Engineering Applications of Artificial Intelligence 23 (2010) 806–819

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Genetic algorithms with immigrants schemes for dynamic multicast problems in mobile ad hoc networks

Hui Cheng *, Shengxiang Yang

Department of Computer Science, University of Leicester, University Road, Leicester LE1 7RH, UK

ARTICLE INFO

Article history:
Received 31 May 2009
Received in revised form
19 November 2009
Accepted 16 January 2010
Available online 9 February 2010

Keywords:
Mobile ad hoc network
Dynamic multicast
Genetic algorithm
Immigrants scheme
Dynamic optimization

ABSTRACT

In this paper, the problem of dynamic quality-of-service (QoS) multicast routing in mobile ad hoc networks is investigated. Lots of interesting works have been done on multicast since it is proved to be a NP-hard problem. However, most of them consider the static network scenarios only and the multicast tree cannot adapt to the topological changes. With the advancement in communication technologies, more and more wireless mobile networks appear, e.g., mobile ad hoc networks (MANETs). In a MANET, the network topology keeps changing due to its inherent characteristics such as the node mobility and energy conservation. Therefore, an effective multicast algorithm should track the topological changes and adapt the best multicast tree to the changes accordingly. In this paper, we propose to use genetic algorithms with immigrants schemes to solve the dynamic QoS multicast problem in MANETs. MANETs are considered as target systems because they represent a new generation of wireless networks. In the construction of the dynamic network environments, two models are proposed and investigated. One is named as the general dynamics model in which the topologies are changed due to that the nodes are scheduled to sleep or wake up. The other is named as the worst dynamics model, in which the topologies are altered because some links on the current best multicast tree are removed. Extensive experiments are conducted based on both of the dynamic network models. The experimental results show that these immigrants based genetic algorithms can quickly adapt to the environmental changes (i.e., the network topology changes) and produce high quality solutions following each change.

1. Introduction

A mobile ad hoc network (MANET) (Perkins, 2001; Siva Ram Murthy and Manoj, 2004; Toh, 2002) is a self-organizing and self-configuring multi-hop wireless network, which is comprised of a set of mobile hosts (MHs) that can move around freely and cooperate in relaying packets on behalf of one another. A MANET supports robust and efficient operations by incorporating the routing functionality into MHs. In multi-hop networks, routing is one of the most important issues that has a significant impact on the network's performance. In a MANET, each mobile node is a router and forwards packets on behalf of other nodes. Multi-hop forwarding paths are established for nodes beyond the direct wireless communication range. Routing protocols for MANETs must discover such paths and maintain connectivity when links in these paths break due to effects such as the node movement, battery drainage, radio propagation, and wireless interference.

Multicast (Wang and Hou, 2000; Kuipers and Mieghem, 2002; Wang et al., 2006) is an important network service, which is the delivery of information from a source to multiple destinations simultaneously using the most efficient strategy to deliver the messages over each link of the network only once, creating copies only when the links to the destinations split. It provides underlying network support for collaborative group communications, such as the video conference, distant education, and content distribution. Group communications in MANETs are also important in the support of mobile nodes to work in a cooperative way (Law et al., 2005). Quality-of-service (QoS) requirements (Xiao and Ni, 1999) proposed by different network applications are often versatile. Among them, the end-to-end delay (Parsa et al., 1998; Jia, 1998) is a pretty important QoS metric since the real-time delivery of multimedia data is required. An efficient QoS multicast algorithm should construct a multicast routing tree, by which the data can be transmitted from the source to all the destinations with a guaranteed QoS. The multicast tree cost, which is used to evaluate the utilization of network resources, is also an important metric especially in wireless mobile networks where limited radio resources are available.

In this paper, the QoS multicast routing in MANETs is investigated. The QoS multicast routing problem involves a classical combinatorial optimization problem arising in many design and planning contexts (Oliveira and Pardalos, 2005; Tyan et al., 2003; Xue, 2003). In a MANET, the network topology keeps changing due to its inherent characteristics, such as the node

* Corresponding author. Tel.: +44 116 2525295.
E-mail address: hc118@le.ac.uk (H. Cheng).

0952-1976/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2010.01.021

mobility and energy conservation. Therefore, the multicast problem in MANETs turns out to be a dynamic optimization problem (DOP). An effective multicast algorithm should track the topological changes and adapt the best multicast tree to the changes accordingly. There are mainly two types of algorithms for the multicast problem: the deterministic algorithms and the search heuristics. Given the multicast request, only one multicast tree is constructed for a given topology by a deterministic algorithm, e.g., the shortest path tree (SPT) (Narvaez et al., 2000) algorithm. However, by the search heuristics, such as genetic algorithms (GAs) (Gen and Cheng, 2000) and simulated annealing (SA) algorithms (Wang et al., 2006), lots of multicast trees are searched and the best one is selected as the final result.

All the deterministic algorithms have polynomial time complexity. Therefore, they will be effective in fixed infrastructure wireless or wired networks. But, they exhibit an unacceptably high computational complexity for real-time communications involving rapidly changing network topologies (Ahn et al., 2001; Ahn and Ramakrishna, 2002). Therefore, for the dynamic multicast problem in a changing network environment, the search heuristics are worthy of investigation. In recent years, studying evolutionary algorithms (EAs) for DOPs has attracted a growing interest due to its importance in EA's real world applications (Yang and Yao, 2008). The simplest way of addressing DOPs is to restart EAs from scratch whenever an environment change is detected. Although the restart scheme really works for some cases (Yang and Yao, 2005), for many DOPs it is more efficient to develop other approaches that make use of knowledge gathered from old environments. Over the years, several approaches have been developed for GAs to address dynamic environments (Branke, 2002; Morrison, 2004; Weicker, 2003), such as maintaining diversity during the run via random immigrants (Grefenstette, 1992; Vavak and Fogarty, 1996) and increasing diversity after a change.

In this paper, we adapt and investigate several genetic algorithms that are developed to deal with general DOPs to solve the dynamic multicast routing problem. First, we design the components of the standard GA specifically for the dynamic multicast problem. Then, we integrate several immigrants schemes into the GA to enhance its searching capacity of the optimal multicast tree in dynamic environments. Once the topology is changed, the new immigrants can help guide the search of good solutions in the new environment. For the comparison purpose, we also implement two traditional GA schemes, i.e., Standard GA and Restart GA, as the peer algorithms. By simulation experiments, these GAs are evaluated under different parameter settings to find the best combinations. More importantly, they are evaluated under various settings of dynamic environments to see the performance and find the best match between algorithms and environmental characteristics. Generally speaking, the investigated well-designed GAs work well in the dynamic real-world networks.

The rest of this paper is organized as follows. Section 2 discusses related work. The network model and problem model are provided in Section 3. The design of a GA for the multicast problem is presented in Section 4. We describe the GAs with immigrants schemes for the dynamic multicast routing problem in Section 5. Section 6 presents the extensive experimental results and analysis. Finally, Section 7 concludes this paper.

2. Related work

Multicast routing trees produced by deterministic algorithms can be classified into two types, i.e., Steiner minimum tree (SMT) (Hwang and Richards, 1992) and shortest path tree (Narvaez et al., 2000). An SMT is also the minimum-cost multicast tree. An SPT is constructed by applying the shortest path algorithm to find the shortest (e.g., minimum cost or delay) path from the source to each destination and then merging them. The problem of finding an SMT has been proved to be NP-complete (Jia et al., 1997) and lots of approximation algorithms (Aharoni and Cohen, 1998; Helvig et al., 2000; Robins and Zelikovsky, 2000) have been developed. An SPT provides a good solution for finding delay-constrained multicast tree because it determines the minimum delay path from the source to each destination. Inspired by SMT and SPT, many heuristic algorithms (Parsa et al., 1998; Jia, 1998; Khuller et al., 1995) have been proposed to construct a QoS-aware multicast tree by making a tradeoff between them. QoS multicast routing is still a challenging problem due to its intractability and comprehensive application backgrounds. The research on it has lasted for decades and is still going on.

Intelligent search heuristics is a type of promising techniques to solve combinatorial optimization problems (Papadimitriou and Steiglitz, 1998) including the SMT problem. GAs are a class of representative intelligent global search heuristics. GAs are a type of stochastic meta-heuristic optimization methods that model the biological principles of Darwinian theory of evolution and Mendelian principles of inheritance (Goldberg, 1989; Holland, 1975). GAs have been extensively used in solving the QoS multicast problems in various networks such as the wired multimedia networks (Wang et al., 2006) and optical networks (Din, 2005). As one of our previous work (Wang et al., 2006), we also developed a unified framework for achieving QoS multicast trees using intelligent search heuristics and proposed three QoS multicast algorithms based on GAs, simulated annealing, and tabu search, separately.

In Wang et al. (2006), the binary encoding is adopted where each bit of the binary string corresponds to a different node in the network. For each binary string, a graph G' is derived from the network topology G by including all the nodes appearing in the string and the links connecting these nodes. Then, the minimum spanning tree T of G' acts as the candidate multicast tree represented by the binary string. This encoding method is a bit complicated and each binary string cannot directly represent a candidate solution. A multicast tree is a union of the routing paths from the source to each receiver. Hence, it is a natural choice to adopt the path-oriented encoding method (Ahn and Ramakrishna, 2002; Din, 2005) instead of the binary encoding.

In MANETs, a number of multicast routing protocols, using a variety of basic routing algorithms and techniques, have been proposed over the past few years (Cordeiro et al., 2003). However, they mainly focus on the discovery of the optimal multicast forwarding structure (i.e., tree or mesh) spanning mobile nodes and do not consider the everlasting changes in the network topologies. Topology dynamics is the inherent characteristics in wireless mobile networks. For example, at time T_1, the network topology is G_1. At time T_2, the network topology may change to G_2. Although G_1 and G_2 are different, they are highly relevant since each change alters part of the topology only. Therefore, the solutions obtained on G_1 could benefit the search of good solutions on G_2. An effective multicast algorithm should track the topological changes and adapt the multicast trees to the changes accordingly. We are not aware of any other work that considers the dynamic multicast routing in the environment where the network topology keeps changing in a contiguous way, although there are quite a few works that are related to some relevant aspects. For example, some researchers have investigated the multicast problem where the dynamic group membership exists (Adelstein et al., 2003; Yong et al., 2008). In a dynamic group, nodes are allowed to join or leave it.

3. Network and problem model

In this section, the model of a MANET is first presented and then the dynamic multicast routing problem is formulated. We consider a MANET operating within a fixed geographical region. It is modelled by a undirected and connected topology graph $G_0(V_0, E_0)$, where V_0 represents the set of wireless nodes (i.e., routers) and E_0 represents the set of communication links connecting two neighboring routers falling into the radio transmission range. A communication link (i, j) cannot be used for packet transmission unless both node i and node j have a radio interface each with a common channel. However, the channel assignment is beyond the scope of this paper. In addition, message transmission on a wireless communication link will incur remarkable delay and cost.

We summarize the notations that are used throughout this paper as follows:

- $G_0(V_0, E_0)$, the initial MANET topology graph.
- $G_i(V_i, E_i)$, the MANET topology graph after the i th change.
- s, the source node of the multicast request.
- $R = \{r_0, r_1, \ldots r_m\}$, the set of receivers of the multicast request.
- $T_i(V_{T_i}, E_{T_i})$, a multicast tree with nodes V_{T_i} and links E_{T_i}.
- $P_{T_i}(s, r_j)$, a path from s to r_j on the tree T_i.
- d_l, the transmission delay on the communication link l.
- c_l, the cost on the communication link l.
- C_{T_i}, the cost of the tree T_i.
- $\Delta(P_i)$, the total transmission delay on the path P_i.

The dynamic multicast routing problem can be informally described as follows. Initially, given a MANET consisting of wireless routers, a delay upper bound, and a multicast communication request from a source node to a set of receiver nodes, we wish to find a delay-bounded least cost loop-free multicast tree on the topology graph.[1] Then, after each topology change, the objective of our problem is to quickly find the new delay-constrained least cost acyclic tree.

It is an extremely difficult job to completely model the network dynamics in a single way. Here, we propose two models to describe it and they are named as the general dynamics model and the worst dynamics model, respectively. In the general model, periodically or stochastically, due to energy conservation or other reasons, some nodes are scheduled to sleep or some sleeping nodes are scheduled to wake up. Therefore, the network topology changes from time to time. Since in most cases, the selected nodes may not belong to the present best multicast tree, the topological changes have relatively moderate effect on the routing problem. In the worst model, each change is generated manually by removing a few links on the present best multicast tree. Thus, the topological changes will destroy the present best solution and thereby cause the worst effect on the problem. Although these two models cannot cover the full cases of network dynamics, they correspond to the general scenario and the worst scenario, respectively. Based on these two representative models, the multicast routing problem can be investigated in a relatively thorough way.

More formally, we consider a MANET $G(V, E)$ and a multicast communication request from the source node s to the set of receivers R with the delay upper bound Δ. The *dynamic delay-constrained multicast routing problem* is to find a series of trees $\{T_i | i \in \{0, 1, \ldots\}\}$ over a series of graphs $\{G_i | i \in \{0, 1, \ldots\}\}$, which satisfy the delay constraint as shown in Eq. (1) and have the least tree cost as shown in Eq. (2):

$$\max_{r_j \in R} \left\{ \sum_{l \in P_T(s, r_j)} d_l \right\} \leq \Delta, \tag{1}$$

$$C(T_i) = \min_{T \in G_i} \left\{ \sum_{l \in T(V_T, E_T)} c_l \right\}. \tag{2}$$

4. Design of the GA for the dynamic QoS multicast problem

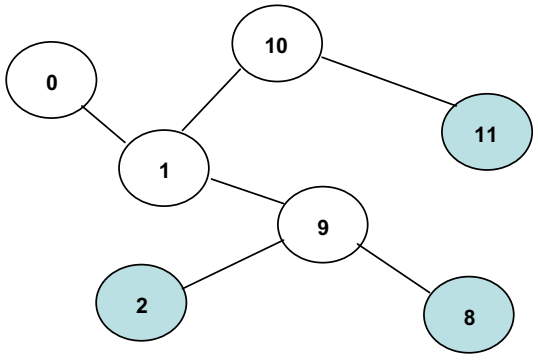
4.1. Genetic representation

As mentioned before, a multicast tree is a union of the routing paths from the source to each receiver. Hence, it is a natural choice to adopt the path-oriented encoding method (Ahn and Ramakrishna, 2002; Din, 2005). A routing path is encoded by a string of positive integers that represent the IDs of nodes through which the path passes. Each locus of the string represents an order of a node. The first locus is for the source and the last one is for the receiver. The length of a routing path should not exceed the maximum length $|V|$, where V is the set of nodes in the network.

For a multicast tree T spanning the source s and the set of receivers R, there are $|R|$ routing paths all originating from s. Therefore, a tree is encoded by an integer array in which each row encodes a routing path along the tree. For example, for a tree T that spans s and R, the j-th row in the corresponding array A lists up node IDs on the routing path from s to r_j along T. Therefore, A is an array of $|R|$ rows. Fig. 1 illustrates a multicast tree and its representation in an array. All the solutions are encoded under the delay constraint. In case the delay constraint is violated, the encoding process is usually repeated so that it is satisfied.

4.2. Population initialization

In a GA, each chromosome corresponds to a potential solution. The initial population Q is composed of a certain number, denoted as q, of chromosomes. A general method to initialize the population is to explore the genetic diversity. That is, for each chromosome, all its routing paths are randomly generated. We start to search a random path from s to $r_j \in R$ by randomly selecting a node v_1 from $N(s)$, the neighborhood of s. Then, we



0	1	9	2			
0	1	9	8			
0	1	10	11			

Fig. 1. Illustration of the array representation of a multicast tree.

[1] Since the end-to-end delay (Parsa et al., 1998) is a pretty important QoS metric to guarantee the real-time data delivery, it is required that the routing path should satisfy the delay constraint.

randomly select a node v_2 from $N(v_1)$. This process is repeated until r_j is reached. Thus, we get a random path $P_T(s, r_j) = \{s, v_1, v_2, \ldots, r_j\}$. Since no loop is allowed on the multicast tree, the nodes that are already included in the current tree are excluded, thereby avoiding reentry of the same node. In this way, the initial population $Q = \{Ch_0, Ch_1, \ldots, Ch_{q-1}\}$ is obtained. The pseudo-code is shown in Algorithm 1.

Algorithm 1. Population initialization.

```
1:  i =: 0;
2:  while i < q do
3:    // Generate chromosome Ch_i
4:    j =: 0;
5:    V_T := E_T := ∅;
6:    while j < |R| do
7:      Search a random path P_T(s, r_j) which can guarantee
        T ∪ P_T be an acyclic graph;
8:      Add all the nodes and links in P_T into V_T and E_T,
        respectively;
9:      j++;
10:   end while
11:   i++;
12: end while
```

4.3. Fitness function

Given a solution, its quality should be accurately evaluated by the fitness value, which is determined by the fitness function. In our algorithms, we aim to find the least cost multicast tree from the source to a set of receivers. The criterion used to evaluate the solution quality is the tree cost. Therefore, among a set of candidate solutions (i.e., multicast trees), we choose the one with the minimal tree cost. The fitness value of chromosome Ch_i (representing the tree T), denoted as $f(Ch_i)$, is given by

$$f(Ch_i) = \left[\sum_{l \in T(V_T, E_T)} c_l \right]^{-1}. \tag{3}$$

The proposed fitness function is to be maximized and only involves the total tree cost. As mentioned above, the delay constraint is checked for each chromosome during the evolutionary process.

4.4. Selection scheme

Selection plays an important role in improving the average quality of the population by passing the high quality chromosomes to the next generation. The selection of chromosomes is based on the fitness value. We adopt the scheme of pair-wise tournament selection without replacement (Lee et al., 2008) as it is simple and effective.

4.5. Crossover and mutation

The performance of a GA relies heavily on two basic genetic operators, i.e., crossover and mutation. Crossover exchanges part of the current solutions in order to find better ones. Mutation helps a GA keep away from local optima. The type and implementation of operators depends on the encoding and also on a problem.

In our algorithm, since a chromosome is expressed by a tree data structure, we adopt a single point crossover to exchange partial chromosomes (sub-trees) at positionally independent crossing sites between two chromosomes (Ahn and Ramakrishna, 2002). With a crossover probability, each time we select two chromosomes Ch_i and Ch_j for crossover. To at least one receiver, Ch_i and Ch_j should possess at least one common node from which one, denoted as v, is randomly selected. In Ch_i, there is a path consisting of two parts: $(s \xrightarrow{Ch_i} v)$ and $(v \xrightarrow{Ch_i} r_k)$. In Ch_j, there is a path consisting of two parts: $(s \xrightarrow{Ch_j} v)$ and $(v \xrightarrow{Ch_j} r_k)$. The crossover operation exchanges the paths $(v \xrightarrow{Ch_i} r_k)$ and $(v \xrightarrow{Ch_j} r_k)$. Fig. 2 illustrates a crossover operation, where node 13 is the selected receiver and node 11 is the selected common node. The partial paths (11→12→13) and (11→8→13) are swapped.

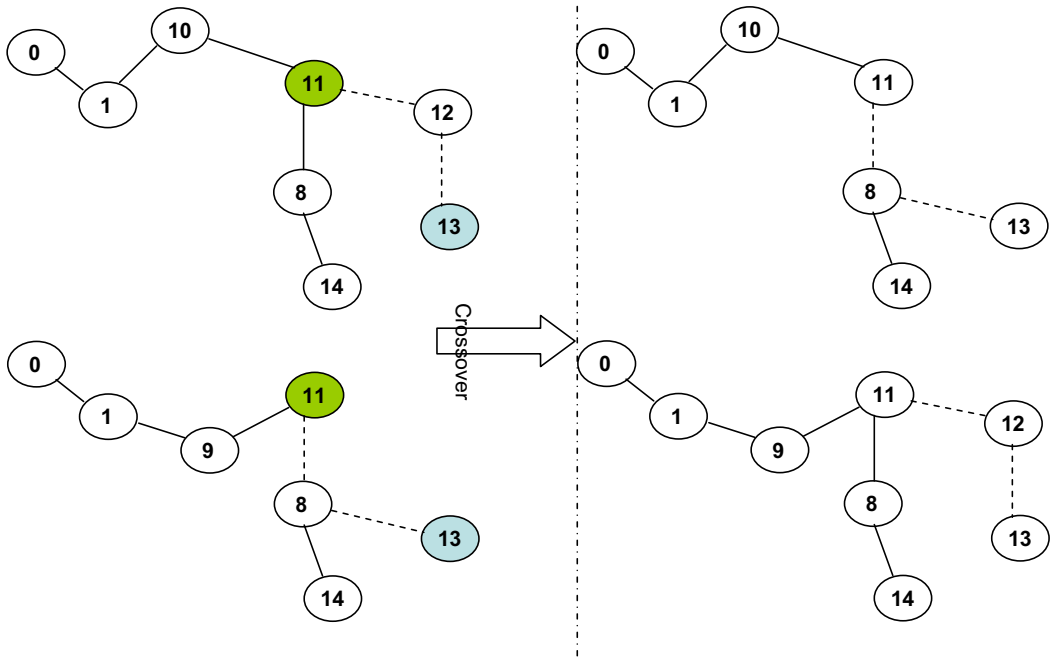


Fig. 2. Illustration of a crossover operation.

The population will undergo the mutation operation after the crossover operation is performed. With a mutation probability, each time we select one chromosome Ch_i on which one receiver r_k is randomly selected. On the path $(s \xrightarrow{Ch_i} r_k)$ one gene is selected as the mutation point (i.e., mutation node) denoted as v. The mutation will replace the path $(v \xrightarrow{Ch_i} r_k)$ by a new random path.

Both crossover and mutation may produce new chromosomes which represent infeasible solutions. Therefore, we check whether the multicast trees represented by the new chromosomes are acyclic. If not, the repair function used in Oh et al. (2006) will be applied to eliminate the loops. The delay checking is incorporated into both the crossover and mutation operations to guarantee that all the new chromosomes produced satisfy the delay constraint.

5. Investigated GAs for the dynamic multicast problem

In this paper, we investigate both traditional GAs and immigrants based GAs for the dynamic QoS multicast problem. As mentioned in Section 3, we propose two models to describe the practical network dynamics. Under the general dynamics model, GAs with immigrants schemes are applied. However, under the worst dynamics model, improved GAs with immigrants schemes are proposed to help conquer the extra difficulties arisen from the topology changes.

5.1. Traditional GAs

The dynamic QoS multicast problem can still be addressed using the specialized GA described above with two variants, denoted *Standard GA* (SGA) and *Restart GA*. In the SGA, when an environmental change leads to infeasible solutions, SGA handles them by taking the measure of penalty. That is, infeasible solutions are set to a very low fitness. In this way, the population in SGA can keep evolving even in a continuously changing environment. In the Restart GA, once a change is detected, the population will be re-initialized based on the new network topology.

5.2. GAs with immigrants schemes

In stationary environments, convergence at a proper pace is usually what we expect for GAs to locate the optimum solutions for many optimization problems. However, for DOPs, convergence usually becomes a big problem for GAs because changing environments usually require GAs to keep a certain population diversity level to maintain their adaptability. To address this problem, the random immigrants approach is a quite natural and simple way (Grefenstette, 1992; Tinos and Yang, 2007; Yang and Tinos, 2007; Yu et al., 2009, 2008). It was proposed by Grefenstette with the inspiration from the flux of immigrants that wander in and out of a population between two generations in nature. It maintains the diversity level of the population through replacing some individuals of the current population with random individuals, called *random immigrants*, every generation. As to which individuals in the population should be replaced, usually there are two strategies: replacing random individuals or replacing the worst ones (Vavak and Fogarty, 1996). In this paper, the random immigrants based GA, denoted *RIGA*, uses the second replacement strategy, i.e., random immigrants replace the worst individuals of the current population. In order to avoid that random immigrants disrupt the ongoing search progress too much, especially during the period when the environment does not change, the ratio of the number of random immigrants to the population size is usually set to a small value, e.g., 0.2.

However, in a slowly changing environment, the introduced random immigrants may divert the searching force of the GA during each environment before a change occurs and hence may degrade the performance. On the other hand, if the environment only changes slightly in terms of the severity of changes, random immigrants may not have any actual effect even when a change occurs because individuals in the previous environment may still be quite fit in the new environment. Based on the above consideration, an immigrants approach, called elitism-based immigrants, was proposed for GAs to address DOPs (Yang, 2007). The elitism-based immigrants GA (EIGA) is also investigated for the dynamic QoS multicast problem in this paper. The pseudo-code for the EIGA is given in Fig. 3.

Within the EIGA, for each generation t, after the normal genetic operations (i.e., selection and recombination), the elite $E(t-1)$ from the previous generation is used as the base to create immigrants. From $E(t-1)$, a set of $r_{ei} * n$ individuals are iteratively generated by mutating $E(t-1)$ with a probability p_m^i, where n is the population size and r_{ei} is the ratio of the number of elitism-based immigrants to the population size. The generated individuals then act as immigrants and replace the worst individuals in the current population. It can be seen that the elitism-based immigrants scheme combines the idea of elitism with traditional random immigrants scheme. It uses the elite from the previous population to guide the immigrants toward the current environment, which is expected to improve the performance of GAs in dynamic environments.

In order to address significant changes that the dynamic QoS multicast problem may suffer, the elitism-based immigrants can be hybridized with traditional random immigrants scheme. The GA with the hybrid immigrants scheme, denoted *HIGA*, is also investigated in this paper. Within HIGA, in addition to the $r_{ei} * n$ immigrants created from the elite of the previous generation, $r_{ri} * n$ immigrants are also randomly created, where r_{ri} is the ratio of the number of random immigrants to the population size. These two sets of immigrants will then replace the worst individuals in the current population. The pseudo-code for HIGA is also shown in Fig. 3.

In our implementation of elitism-based immigrants in both EIGA and HIGA, if the mutation probability p_m^i is satisfied, the elite $E(t-1)$ will be used to generate new immigrants by a mutation operation; otherwise, $E(t-1)$ itself will be directly used as a new immigrant.

```
t := 0
initialize population P(0) randomly
repeat
  evaluate population P(t)
  P'(t) := selectForReproduction(P(t))
  crossover(P'(t), p_c) // p_c is the crossover probability
  mutate(P'(t), p_m)  // p_m is the mutation probability
  evaluate the interim population P'(t)

  // perform elitism-based immigration
  denote the elite in P(t-1) by E(t-1)
  generate r_ei*n immigrants by mutating E(t-1) with p_m^i
  evaluate these elitism-based immigrants

  if the hybrid scheme is used then
    generate r_ri*n random immigrants
    evaluate these random immigrants

  replace the worst individuals in P'(t) with the immigrants
  P(t+1) := P'(t)
until the termination condition is met // e.g., t > t_max
```

Fig. 3. Pseudocode for the elitism-based immigrants GA (EIGA) and the hybrid immigrants GA (HIGA), where the elitism of size one is used.

5.3. Improved GAs with immigrants schemes

In the proposed worst model of network dynamics, every change is caused by removing a few links from the present optimal multicast tree. Therefore, when the environment changes, the present population undergoes dramatic changes and some individuals become infeasible. The general immigrants based GAs do not consider this case and thereby cannot perform well under the worst dynamics model. We propose improved RIGA, EIGA, and HIGA, denoted as *iRIGA*, *iEIGA*, and *iHIGA*, respectively, to address these difficulties.

When there is no environmental changes detected, the above three improved immigrants based GAs just follow the procedures of their corresponding original GAs, respectively. When a change occurs, in iRIGA, all the infeasible individuals are replaced by random immigrants. In iEIGA, all the infeasible individuals are repaired to become feasible and then the elitism is re-selected. In iHIGA, when the environmental change occurs, for each infeasible solution, we either replace it by a random immigrant or repair it with an equal probability of 0.5.

In iEIGA, since the infeasible solutions are previous elitisms, it is required to keep as many feasible components in them as possible. Therefore, the repair should result in the least change to the tree structure. The proposed repair method works as follows. For each removed link, we search a random path starting from its downstream node. Once an existing tree node is encountered, the search ends. This random path is added to the tree to solve the unconnected problem caused by that removed link. After all the removed links are dealt with, the tree becomes feasible again. Intuitively, this simple method can repair an infeasible tree with the least cost added.

6. Performance evaluation

In the simulation experiments, we implement the two traditional GAs (i.e., SGA and Restart GA) and the six immigrants based GAs (i.e., RIGA, EIGA, HIGA, iRIGA, iEIGA, and iHIGA) for the dynamic QoS multicast problem. In SGA, if the change makes one individual in the current population infeasible (e.g., one or more links in the corresponding path are lost), a penalty value is added to its tree cost. By simulation experiments, their performance is evaluated in a continuously changing wireless network. The simulation software for both the dynamic topology generation and the various GAs are developed using C++. The experiments were run on a 160 CPU AMD Opteron cluster system, which is provided by the University of Leicester's Centre for Mathematical Modelling as the shared high-performance computing resources.

6.1. Dynamic test environments

6.1.1. Common issues

Since we consider two dynamics models (i.e., the general one and the worst one), two dynamic test environments are set up accordingly. The initial network topology is generated using the following method. We first specify a square region with the area of 200×200 that has the width [0, 200] on the x axis and the height [0, 200] on the y axis. Then, we generate 100 nodes and the position (x, y) of each node is randomly specified within the square area. If the distance between two nodes falls into the radio transmission range D, a link will be added to connect them and both the cost and the delay of this link are randomly assigned within the corresponding ranges. Finally, we check if the generated topology is connected. If not, the above process is repeated until a connected topology is generated. In the experiments, D is given a reasonable value 50. Each topology is represented by three arrays. One array uses 1 or 0 to represent whether two links are connected or not. The other two arrays give the corresponding cost and delay values of each link, respectively.

In both models, all the algorithms start from the initial network topology. Every certain number (say, I) of generations (i.e., the change interval), the network topology is changed in a way corresponding to the dynamics model used. It can be seen that I determines the change frequency. The larger the value of I, the slower the problem changes. In the following experiments, we set I to 5, 10 and 15 separately to see the impact of the change frequency on the performance of dynamic GAs. In all the experiments, the crossover probability was set to 0.95 and the mutation probability was set to 0.05. For RIGA, iRIGA, EIGA, and iEIGA, the ratios of the number of immigrants to the population size, r_{ri} and r_{ei}, were set to 0.2. However, in HIGA and iHIGA, to guarantee the comparison fairness, that is, the same number of immigrants are introduced every generation, r_{ri} and r_{ei} were set to 0.1. In EIGA, iEIGA, HIGA, and iHIGA, the mutation probability p_m^i for generating new immigrants, was set to 0.8. Both the source and destination nodes were randomly selected. The delay upper bound Δ was set to be 2 times of the minimum end-to-end delay.

In order to have fair comparisons among GAs, the population size and immigrants ratios were set such that each GA has 60 fitness evaluations per generation as follows:

$$(1 + r_i) * n = 60, \tag{4}$$

where n is the whole population size, which was set to 50 in Section 6.2.1. Hence, we have n=60 for SGA and Restart GA, and n=50 for RIGA, EIGA, HIGA, iRIGA, iEIGA, and iHIGA. At each generation, for each algorithm, we select the best individual from the current population and output the cost of the optimal tree represented by it. For each experiment of an algorithm on a dynamic problem, 10 independent runs were executed with the same set of random seeds. For each run, 20 environmental changes were allowed under both dynamics models, which are equivalent to 105, 210, and 315 generations, including the initial environment, for I=5, 10, and 15, respectively. For each run, the best-of-generation fitness was recorded every generation. The overall offline performance of a GA on a DOP is defined as

$$\overline{F}_{BOG} = \frac{1}{G}\sum_{i=1}^{G}\left(\frac{1}{N}\sum_{j=1}^{N} F_{BOG_{ij}}\right), \tag{5}$$

where G is the total number of generations for a run, N=10 is the total number of runs, and $F_{BOG_{ij}}$ is the best-of-generation fitness of generation i of run j. $\overline{F}_{BOG}$ is the off-line performance, i.e., the best-of-generation fitness averaged over the 10 runs and then over the data gathering period.

6.1.2. Environmental changes under the general dynamics model

In the general dynamics model, every I generations, a certain number (say, M) of nodes are scheduled to sleep or wake up depending on their current status. It means that the selected working nodes will be turned off to sleep and the selected sleeping nodes will be turned on to work. Therefore, the network topology is changed accordingly since some links are lost and some other links appear again. The nodes are randomly selected and thereby the affected links may belong to the present multicast tree or not. The source and destination nodes are not allowed to be scheduled in any change. By this means, we create a series of network topologies corresponding to the continuous environmental changes. Furthermore, these adjacent topologies are highly related since each time the change affects only part of the nodes.

It can be seen that M determines the change severity. The larger the value of M, the more severe the changes. We set M to 2 and 4, respectively. Thus, by the number of nodes changed per time, we have two different series of topologies. When M is set to 2 and 4, we generate the topology series #2 and #4, respectively. Each of these two series has 21 different topologies. All the experiments under the general model are based on these two topology series. We set up experiments to evaluate the population size and the improvements over traditional GAs using RIGA, EIGA and HIGA.

6.1.3. Environment changes under the worst dynamics model

In the worst dynamics model, every I generations, the present best multicast tree is first identified. Then, a certain number (say, U) of links on the tree are selected for removal. It means that the selected links will be forced to be removed from the network topology. Just before the next change occurs, the network topology is recovered to its original state and ready for the coming change. The population is severely affected by each topology change since the optimal solution and possibly some other good solutions become infeasible suddenly. To be fair, at most one link is allowed to be removed on the tree path from the source to each receiver. We let U range from 1 to 3 to see the effect of the change severity.

Under the worst dynamics model, the topology series cannot be generated in advance because every change is correlated with the algorithm running. However, similarly, we also allow 20 changes. We set up the experiments to evaluate the impact of the change interval and the change severity, and the improvements over traditional GAs using iRIGA, iEIGA and iHIGA.

6.2. Experimental results and analysis

6.2.1. Under the general dynamics model

First, we investigate the effect of the population size on the performance of GAs and determine a proper population size that ensures a specified quality of solutions. We pick up HIGA as an example and run it over topology series #2. Here, I was set to 10. The population size was varied from 30 to 60 to see if an appropriate population size can be determined for this problem. Since there are 21 topologies in each series, HIGA evolves $21 * I$ generations in total. We sample the first 200 generations to plot the figures. Figs. 4(a) and (b) show the results over topology series #2.

Figs. 4(a) and (b) show that on the average HIGA achieves the best performance at the population size of 50. When the population size is increased to 60, the algorithm performance degrades on most of the time. Other algorithms are checked and similar results are found. Therefore, we conclude that 50 is the best choice for the population size for our problem. From Figs. 4(a) and (b), it can also be seen that many times when a change occurs, the algorithms are not affected. The reason lies in that the topology changes may not always affect the current population, especially the optimal individual in the population. For example, if the nodes that are scheduled to sleep or wake up in one change are not on the tree represented by the optimal

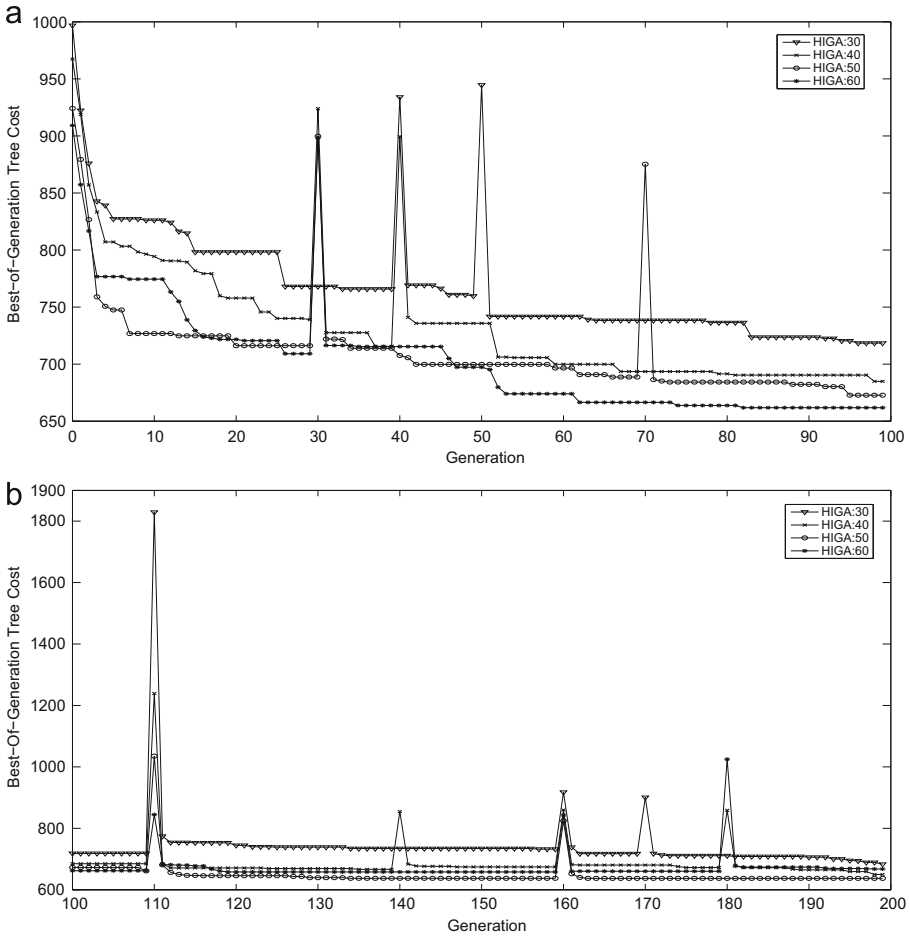


Fig. 4. Comparison results of the quality of solution for HIGA with different population sizes over topology series #2 from: (a) generation 0 to 99 and (b) generation 100 to 199.

individual, the optimal individual has a very high probability to stay in the population unaffected. This explains why the algorithms rarely react to the change drastically. However, under the worst dynamics model, we will see totally different results.

Since under the general dynamics model, the environment changes do not have significant effects on the GAs, the investigation on both the change interval and the change severity are put under the worst dynamics model. However, under this model, we are still interested in the comparison between the dynamic GAs with the traditional GAs over the dynamic multicast problem. Since the dynamic GAs are designed for the dynamic environments, they should show a better performance than the traditional GAs over our problem. We compared RIGA, EIGA, and HIGA with SGA and Restart GA in the experiments using topology series #2 and #4 as the two dynamic environments. The interval of changes was set to 10 here.

Figs. 5(a) and (b) show the comparison results over topology series #2 and #4, respectively. From Figs. 5(a) and (b), it can be seen that SGA always exhibits the worst performance. When the topology is changed and infeasible solutions occupy the population, SGA cannot recover the population by generating new feasible solutions through the standard evolutionary operations. Therefore, simple penalty cannot make the population adapt to the complicated environmental changes. On average, the Restart GA is also worse than any of the three immigrants based GAs. The reason is that the Restart GA does not exploit any useful information in the old environment and that the frequent restart sacrifices its evolving capability. Immigrants bring more diversity to the populations in RIGA, EIGA and HIGA and therefore enhance their search capabilities. Among the three dynamic GAs, EIGA achieves the worst performance in most of the time. The reason lies in that in EIGA, new immigrants are generated from the mutation of the elitism. In our problem, the mutation operation just changes a partial path on the tree. Thus, the new immigrants share most of the tree components and bring less diversity into the population than RIGA and HIGA.

The corresponding statistical results of comparing these GAs under the general dynamics model by a one-tailed t-test with 18 degrees of freedom at a 0.05 level of significance are given in Table 1. In Table 1, the t-test result regarding *Alg.* 1 − Alg. 2 is shown as "+", "−", "s+", or "s−" when *Alg.* 1 is insignificantly better than, insignificantly worse than, significantly better than, or significantly worse than *Alg.* 2, respectively.

Table 1
The t-test results of comparing GAs on the general dynamics model with I=10.

t-Test result	Topology series #2	Topology series #4
RIGA–SGA	s+	s+
EIGA–SGA	s+	s+
HIGA–SGA	s+	s+
RIGA–Restart GA	s+	s+
EIGA–Restart GA	s+	s+
HIGA–Restart GA	s+	s+
RIGA–HIGA	+	+
RIGA–EIGA	s+	+
EIGA–HIGA	s−	−

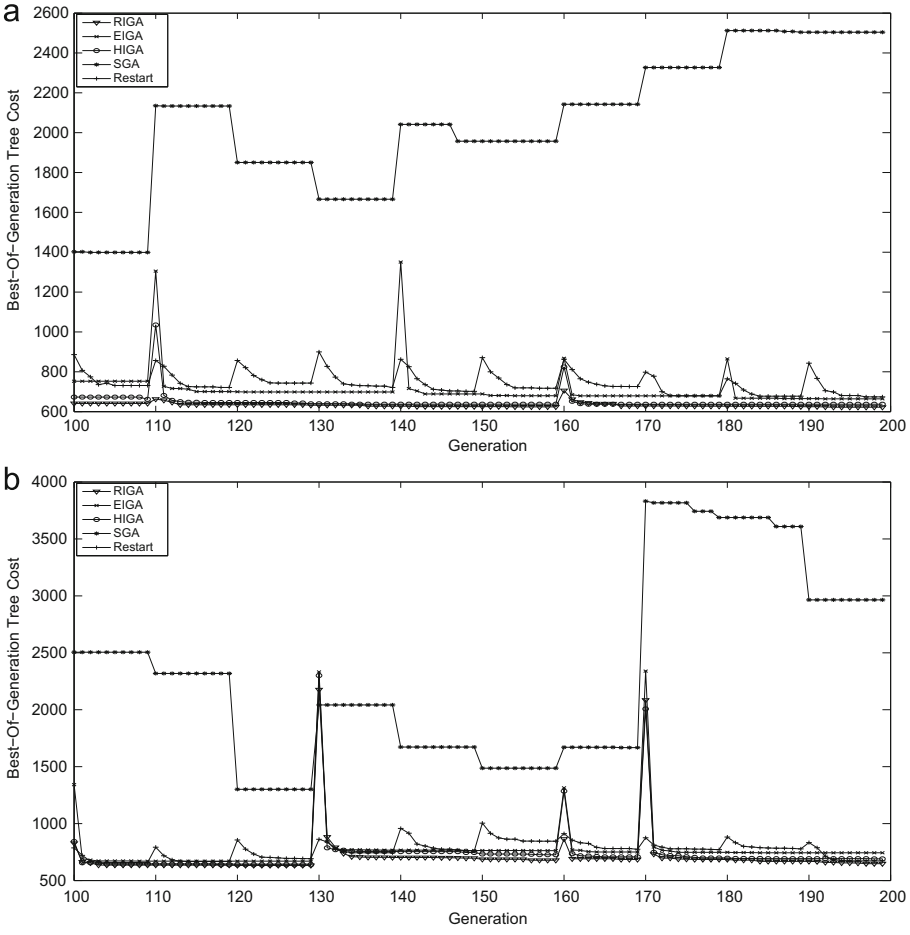


Fig. 5. Comparison results of the quality of solution for RIGA, EIGA, HIGA, SGA, and Restart GA over: (a) topology series #2 and (b) topology series #4.

In addition to the overall best-of-generation solution quality, we also want to compare different GAs in a statistical way. Since each algorithm is repeated 10 times under the same experimental setting and dynamic environment, each algorithm has 10 independent evolution procedures. For each run, we calculate the mean value of the best-of-generation solutions over the whole evolution procedure. Therefore, we get 10 mean values for each algorithm in the same experiment. Table 2 shows the comparison results in terms of the mean value and the corresponding variance of the best solutions at all the generations of each run over topology series #2. The change interval was set to 10. Since the best-of-generation value at each change point may be drastically different from the values at other generations, we exclude them from the calculation. The results in Table 2 approximately match the results presented in Fig. 5(a).

All the above results reveal the algorithm performance by the best-of-generation solution quality. However, not only the statistical values of the best-of-generation are interesting, but also the statistics of the total population can give some insights into the operation of algorithms. For all the GAs, we compared the results about the mean-of-generation solution quality over a typical run. Fig. 6 shows the comparison results for RIGA, EIGA, HIGA, SGA, and Restart GA over topology series #2. We also calculated the variance for the whole population at each generation. However, due to the extremely large value interval where these variance values fall into, the figures cannot be plotted. We list the variance values of 10 generations in Table 3. However, these results are just based on one run, they can only be used to observe the whole population performance instead of evaluating the algorithms.

6.2.2. Under the worst dynamics model

First, we investigate the impact of the change interval on the performance of algorithms. Here, the number of links removed per change was set to 2. When the change interval is 5, the population evolves only five generations between two sequential changes. Intuitively, a larger interval will give the population more time to evolve and search better solutions than what a smaller interval does. We take both iRIGA and iHIGA as examples to compare the quality of solutions obtained under different change intervals. However, one problem is that the total generations are different for different change intervals. There

Table 2
The results of comparing GAs in terms of the mean value and variance of the best solutions at each run under the general dynamics model.

Run #	1st	2nd	3rd	4th
RIGA	854.958 ± 14653.9	842.668 ± 26164.6	853.211 ± 14268.9	915.037 ± 19876.1
EIGA	803.089 ± 5061.28	916.079 ± 17756.7	913.289 ± 20697.2	1008.25 ± 18001.8
HIGA	874.226 ± 18733.9	974.621 ± 17046.4	858.337 ± 13116.1	934.453 ± 19548.7
SGA	1374.81 ± 704844	1521.08 ± 1.1076e+06	1287.83 ± 731091	1595.97 ± 1.9045e+06
Restart GA	974.353 ± 17858.6	1008.47 ± 19107.9	976.653 ± 16014.1	1023.45 ± 19454.1
Run #	5th	6th	7th	8th
RIGA	774.768 ± 32792.1	883.232 ± 22736.9	833.8 ± 21138	741.105 ± 12887.6
EIGA	867.463 ± 16180.3	829.632 ± 10744.3	932.253 ± 4126.35	958.205 ± 28485.4
HIGA	880.847 ± 14668.6	910.458 ± 11298.4	935.242 ± 8871.88	962.542 ± 13073.9
SGA	1856.53 ± 751736	1978.07 ± 1.3042e+06	2583.98 ± 1.7888e+06	1854.11 ± 954893
Restart GA	979.258 ± 16898.5	1002.62 ± 23240.9	976.116 ± 15677.8	1036.11 ± 16717.1
Run #	9th	10th		
RIGA	933.263 ± 15426.2	923.663 ± 35886.6		
EIGA	946.226 ± 11460.3	910.416 ± 13384.2		
HIGA	902.321 ± 6254.98	924.3 ± 17311.6		
SGA	2531.79 ± 1.7547e+06	1393.49 ± 1.3180e+06		
Restart GA	976.579 ± 15878.5	1027.37 ± 14813.5		

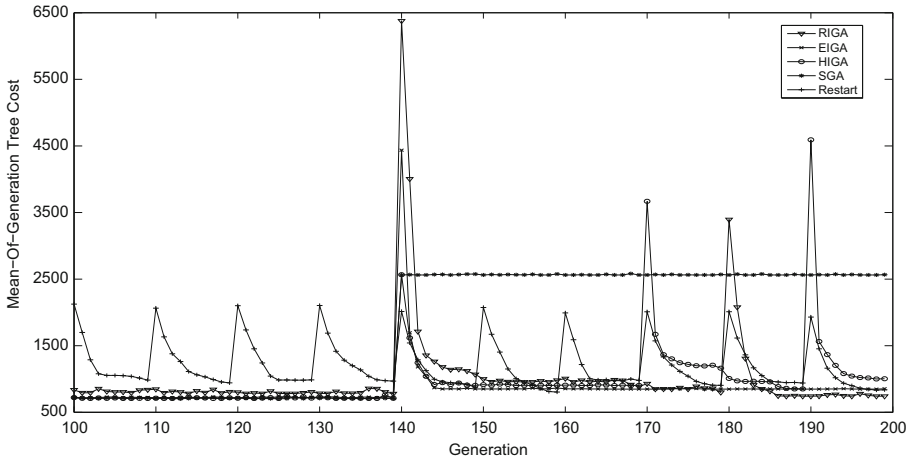


Fig. 6. Comparison results of the mean quality of solution of the whole population at each generation for RIGA, EIGA, HIGA, SGA, and Restart GA over topology series #2.

are 100, 200, and 300 generations corresponding to the change interval 5, 10, and 15, respectively, when there are 20 different topologies. Since the number of change points (i.e., the generation at which a new topology is applied) is the same for all the intervals, we take the data at each change point and its previous two and next two generations. Thus, the three data sets can be aligned over the three intervals.

Table 3
The results of comparing GAs in terms of the variance of the solutions in the whole population at each generation under the general dynamics model.

Generation #	RIGA	EIGA	HIGA	SGA	Restart GA
100th	26400.4	0	4673.6	5123.79	122592
101st	0	0	282.24	1166.53	101541
102nd	0	0	0	2046.66	59445.8
103rd	75433.6	148.352	1789.93	880.902	7869.4
104th	21526.8	3601.16	138.298	3556.93	308.616
105th	6212.59	3156.67	2032.21	3529.47	1132.71
106th	2934.61	0.0784	0	1304.65	350.982
107th	0	2180.35	0	285.446	1295.89
108th	26674.3	527.162	162.308	5771.29	1183.33
109th	42225.1	1009.97	0	2140.63	3922.33
110th	112474	0	325.018	0	134707
111th	3082.47	1397.26	0	2321.4	63631.7
112th	20378.1	0	0	0	26409
113th	18893.3	685.392	1847.28	0	17445
114th	0	566.44	7533.23	331.24	7611.3
115th	38018.4	0	6234.68	2314.45	6357.6
116th	5127.45	0	1546.18	1172.74	4150.54
117th	49666.6	4150.95	0	7100.54	4069.32
118th	2387.3	3073.59	0	521.424	861.626
119th	18028.1	0	2672.45	2428.52	2324.28

Figs. 7(a) and (b) show the results regarding iRIGA and iHIGA, respectively. Since the generation number does not correspond to the actual generation number when the interval is 10 or 15, we rename it as pseudo generation. In Fig. 7(a), over the 20 topologies, the iRIGA with change interval 15 achieves 11 best solutions while the iRIGA with change intervals 10 and 5 achieve 6 and 3, respectively. In Fig. 7(b), over the 20 topologies, the iRIGA with change interval 15 achieves 16 best solutions while the iRIGA with change interval 10 achieves 4. It can be concluded that the solution quality becomes better when the change interval becomes larger. Therefore, in a relatively slowly changing environment, the improved immigrants based GAs can achieve a good performance.

Second, we investigate the impact of the change severity on the performance of algorithms. Under the worst dynamics model, the change severity is reflected by the number of links removed from the present optimal tree per change. Therefore, we generate two topology series by removing different number of links each change. One is to remove only one link each change and the other is to remove three links each change. These two topology series act as the two environments with different change severities. This time, we pick up iRIGA, iEIGA, and iHIGA together as the example algorithms and we set the change interval to 10. Figs. 8(a) and (b) show the results in the two different environments, respectively.

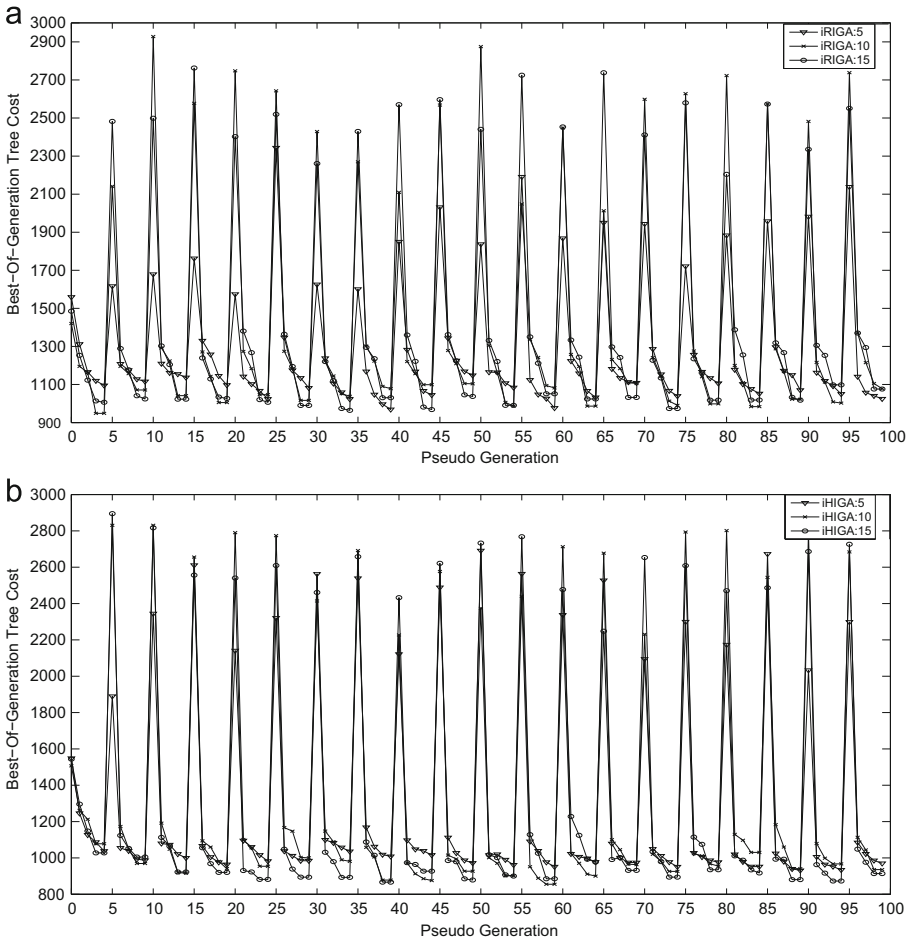


Fig. 7. Comparison results of the quality of solution under different change intervals for: (a) iRIGA and (b) iHIGA.

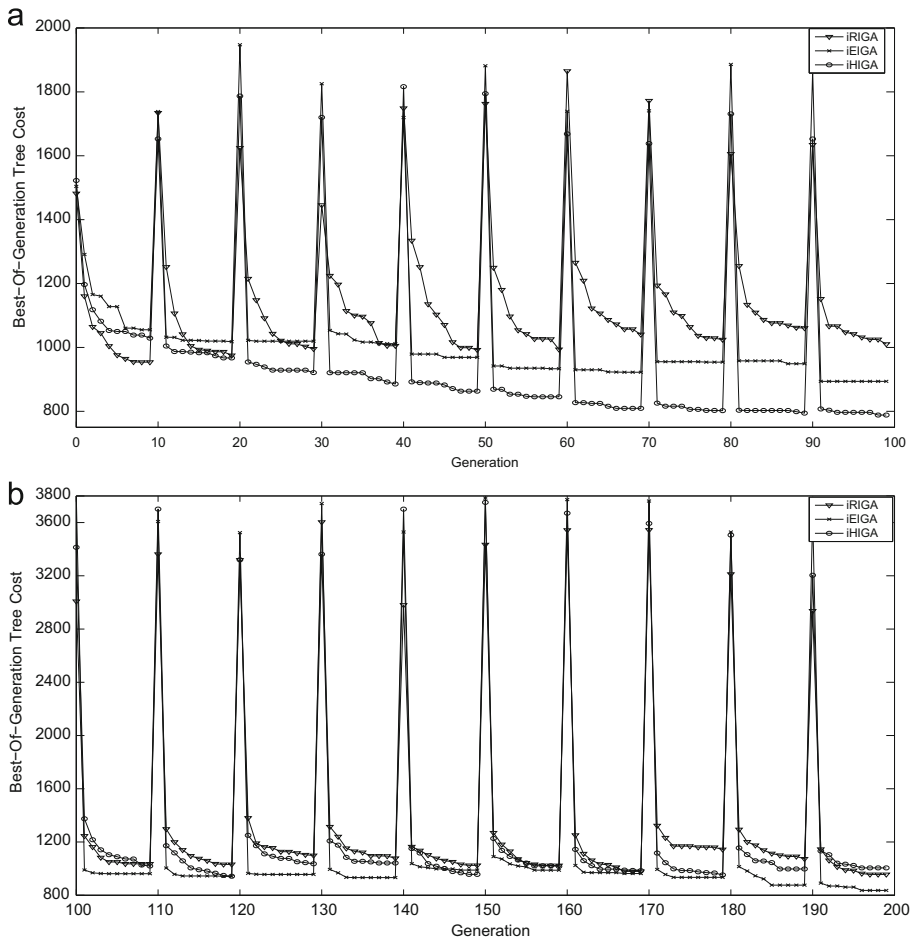


Fig. 8. Comparison results of the response speed to changes for iRIGA, iEIGA, and iHIGA over two different topology series where: (a) each change removes one link and (b) each change removes three links from the present optimal tree.

From Fig. 8(a), in can be seen that iRIGA takes almost nine generations to get its best solution after one change occurs. iEIGA takes about five generations to get its best solution. However, iHIGA can quickly adapt to the environmental changes and get the best solution among these three GAs in 90% of the time. Therefore, in the environment with a low change severity, iHIGA performs the best since it takes the advantages of both iRIGA and iEIGA. From Fig. 8(b), it can be seen that for both iRIGA and iHIGA, they need almost nine generations to get their best solutions after one change occurs and iHIGA achieves a better solution quality than iRIGA. However, in this environment with a high change severity, iEIGA performs very well which takes two to five generations to get the overall best solution. The reason is that in the highly dynamic environment, after a change occurs, the proposed repair method can reserve the useful components of the elitism and repair the broken part with the least added cost. The new immigrants generated by repairing the elitisms can quickly adapt to the severe changes. However, as we have discussed under the general dynamics model, iEIGA brings a less diversity to the population compared to both iRIGA and iHIGA. Therefore, we can conclude that these dynamic GAs respond to the environmental changes in a reasonable speed and perform well.

Third, we compare the dynamic GAs with the traditional GAs under the worst dynamics model. Here, the number of links removed per change is set to 2 and the change interval is 10. Figs. 9(a) and (b) show the results. Similar as under the general model, SGA performs the worst since it does not explicitly handle the environmental changes. Most of the time, Restart GA performs worse than all the improved immigrants based GAs. However, occasionally, iRIGA is worse than it. Overall, iEIGA is better than iHIGA as it has shown in the environment with a high change severity. It can be concluded that these three improved immigrants based GAs greatly outperform those two standard GAs under the worst dynamics model.

The corresponding statistical results of comparing these GAs under the worst dynamics model by a one-tailed t-test with 18 degrees of freedom at a 0.05 level of significance are given in Table 4.

Similarly, as in the experiments under the general dynamics model, we also want to compare different GAs in a statistical way under the worst dynamics model. For each of the 10 runs, we calculated the mean value of the best-of-generation solutions over the whole evolution procedure. Table 5 shows the comparison results in terms of the mean value and the corresponding variance of the best solutions at all the generations of each run. The change interval was set to 10 and the number of links removed per change was set to 2. Similarly, we also excluded the best-of-generation value at each change point from the calculation. The results in Table 5 approximately match the results presented in Fig. 9.

Similarly, we are also interested in the statistical values of the total population under the worst dynamics model since they can give insights on the operation of GAs. For all the improved GAs and traditional GAs, we compare the results regarding the mean-of-generation solution quality over a typical run. Fig. 10 shows the

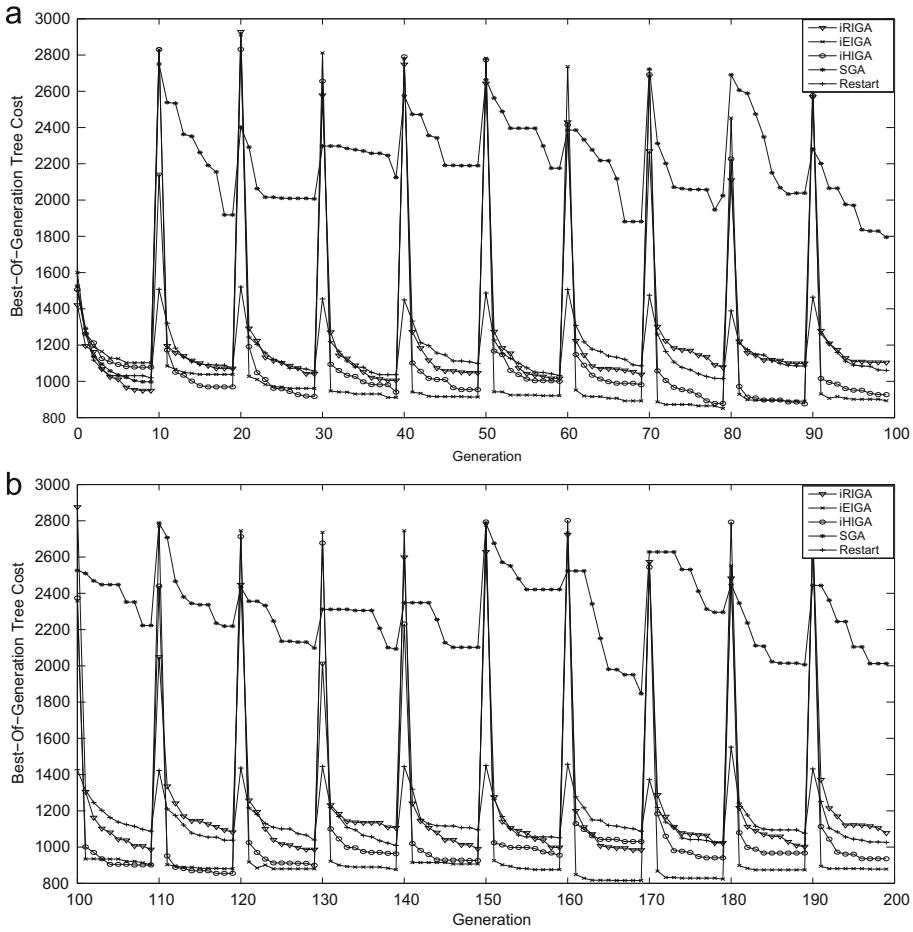


Fig. 9. Comparison results of the quality of solution for iRIGA, iEIGA, iHIGA, SGA, and Restart GA from: (a) generation 1 to 99 and (b) generation 100 to 199.

Table 4
The *t*-test results of comparing GAs under the worst dynamics model with $I=10$ and the number of links removed per change $U=2$.

Compared algorithms	*t*-Test result
iRIGA–SGA	*s*+
iEIGA–SGA	*s*+
iHIGA–SGA	*s*+
iRIGA–Restart GA	+
iEIGA–Restart GA	*s*+
iHIGA–Restart GA	*s*+
iRIGA–iHIGA	*s*−
iRIGA–iEIGA	*s*−
iEIGA–iHIGA	+

comparison results for iRIGA, iEIGA, iHIGA, SGA, and Restart GA. We also calculated the variance for the whole population at each generation of each algorithm. However, also due to the extremely large value interval where these variance values fall into, the figures cannot be plotted. We list the variance values of 20 generations in Table 6. It can be seen that iRIGA and Restart GA bring much higher variance values than other GAs.

7. Conclusions

Mobile ad hoc networks (MANETs) have seen various collaborative multimedia applications which require an efficient information delivery service from a designated source to multiple receivers. An QoS multicast tree is preferred to support this service. However, the optimal QoS multicast routing problem is proved to be NP-hard. Quite some works have been done to address the static multicast problem by genetic algorithms. However, in MANETs, the topology dynamics makes this problem much harder to solve. So far, little work has been done on the dynamic multicast problem in mobile networks. By observing that immigrants based GAs (i.e., RIGA, EIGA, HIGA) perform very well over many dynamic benchmark problems, we apply them to the dynamic QoS multicast problem in MANETs in this paper. Based on the problem characteristics, we also propose three improved versions of immigrants based GAs, i.e., iRIGA, iEIGA, and iHIGA, to handle highly dynamic environments.

Extensive simulation experiments have been conducted to compare the proposed algorithms with traditional GAs. We highlight the investigation on the best-of-generation solution quality since it represents the performance of algorithms. Meanwhile, we also investigate the mean value and variance of the best solutions over all the generations of each run. Furthermore, we also look inside the population by investigating the mean value and variance of all the individuals in the same population at each generation. All these experiments help reveal the performance of GAs in various aspects. Experimental results demonstrate that our algorithms can adapt to the environmental changes well and achieve better solutions after each change than the traditional GAs. Therefore, they are promising techniques for dealing with dynamic telecommunication problems.

Table 5
The results of comparing GAs in terms of the mean value and variance of the best solutions at each run under the worst dynamics model.

Run #	1st	2nd	3rd	4th
iRIGA	310.295 ±3025.99	308.411 ±3472.15	291.184 ±3137.23	291.632 ±3003.89
iEIGA	243.216 ±1694.03	303.253 ±2100.26	237.837 ±2496.82	248.342 ±1132.37
iHIGA	290.584 ±3088.98	296.7 ±3588.48	270.111 ±2014.81	260.384 ±2137.82
SGA	936.511 ±122170	980.895 ±65683.4	925.611 ±80954.8	1100.78 ±45708.3
Restart GA	301.342 ±2448.95	305.247 ±4651.78	292.474 ±3076.83	296.047 ±2772.08
Run #	5th	6th	7th	8th
iRIGA	292.037 ±3429.24	285.032 ±4582.25	314.468 ±3602.13	303.837 ±3255.5
iEIGA	244.037 ±1378.82	251.316 ±1692.03	249.974 ±1731.54	243.847 ±2360.47
iHIGA	293.511 ±2290.93	262.284 ±1670.74	278.811 ±1447.87	280.016 ±2174.41
SGA	1027.09 ±80585.3	1026.32 ±80714.9	1036.07 ±66072.6	955.2 ±82530.5
Restart GA	281.595 ±6145.27	310.047 ±2850.66	297.979 ±5658.09	297.342 ±4878.44
Run #	9th	10th		
iRIGA	302.742 ±2607.67	309.289 ±3298.96		
iEIGA	207.958 ±1285.54	245.753 ±1138.94		
iHIGA	270.153 ±3478.76	299.642 ±1715.35		
SGA	1015.56 ±63934.3	975.963 ±95070.8		
Restart GA	297.484 ±4395.76	317.926 ±3983.98		

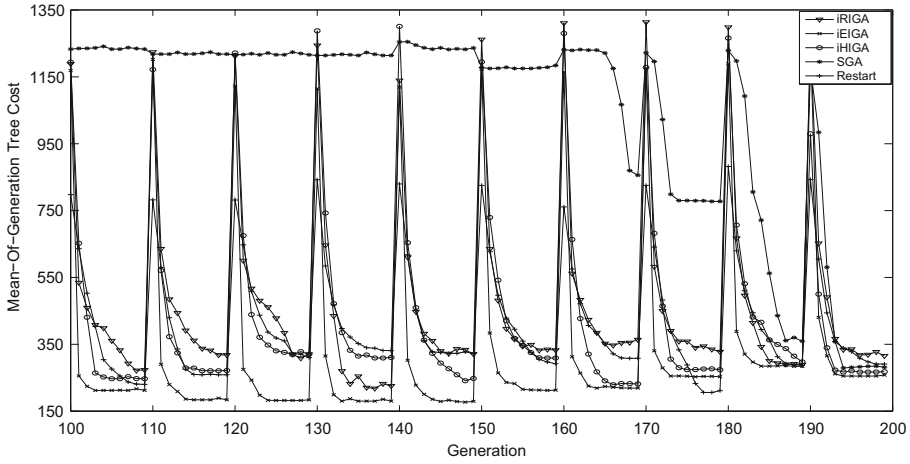


Fig. 10. Comparison results of the mean quality of solution of the whole population at each generation for iRIGA, iEIGA, iHIGA, SGA, and Restart GA.

Table 6
The results of comparing GAs in terms of the variance of the solutions in the whole population at each generation under the worst dynamics model.

Generation #	iRIGA	iEIGA	iHIGA	SGA	Restart GA
100th	2764.29	0	0	0	34727.9
101st	14139.8	976.356	40303.8	184.416	13203.1
102nd	6986.85	208.322	14709.1	228.614	9626.7
103rd	315.496	0	1270.68	663.578	7876.52
104th	2997.16	0	1376.41	1466.82	3326.92
105th	849.658	0	0	0	2744.98
106th	1282.99	24.01	0	0	338.064
107th	1733.2	0	1982.03	511.488	207.36
108th	900.294	1176.49	11.2896	165.894	57.1536
109th	1557.05	0	0	0	2.3716
110th	50.9796	0	2484.03	0	45966.7
111th	16027.6	3590.62	40036.6	0	18138.3
112th	3741.12	354.92	3053.13	0	7569.53
113th	2945.37	423.054	4366.7	1176.49	4022.57
114th	1770.13	104.68	757.364	0	1410.83
115th	1005.16	0	2772.33	6.3504	34.5744
116th	456.574	0	0	296.528	0
117th	2016.34	0	0	843.534	98.8036
118th	0.49	1147.85	0	0	0
119th	0	0	16.4836	0	0

Acknowledgments

The authors are grateful to the anonymous reviewers for their thoughtful suggestions and constructive comments. This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of UK under Grant EP/E060722/1.

References

Adelstein, F., Richard, G., Schwiebert, L., 2003. Distributed multicast tree generation with dynamic group membership. Comput. Commun. 26 (10), 1105–1128.
Aharoni, E., Cohen, R., 1998. Restricted dynamic Steiner trees for scalable multicast in datagram networks. IEEE/ACM Trans. Networking 6 (3), 286–297.
Ahn, C., Ramakrishna, R., 2002. A genetic algorithm for shortest path routing problem and the sizing of populations. IEEE Trans. Evol. Comput. 6 (6), 566–579.
Ahn, C., Ramakrishna, R., Kang, C., Choi, I., 2001. Shortest path routing algorithm using Hopfield neural network. Electron. Lett. 37 (19), 1176–1178.
Branke, J., 2002. Evolutionary Optimization in Dynamic Environments. Kluwer Academic Publishers, Dordrecht.
Cordeiro, C., Gossain, H., Agrawal, D., 2003. Multicast over wireless mobile ad hoc networks: present and future directions. IEEE Network 17 (1), 52–59.
Din, D., 2005. Anycast routing and wavelength assignment problem on WDM network. IEICE Trans. Commun. E88-B (10), 3941–3951.

Gen, M., Cheng, R., 2000. Genetic Algorithms and Engineering Optimization. Wiley, New York.
Goldberg, D., 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA, USA.
Grefenstette, J., 1992. Genetic algorithms for changing environments. In: Proceedings of the Second International Conference on Parallel Problem Solving from Nature, pp. 137–144.
Helvig, C., Robins, G., Zelikovsky, A., 2000. An improved approximation scheme for the group Steiner problem. Networks 37 (1), 8–20.
Holland, J., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor.
Hwang, F., Richards, D., 1992. Steiner tree problems. Networks 22 (1), 55–89.
Jia, X., 1998. A distributed algorithm of delay-bounded multicast routing for multimedia applications in wide area networks. IEEE/ACM Trans. Networking 6 (6), 828–837.
Jia, X., Pissinou, N., Makki, K., 1997. A real-time multicast routing algorithm for multimedia applications. Comput. Commun. 20 (12), 1098–1106.
Khuller, S., Raghavachari, B., Young, N., 1995. Balancing minimum spanning and shortest path trees. Algorithmica 14 (4), 305–321.
Kuipers, F., Mieghem, P.V., 2002. MAMCRA: a constrained-based multicast routing algorithm. Comput. Commun. 25 (8), 802–811.
Law, L., Krishnamurthy, S.V., Faloutsos, M., 2005. Fireworks: an adaptive group communications protocol for mobile ad hoc networks. In: Proceedings of the Fourth IFIP Networking, pp. 853–868.
Lee, S., Soak, S., Kim, K., Park, H., Jeon, M., 2008. Statistical properties analysis of real world tournament selection in genetic algorithms. Appl. Intell. 28 (2), 195–2205.
Morrison, R.W., 2004. Designing Evolutionary Algorithms for Dynamic Environments. Springer, Berlin.
Narvaez, P., Siu, K.-Y., Tzeng, H.-Y., 2000. New dynamic algorithms for shortest path tree computation. IEEE/ACM Trans. Networking 8 (6), 734–746.
Oh, S., Ahn, C., Ramakrishna, R., 2006. A genetic-inspired multicast routing optimization algorithm with bandwidth and end-to-end delay constraints. In: Proceedings of the 13th International Conference on Neural Information Processing (ICONIP 2006), October, Lecture Notes in Computer Science, vol. 4234. Springer, Berlin, pp. 807–816.
Oliveira, C., Pardalos, P., 2005. A survey of combinatorial optimization problems in multicast routing. Comput. Oper. Res. 32 (8), 1953–1981.
Papadimitriou, C., Steiglitz, K., 1998. Combinatorial Optimization: Algorithms and Complexity. Dover Publications Inc, New York.
Parsa, M., Zhu, Q., Garcia-Luna-Aceves, J., 1998. An iterative algorithm for delay-constrained minimum-cost multicasting. IEEE/ACM Trans. Networking 6 (4), 461–474.
Perkins, C.E., 2001. Ad Hoc Networking. Addison-Wesley, London.
Robins, G., Zelikovsky, A., 2000. Improved Steiner tree approximation in graphs. In: Proceedings of the ACM/SIAM Symposium on Discrete Algorithms.
Siva Ram Murthy, C., Manoj, B.S., 2004. Ad Hoc Wireless Networks: Architectures and Protocols. Prentice-Hall PTR, Englewood Cliffs, NJ.
Tinos, R., Yang, S., 2007. A self-organizing random immigrants genetic algorithm for dynamic optimization problems. Genet. Program. Evol. Mach. 8 (3), 255–286.
Toh, C.-K., 2002. Ad Hoc Mobile Wireless Networks: Protocols and Systems. Prentice-Hall PTR, Englewood Cliffs, NJ.
Tyan, H.-Y., Hou, J.C., Wang, B., 2003. Many-to-many multicast routing with temporal quality of service guarantees. IEEE Trans. Comput. 52 (6), 826–832.
Vavak, F., Fogarty, T., 1996. A comparative study of steady state and generational genetic algorithms for use in nonstationary environments. In: Proceedings of the AISB Workshop on Evolutionary Computing, pp. 297–304.
Wang, B., Hou, J., 2000. A survey on multicast routing and its QoS extensions: problems, algorithms, and protocols. IEEE Network 14 (1), 22–36.
Wang, X., Cao, J., Cheng, H., Huang, M., 2006. QoS multicast routing for multimedia group communications using intelligent computational methods. Comput. Commun. 29 (12), 2217–2229.
Weicker, K., 2003. Evolutionary Algorithms and Dynamic Optimization Problems. Der andere Verlag, Osnabrück, Germany.
Xiao, X., Ni, L., 1999. Internet QoS: a big picture. IEEE Network 13 (2), 8–18.
Xue, G., 2003. Minimum-cost QoS multicast and unicast routing in communication networks. IEEE Trans. Commun. 51 (5), 817–824.
Yang, S., 2007. Genetic algorithms with elitism-based immigrants for changing optimization problems. In: Evoworkshops 2007: Applications of Evolutionary Computing. Lecture Notes in Computer Science, vol. 4448, pp. 627–636.
Yang, S., Tinos, R., 2007. A hybrid immigrants scheme for genetic algorithms in dynamic environments. Int. J. Automat. Comput. 4 (3), 243–254.
Yang, S., Yao, X., 2005. Experimental study on population-based incremental learning algorithms for dynamic optimization problems. Soft Comput. 9 (11), 815–834.
Yang, S., Yao, X., 2008. Population-based incremental learning with associative memory for dynamic environments. IEEE Trans. Evol. Comput. 12 (5), 542–561.
Yong, K., Poo, G., Cheng, T., 2008. Proactive rearrangement in delay constrained dynamic membership multicast. Comput. Commun. 31 (10), 2566–2580.
Yu, X., Tang, K., Yao, X., 2008. An immigrants scheme based on environmental information for genetic algorithms in changing environments. In: Proceedings of the 2008 Congress on Evolutionary Computing, pp. 1141–1147.
Yu, X., Tang, K., Chen, T., Yao, X., 2009. Empirical analysis of evolutionary algorithms with immigrants schemes for dynamic optimization. Memetic Comput. 1 (1), 3–24.