multicast Tree Cost				Multicast Tree Delay				Multicast Tree Jitter			
	IBF/PSO	NACO	QPSO	GA/PSO	IBF/PSO	NACO	QPSO	GA/PSO	IBF/PSO	NACO	QPSO	GA/PSO
10	1146	1252	1328	1468	30	33	35	36	22	23	27	30
20	1731	1950	2164	2424	38	40	43	44	34	36	38	41
30	2324	2694	2948	3243	32	33	35	38	36	41	43	44
40	2734	3199	3498	4164	36	37	38	40	45	49	50	54
50	3066	3499	3750	4253	38	39	42	45	52	55	57	58
60	3409	4000	4097	4465	38	40	45	47	55	58	59	62
70	3883	4491	4797	5014	36	39	42	46	57	62	63	66
80	4091	4800	4996	5363	38	45	44	49	63	66	70	77

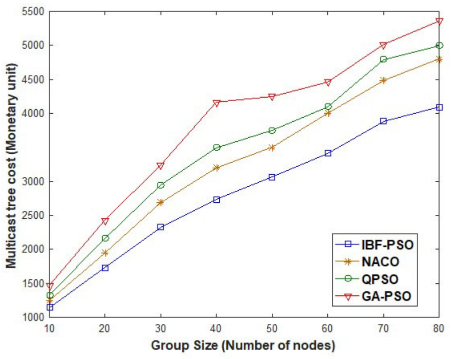


Figure 9. Multicast tree cost verses group size with network size = 100

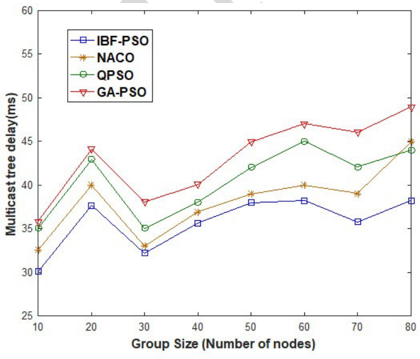


Figure 10. Multicast tree delay verses group size with network size = 100

Figures 3 and 4 show the multicast trees costs that are generated with our algorithm, NACO, QPSO, and PSO/GA for increasing numbers of nodes, with 10% and 25% of the nodes considered the destinations, respectively. Figures 5 and 6 show the maximal end-to-end delay that is obtained from the multicast tree obtained by our algorithm, NACO, QPSO, and PSO/GA. The delay jitter of the multicast trees that are generated with our proposed algorithm, NACO, QPSO, and PSO/GA for various network sizes are shown in Figures 7 and 8. From Figures 3 through 8, it can be observed that the generated multicast tree using the proposed algorithm outperforms the multicast trees that are generated by NACO, QPSO, and PSO/GA in terms of the multicast tree cost, maximal end-to-end delay, and delay jitter. The figures show that the performance of our proposed algorithm significantly improves with increased network sizes. This is because the search power of the BFO algorithm optimally combines the paths to the set of destinations. The loop-deletion process proposed in our algorithm removes the edges with high costs without violating the bandwidth, delay, loss rate, or delay jitter constraints. Furthermore, the PSO technique used in our algorithm increases the convergence speed. The performance of our algorithm increases significantly with an increased number of nodes. This shows the scalability of our algorithm as related to the network size.

5.2. Performance with Number of Destinations

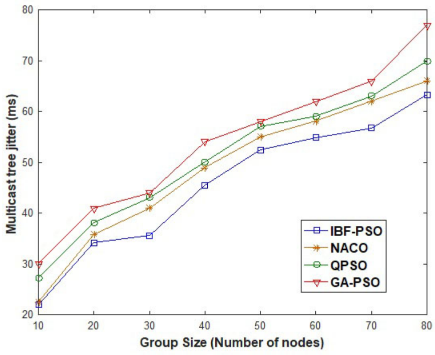


Figure 11. Multicast tree jitter verses group size with network size = 100

This segment presents a performance comparison of the proposed algorithm with the other pre-existing algorithms by varying the numbers of destinations for a fixed network size. We varied the number of destinations from 10% to 80% of the number of nodes present in the network. We generated a network of 100 nodes randomly in each iteration and selected the source and destination nodes randomly as well. We ran the simulations for 50 iterations to provide a fair comparison among the algorithms.

Figures 9 through Figure 11 show the performance of the multicast trees that are generated by our proposed algorithm, NACO, QPSO, and PSO/GA for increasing numbers of destination nodes in terms of multicast tree cost, maximal end-to-end delay, and delay jitter, respectively. From the figures, it can observed that the difference between the cost, delay, and delay jitter obtained by our algorithm and the other existing algorithms is quite significant with the number of iterations. Hence, the proposed algorithm significantly outperforms the existing algorithms considering different numbers of destinations.

5.3. Execution Time with Number of Nodes

The execution time of our proposed algorithm along with the other existing algorithms is presented in Table 4. It can be observed that our algorithm consumes less time as compared to the others; this is because the loop-deletion algorithm used in our paper takes less time as compared to the other existing algorithms. The proposed loop-deletion algorithm is applied after combining all of the paths to the destinations.

Table 4
Multicast tree execution time with different network sizes

Network Size	Multicast execution time having 10% of nodes as destinations				Multicast execution time having 25% of nodes as destinations			
	IBF/PSO	NACO	QPSO	GA/PSO	IBF/PSO	NACO	QPSO	GA/PSO
20	39.42	41.3	42.8	44.7	66.32	70.62	74.76	79.22
40	78.2	81.4	86.2	95.4	137.41	143.42	152.52	168.43
60	117.3	125.8	134.5	151.2	208.84	226.35	237.65	254.13
80	185.4	199.6	214.3	237.4	347.32	364.71	386.31	414.06
100	277.6	297.1	312.4	344.2	491.27	507.61	531.08	497.84
120	368.5	404.6	425.3	461.8	695.83	716.11	749.01	718.52
140	481.8	521.3	547.5	586.7	903.41	924.53	958.75	926.03
160	593.2	637.4	674.2	733.2	1148.65	1162.36	1196.14	1171.19

This considers all ages of the multicast tree one-by-one and calculates the indegree of each node in the multicast tree. This requires $O(n^2)$ time in the worst-case scenario, as there are a maximum of n^2 edges in a network of n nodes. If the indegree of the nay node is more than 1, then the edges incident on that node that have higher costs are removed. In QPSO, the loop-deletion process is applied after combining one path to the existing multicast tree; this requires $O(mn^2)$ time, where m is the number of destinations. Therefore, the loop-deletion procedure removes the high-cost edges from the loops in the multicast tree, improving the solution quality in terms of multicast tree cost, whereas an additional overhead of CPU time spent for loop deletion.

6. Conclusions and Future scope

This paper proposed an improved version of the hybridized bacteria foraging and particle swarm optimization algorithm (IBF/PSO) for QoS multicast tree generation. The primary focus of the proposed work is to construct a multicast QoS tree with multiple constraints such as end-to-end delay, bandwidth, delay jitter, and cost. The proposed algorithm uses an improved BFO method to explore the search space and combines the set of paths sensibly for constructing a multicast tree in each iteration. Furthermore, the loop-deletion procedure is used in the algorithm to generate a better cost-sensitive tree than its counterparts. In order to maintain a tradeoff between the convergence speed and the cost effectiveness, we used PSO to adjust the BF parameters. This algorithm is well-suited to support QoS multicast communication over Mobile Adhoc Networks, sensor networks, the Internet, and other high-performance networks. However, some additional parameters such as the mobility and energy constraint nature of mobile/sensors are required to be considered in addition to the QoS parameters. The proposed work was simulated using a multicast routing simulator

(MRSIM) tool. The experimental results on different network sizes with variable numbers of destinations has shown that the proposed BF/PSO algorithm generates minimum-cost QoS-aware multicast trees as compared to the trees that are generated by QPSO, PSO/GA, and NACO. In the future, the multicast communication can be extended to the Mobile Adhoc Network(MANET), in which the major challenges will be due to the mobility, energy constraints, reliability, and lack of bandwidth in the mobile nodes. This work can be further extended to support QoS multicast communication over MANET, which is a challenging problem. The multicast routing problem can also be formulated as a multi-objective problem in which a number of QoS parameters are required to be simultaneously optimized.

References

[1] Abdel-Kader R.F.: Hybrid discrete PSO with GA operators for efficient QoS-multicast routing. In: *Ain Shams Engineering Journal*, vol. 2(1), pp. 21–31, 2011.

[2] Ahlgren B., Dannewitz C., Imbrenda C., Kutscher D., Ohlman B.: A survey of information-centric networking. In: *IEEE Communications Magazine*, vol. 50(7), pp. 26–36, 2012.

[3] Eberhart R., Kennedy J.: A new optimizer using particle swarm theory. In: *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39–43. Ieee, 1995.

[4] Felber P., Kropf P., Schiller E., Serbu S.: Survey on load balancing in peer-to-peer distributed hash tables. In: *IEEE Communications Surveys & Tutorials*, vol. 16(1), pp. 473–492, 2014.

[5] Forsati R., Haghighat A.T., Mahdavi M.: Harmony search based algorithms for bandwidth-delay-constrained least-cost multicast routing. In: *Computer Communications*, vol. 31(10), pp. 2505–2519, 2008.

[6] Gong B., Li L., Wang X.: Multicast routing based on ant algorithm with multiple constraints. In: *2007 International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1945–1948. IEEE, 2007.

[7] Haghighat A.T., Faez K., Dehghan M., Mowlaei A., Ghahremani Y.: GA-based heuristic algorithms for bandwidth-delay-constrained least-cost multicast routing. In: *Computer Communications*, vol. 27(1), pp. 111–127, 2004.

[8] Hwang R.H., Do W.Y., Yang S.C.: Multicast routing based on genetic algorithms. In: *J. Inf. Sci. Eng.*, vol. 16(6), pp. 885–901, 2000.

[9] Jiang D., Huo L., Li Y.: Fine-granularity inference and estimations to network traffic for SDN. In: *PloS one*, vol. 13(5), p. e0194302, 2018.

[10] Jiang D., Huo L., Lv Z., Song H., Qin W.: A joint multi-criteria utility-based network selection approach for vehicle-to-infrastructure networking. In: *IEEE Transactions on Intelligent Transportation Systems*, (99), pp. 1–15, 2018.

[11] Jiang D., Li W., Lv H.: An energy-efficient cooperative multicast routing in multi-hop wireless networks for smart medical applications. In: *Neurocomputing*, vol. 220, pp. 160–169, 2017.

[12] Jiang D., Xu Z., Li W., Chen Z.: Network coding-based energy-efficient multicast routing algorithm for multi-hop wireless networks. In: *Journal of Systems and Software*, vol. 104, pp. 152–165, 2015.

[13] Jiang D., Xu Z., Li W., Yao C., Lv Z., Li T.: An energy-efficient multicast algorithm with maximum network throughput in multi-hop wireless networks. In: *Journal of communications and networks*, vol. 18(5), pp. 713–724, 2016.

[14] Jiang D., Zhang P., Lv Z., Song H.: Energy-efficient multi-constraint routing algorithm with load balancing for smart city applications. In: *IEEE Internet of Things Journal*, vol. 3(6), pp. 1437–1447, 2016.

[15] Li W., Li K., Huang Y., Yang S., Yang L.: A EA-and ACA-based QoS multicast routing algorithm with multiple constraints for ad hoc networks. In: *Soft Computing*, vol. 21(19), pp. 5717–5727, 2017.

[16] Molnár M., Bellabas A., Lahoud S.: The cost optimal solution of the multi-constrained multicast routing problem. In: *Computer Networks*, vol. 56(13), pp. 3136–3149, 2012.

[17] Passino K.M.: Biomimicry of bacterial foraging for distributed optimization and control. In: *IEEE control systems magazine*, vol. 22(3), pp. 52–67, 2002.

[18] Patel M.K., Kabat M.R., Tripathy C.R.: A hybrid ACO/PSO based algorithm for QoS multicast routing problem. In: *Ain Shams Engineering Journal*, vol. 5(1), pp. 113–120, 2014.

[19] Peng B., Li L.: A method for QoS multicast routing based on genetic simulated annealing algorithm. In: *International Journal of Future Generation Communication and Networking*, vol. 5(1), pp. 43–60, 2012.

[20] Qu R., Xu Y., Castro J.P., Landa-Silva D.: Particle swarm optimization for the Steiner tree in graph and delay-constrained multicast routing problems. In: *Journal of Heuristics*, vol. 19(2), pp. 317–342, 2013.

[21] Salama H.F., Reeves D.S., Viniotis Y.: Evaluation of multicast routing algorithms for real-time communication on high-speed networks. In: *IEEE Journal on Selected Areas in Communications*, vol. 15(3), pp. 332–345, 1997.

[22] Sangeetha J., Keerthiraj N., Murthy K.B., Rustagi P.R.: A new approach for analyzing the performance of the WiMAX networks based on QoS traffic prediction routing protocol using gene expression programming. In: *International Journal of Applied Metaheuristic Computing (IJAMC)*, vol. 7(2), pp. 16–38, 2016.

[23] Sharma S., Kumar S., Singh B.: AntMeshNet: An ant colony optimization based routing approach to wireless mesh networks. In: *International Journal of Applied Metaheuristic Computing (IJAMC)*, vol. 5(1), pp. 20–45, 2014.

[24] Shi L., Li L., Zhao W., Qu B.: A Delay-Constrained Multicast Routing Algorithm Based on the Ant Colony Algorithm. In: *Proceedings of the International Conference on Information Engineering and Applications (IEA) 2012*, pp. 875–882. Springer, 2013.

[25] Sun J., Fang W., Wu X., Xie Z., Xu W.: QoS multicast routing using a quantum-behaved particle swarm optimization algorithm. In: *Engineering Applications of Artificial Intelligence*, vol. 24(1), pp. 123–131, 2011.

[26] Wang H., Meng X., Li S., Xu H.: A tree-based particle swarm optimization for multicast routing. In: *Computer Networks*, vol. 54(15), pp. 2775–2786, 2010.

[27] Wang H., Shi Z., Li S.: Multicast routing for delay variation bound using a modified ant colony algorithm. In: *Journal of Network and Computer Applications*, vol. 32(1), pp. 258–272, 2009.

[28] Wang H., Xu H., Yi S., Shi Z.: A tree-growth based ant colony algorithm for QoS multicast routing problem. In: *Expert Systems with Applications*, vol. 38(9), pp. 11787–11795, 2011.

[29] Wang Z., Crowcroft J.: Quality-of-service routing for supporting multimedia applications. In: *IEEE Journal on selected areas in communications*, vol. 14(7), pp. 1228–1234, 1996.

[30] Xylomenos G., Ververidis C.N., Siris V.A., Fotiou N., Tsilopoulos C., Vasilakos X., Katsaros K.V., Polyzos G.C.: A survey of information-centric networking research. In: *IEEE Communications Surveys & Tutorials*, vol. 16(2), pp. 1024–1049, 2014.

[31] Yadav A.K., Tripathi S.: QMRPRNS: Design of QoS multicast routing protocol using reliable node selection scheme for MANETs. In: *Peer-to-Peer Networking and Applications*, vol. 10(4), pp. 897–909, 2017.

[32] Yen J.Y.: Finding the k shortest loopless paths in a network. In: *management Science*, vol. 17(11), pp. 712–716, 1971.

[33] Yin P.Y., Chang R.I., Chao C.C., Chu Y.T.: Niched ant colony optimization with colony guides for QoS multicast routing. In: *Journal of Network and Computer Applications*, vol. 40, pp. 61–72, 2014.

[34] Younes A.: An ant algorithm for solving qos multicast routing problem. In: *International Journal of Computer Science and Security (IJCSS)*, vol. 5(1), pp. 156–167, 2011.

[35] Zhang K., Wang H., Liu F.: Multicast routing for delay and delay variation bounded Steiner tree using simulated annealing. In: *Proceedings. 2005 IEEE Networking, Sensing and Control, 2005.*, pp. 682–687. IEEE, 2005.

[36] Zhang L., Cai L.b., Li M., Wang F.h.: A method for least-cost QoS multicast routing based on genetic simulated annealing algorithm. In: *Computer Communications*, vol. 32(1), pp. 105–110, 2009.

Affiliations

Satya Prakash Sahoo
Veer Surendra Sai University of Technology, Department of Computer Science & Engineering, sahoo.satyaprakash@gmail.com

Manas Ranjan Kabat
Veer Surendra Sai University of Technology, Department of Computer Science & Engineering, kabatmanas@gmail.com

Received: 11.12.2018
Revised: 18.04.2019
Accepted: 19.04.2019