IMMIGRANTS-ENHANCED MULTI-POPULATION GENETIC ALGORITHMS FOR DYNAMIC SHORTEST PATH ROUTING PROBLEMS IN MOBILE AD HOC NETWORKS

Hui Cheng¹, Shengxiang Yang², and Xingwei Wang³
¹Department of Computer Science and Technology, University of Bedfordshire, Luton, UK
²Department of Information Systems and Computing, Brunel University, Uxbridge, UK
³College of Information Science and Engineering, Northeastern University, Shenyang, China

One of the most important characteristics in mobile wireless networks is the topology dynamics, that is, the network topology changes over time as a result of energy conservation or node mobility. Therefore, the shortest path (SP) routing problem turns out to be a dynamic optimization problem in mobile wireless networks. In this article, we propose to use multi-population genetic algorithms (GAs) with an immigrants scheme to solve the dynamic SP routing problem in mobile ad hoc networks, which are the representative of new generation wireless networks. Two types of multi-population GAs are investigated. One is the forking GA in which a parent population continuously searches for a new optimum and a number of child populations try to exploit previously detected promising areas. The other is the shifting-balance GA in which a core population is used to exploit the best solution found and a number of colony populations are responsible for exploring different areas in the solution space. Both multi-population GAs are enhanced by an immigrants scheme to handle the dynamic environments. In the construction of the dynamic network environments, two models are proposed and investigated. One is called the general dynamics model, in which the topologies are changed because the nodes are scheduled to sleep or wake up. The other is called the worst dynamics model, in which the topologies are altered because some links on the current best shortest path are removed. Extensive experiments are conducted based on these two models. The experimental results show that the proposed multi-population GAs with immigrants enhancement can quickly adapt to the environmental changes (i.e., the network topology changes) and produce high-quality solutions after each change.

This work was supported by the Rising Star Programme in University of Bedfordshire, UK, the Engineering and Physical Sciences Research Council of UK under Grant EP/E060722/1 and Grant EP/E060722/2, the National Natural Science Foundation of China under Grant No. 61070162, No. 71071028 and No. 70931001.

Address correspondence to Xingwei Wang, College of Information Science and Engineering, Northeastern University, Shenyang 110004, China. E-mail: wangxw@mail.neu.edu.cn
INTRODUCTION

A mobile ad hoc network (Perkins, 2001; Toh, 2002; Siva Ram Murthy and Manoj, 2004) is a self-organizing and self-configuring multihop wireless network, comprising of a set of mobile hosts (MHs) that can move around freely and cooperate in relaying packets on behalf of one another. A mobile ad hoc network (MANET) supports robust and efficient operations by incorporating routing functionality into MHs. In MANETs, unicast routing establishes a multihop forwarding path for two nodes beyond the direct wireless communication range. Routing protocols also maintain the connectivity when links on these paths break because of node movement, battery drainage, radio propagation, or wireless interference. In multihop networks, routing is one of the most important issues that has significant impact on the performance of networks.

In this article, we investigate the shortest path (SP) routing, which is concerned with finding the shortest path from a specific source to a specific destination in a given network while minimizing the total cost associated with the path. The SP routing problem has been investigated extensively. It involves a classical combinatorial optimization problem arising in many design and planning contexts (Ali and Kamoun, 1993; Ahn et al. 2001). There are several search algorithms for the SP routing problem: the Dijkstra’s algorithm, the breadth-first search algorithm, the Bellman-Ford algorithm, etc. All these algorithms have a polynomial time complexity. Therefore, they will be effective in fixed infrastructure wireless or wired networks. However, they exhibit unacceptably high computational complexity for real-time communications involving rapidly changing network topologies (Ahn et al. 2001; Ahn and Ramakrishna, 2002). Because the algorithms with a polynomial time complexity are not suitable for the real-time computation of shortest paths, quite a few research works have been conducted to solve the SP routing problems using artificial intelligence techniques, for example, artificial neural networks (ANNs) (Ahn et al. 2001), genetic algorithms (GAs) (Ahn and Ramakrishna, 2002), and particle swarm optimization (PSO) (Mohemmed, Sahoo, and Geok 2008).

However, all these algorithms still mainly address the static SP routing problem. When the network topology changes, they will regard it as a new network and will restart the algorithms over the new topology. It is well known that the topology changes rapidly in MANETs as a result of the characteristics of wireless networks, for example, battery exhaustion and node mobility. Therefore, for the dynamic SP routing problem (DSPRP) in MANETs, these algorithms are not good choices because they require frequent restarts and cannot meet the real-time requirement. Furthermore, in practice, the routing problem has more requirements on robustness than is optimum. It means that for a continuously changing problem, a
new routing path should have the least modification to the old one. Therefore, for the DSPRP in a changing network environment, we need to employ new appropriate approaches.

Because the static SP routing problem is a combinatorial optimization problem, the dynamic SP routing problem turns out to be one dynamic optimization problem (DOP). In recent years, studying evolutionary algorithms (EAs) for DOPs has attracted a growing interest because of the importance of DOPs 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. One of the possible approaches is to maintain and reintroduce the diversity during the run of EAs, in other words, the immigrants schemes (Yang and Tinos, 2007; Tinos and Yang, 2007; Yang, 2008). A multi-population approach (Branke et al. 2000) is also an effective technique for DOPs. In a multi-population GA (MPGA), some populations are responsible for exploiting and others are responsible for exploring. By both exploiting and exploring the solution space, MPGAs can adapt well to the environmental changes.

In this article, the immigrants-enhanced multi-population GAs (iMPGAs) are implemented and applied to solve the DSPRP in MANETs. These algorithms can keep running over the continuously changing topologies and avoid the expensive and inefficient restart. Based on two different types of MPGAs,—forking GA (FGA) (Tsutsui, Fujimoto, and Ghosh 1997) and shifting balance GA (SBGA) (Oppacher and Wineberg, 1999)—two algorithms have been developed and they are denoted as iMPGA1 and iMPGA2, respectively. First, the specific components of the proposed GAs are designed for the DSPRP. Then, in either iMPGA1 or iMPGA2, a large population is created, which will split into several small populations after evolving for a certain time. These small populations continue the search by either exploiting or exploring the solution space. Once the topology is changed, all the small populations are processed in an appropriate way and are then merged together. At each generation, in order to enhance the diversity, a small number of random immigrants are added into the single whole population or the small populations which are responsible for exploring. This process is repeated for each change interval.

The difference between iMPGA1 and iMPGA2 lies in the job assigned to the populations. In iMPGA1, a parent population is responsible for exploring, and all the other child populations are responsible for exploiting. In iMPGA2, a core population is responsible for exploiting, and all the other colony populations are responsible for exploring. In addition, because the end-to-end delay (Parsa, Zhu, and Garcia-Luna-Aceves 1998) is an important quality-of-service (QoS) metric in order to guarantee the
real-time data delivery, we require the routing path to satisfy the delay con-
straint. For comparison purposes, we also implement the standard GA (SGA), the restart GA (RGA), and the random immigrants GA (RIGA). By simulation experiments, we evaluate their performances on the DSPRP. The experimental results show that both iMPGA1 and iMPGA2 significantly outperform the other three GA methods. It is verified that both of them work well for the dynamic real-world network problem.

The rest of this article is organized as follows. We present the MANET network model and the DSPRP model studied in this paper in “Model.” “Specialized GA for the SP Routing Problem” describes the design of a specialized GA for the static SP routing problem. “Random Immigrants GA” briefly reviews the random immigrants scheme for DOPs. The proposed GAs that integrate the multi-population and immigrants schemes and the specialized GA for the DSPRP are described in “Immigrants-Enhanced Multi-Population GAs”. The extensive experimental study and relevant analysis are presented in “Experimental Study”. Finally, “Conclusions” ends this article with some discussions on future work.

**MODEL**

In this section, we first present our MANET network model and then formulate the DSPRP in MANETs. We consider a MANET operating within a fixed geographical region. We model it by an 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 until 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.

Here, we summarize some notations that we use throughout this paper.

- \( G_0(V_0, E_0) \), the initial MANET topology graph.
- \( G_i(V_i, E_i) \), the MANET topology graph after the \(i\)th change.
- \( s \), the source node.
- \( r \), the destination node.
- \( P_i(s, r) \), a path from \(s\) to \(r\) on the graph \( G_i \).
- \( d_l \), the transmission delay on the communication link \(l\).
- \( c_l \), the cost of the communication link \(l\).
- \( \Delta(P_i) \), the total transmission delay on the path \(P_i\).
- \( C(P_i) \), the total cost of the path \(P_i\).
The DSPRP on the considered MANET can be informally described as follows: Initially, given a network of wireless routers, a delay upper bound, a source node, and a destination node, we wish to find a delay-bounded, least-cost, loop-free path on the topology graph. Then, periodically or stochastically, as a result of energy conservation or other issues, some nodes are scheduled to sleep and some sleeping nodes are scheduled to wake up. Therefore, the network topology changes from time to time. The objective of our problem is to quickly find the new optimal delay-constrained, least-cost acyclic path after each topology change.

More formally, consider a MANET $G(V, E)$ and a unicast communication request from the source node $s$ to the destination node $r$ with the delay upper bound $\Delta$. The dynamic delay-constrained shortest path problem is to find a series of paths $\{P_i\mid i \in \{0, 1, \ldots\}\}$ over a series of graphs $\{G_i\mid i \in \{0, 1, \ldots\}\}$, which satisfy the delay constraint as shown in Equation (1) and have the least-cost path as shown in Equation (2).

\[
\Delta(P_i) = \sum_{l \in P_i(s, r)} d_l \leq \Delta. \quad (1)
\]

\[
C(P_i) = \min_{P \in G_i} \left\{ \sum_{l \in P(s, r)} c_l \right\}. \quad (2)
\]

**SPECIALIZED GA FOR THE SP ROUTING PROBLEM**

This section describes the design of the GA for the SP routing problem. The GA operations consist of several key components: genetic representation, population initialization, fitness function, selection scheme, crossover, and mutation. A routing path consists of a sequence of adjacent nodes in the network. Hence, it is a natural choice to adopt the path-oriented encoding method. For the routing problems, the path-oriented encoding and the path-based crossover and mutation are also very popular (Ahn and Ramakrishna, 2002; Din, 2005).

**Genetic Representation**

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 (indicated by the gene of the locus). The gene of the first locus is for the source node and the gene of the last locus is for the destination node. The length of a routing path should not exceed the maximum length $|V_0|$, where $V_0$ is the set of nodes in the MANET. Chromosomes are encoded under the delay constraint. In case it is violated, the encoding process is usually repeated in order to satisfy the delay constraint.
Population Initialization

In GA, each chromosome corresponds to a potential solution. The initial population $Q$ is composed of a certain number, denoted as $q$, of chromosomes. In order to explore the genetic diversity, for each chromosome in our algorithm, the corresponding routing path is randomly generated. We start to search a random path from $s$ to $r$ by randomly selecting a node $v_1$ from $N(s)$, the neighborhood of $s$. Then, we randomly select a node $v_2$ from $N(v_1)$. This process is repeated until $r$ is reached. Thus, we get a random path $P(s, r) = \{s, v_1, v_2, \ldots, r\}$. Because the path should be loop-free, the nodes that are already included in the current path are excluded, thereby avoiding re-entry of the same node. The initial population is generated as follows:

[Step 1:] Start ($i = 0$);
[Step 2:] Generate chromosome $Ch_i$: search a random loop-free path $P(s, r)$;
[Step 3:] $i = i + 1$. If $i < q$, go to Step 2; otherwise, stop.

Thus, the initial population $Q = \{Ch_0, Ch_1, \ldots, Ch_{q-1}\}$ is obtained.

Fitness Function

Given a solution, we should accurately evaluate its quality (i.e., fitness value), which is determined by the fitness function. In our algorithm, we aim to find the least-cost path between the source and the destination. Our primary criterion of the solution quality is the path cost. Therefore, among a set of candidate solutions (i.e., unicast paths), we choose the one with the least-path cost. The fitness value of chromosome $Ch_i$ (representing the path $P_i(s, r)$), denoted as $F(Ch_i)$, is given by:

$$F(Ch_i) = \left[ \sum_{l \in P_i(s,r)} c_l \right]^{-1}.$$  \hspace{1cm} (3)

The proposed fitness function involves only the total path cost. As mentioned previously, the delay constraint is checked for each chromosome during the course of the run.

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 chromosome is based on the fitness value. We adopt the scheme of pair-wise tournament selection without replacement (Lee et al. 2008) because it is simple and effective. The tournament size is 2.

**Crossover and Mutation**

Usually, the performance of a GA depends on two basic genetic operators, crossover and mutation. Crossover processes the current solutions in order to find better ones. Mutation helps a GA jump away from local optima (Ahn and Ramakrishna 2002). The type and implementation of crossover and mutation operators depend on the encoding as well as the problem being solved.

In our algorithms, because chromosomes are expressed by the path structure, we adopt the single-point crossover to exchange partial chromosomes (subpath) at positionally independent crossing sites between two chromosomes (Ahn and Ramakrishna 2002). With the crossover probability, each time we select two chromosomes, \(Ch_i\) and \(Ch_j\), for crossover. Both \(Ch_i\) and \(Ch_j\) should possess at least one common node. Among all the common nodes, one node, denoted as \(v\), is randomly selected. Suppose that there is a path consisting of two parts: \((s \rightarrow v)\) and \((v \rightarrow r)\) in \(Ch_i\) and there is a path consisting of two parts: \((s \rightarrow v)\) and \((v \rightarrow r)\) in \(Ch_j\). The crossover operation exchanges the subpaths \((v \rightarrow r)\) and \((v \rightarrow r)\).

The population will undergo the mutation operation after the crossover operation is performed. With the mutation probability, each time we select one chromosome \(Ch_i\) on which one gene is randomly selected as the mutation point (i.e., mutation node), denoted as \(v\). The mutation will replace the subpath \((v \rightarrow r)\) by a new random subpath.

Both crossover and mutation may produce new chromosomes that are infeasible solutions. Therefore, we check whether the paths represented by the new chromosomes are acyclic. If not, a repair function (Oh, Ahn, and Ramakrishna 2006) will be applied to eliminate the loops. The delay checking is integrated into the crossover and mutation operations to ensure that all the new chromosomes produced by crossover or mutation satisfy the delay constraint.

**RANDOM IMMIGRANTS GA**

In stationary environments, convergence at a proper pace is really what we expect in order for GAs to locate the optimal 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 in order to maintain their adaptability. To address this problem, the random-immigrants approach is a quite natural and simple way (Cobb and Grefenstette 1993). It was first proposed by Grefenstette with the inspiration from the flux of immigrants that wander in and out of a population between two generations in nature (Grefenstette, 1992). 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 order to avoid random immigrants disrupting 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, For example, 0.2. The pseudocode for the RIGA is shown in Figure 1. Within RIGA, for each generation $t$, after the normal genetic operations (i.e., selection and recombination), a set of $r_i \times n$ individuals are generated randomly, where $n$ is the population size and $r_i$ is the ratio of the number of random immigrants to the population size. The generated individuals then act as immigrants to replace the worst individuals in the current population.

**IMMIGRANTS-ENHANCED MULTI-POPULATION GAs**

The traditional GA has a single population searching through the entire search space. The multi-population approach tries to divide the
search space into several parts and then uses a number of small populations to search them separately. Normally, one of the small populations acts as the parent population or the core population. In the forking GA (Tsutsui, Fujimoto, and Ghosh 1997), the parent population continuously searches for a new optimum, whereas a number of child populations try to exploit previously detected promising areas. In the shifting balance GA (Oppacher and Wineberg 1999), the core population is used to exploit the best solution found, whereas the colony populations are responsible for exploring different areas in the solution space.

In this article, we develop two immigrants-enhanced, multi-population GAs, denoted iMPGA1 and iMPGA2, by following the ideas of FGA and SBGA, respectively. However, in order to address the DSPRP in MANETs, we still need to make a specific design in our algorithms. In order to measure the similarity degree between two individuals, we define the distance between any two individuals by counting the number of same links shared by them. The more same links they share, the closer they are. For a small population that is responsible for exploration, we expect that the individuals in the population are kept far away from each other. Thus, the population can search a wide area. For a small population that is responsible for exploitation, we expect that the individuals in the population stay close to an optimum and perform a great deal of local search.

iMPGA1

In iMPGA1, initially we randomly generate a large single population. Suppose the problem changes every $I$ generations ($I$ is called the change interval); all individuals of the whole population will evolve together for $\lceil I/2 \rceil$ generations. Then, the single population is split into three small populations. Of them, one small population will act as the parent population for exploring the search space, and the other two will act as the child populations for exploiting the local area. To achieve this goal, we develop the following splitting method: First, we identify the present optimal individual $Pop_{1_{opt}}$ in the whole population. Then, we find its closest neighbor $Pop_{1}$, 2nd closest neighbor $Pop_{2}$, 3rd closest neighbor $Pop_{3}$, until we get the $(m-1)$th closest neighbor $Pop_{m-1}$. All these $m$ individuals form the first child population \{$Pop_{1_{opt}}, Pop_{1}, Pop_{2}, Pop_{3}, \ldots, Pop_{m-1}$\}. Among all the remaining individuals of the whole population, the optimal one is identified again, denoted as $Pop_{2_{opt}}$. Similarly, among the remaining individuals, we determine its closest neighbor $Pop_{2}$, 2nd closest neighbor $Pop_{2}$, 3rd closest neighbor $Pop_{3}$, to the $(m-1)$th closest neighbor $Pop_{m-1}$. All these $m$ individuals form the second child population \{$Pop_{2_{opt}}, Pop_{2}, Pop_{3}, \ldots, Pop_{m-1}$\}. All the remaining individuals form the third population $Pop_{3}$, or, the parent population.
These three small populations keep evolving independently until the change interval ends. When a new change is detected (i.e., the topology is modified), all of them need to be processed appropriately and then merged together in order to form a single population again. We develop the following processing method for these small populations to adapt to the environmental changes. For each of them, if the optimal individual in it becomes infeasible, the whole population will be replaced by random immigrants. Otherwise, if the optimal individual in it is feasible, only the infeasible individuals in the population will be replaced by random immigrants. The reason to do so is that if the optimal individual becomes infeasible, all the other individuals are also infeasible and therefore the whole population should be abandoned. However, if the optimal individual is suitable for the new environment, we want to keep other individuals that are also suitable for the new environment. Thus, the useful information in the old environment can be reused to guide the search in the new environment.

In this algorithm, at each generation, a small number of random immigrants is replaced into the population. Before the splitting of the population, all the random immigrants are imported into the single population to replace the worst ones. After the splitting, all the random immigrants are imported only into the parent population because it is responsible for exploring.

**iMPGA2**

In iMPGA2, we also randomly generate a large initial population. For each given change interval $I$, all the individuals of the whole population will evolve together for $\lceil I/2 \rceil$ generations. Then, the single population is also split into three small populations. Of them, one small population will act as the core population for exploiting the promising area and the other two will act as the colony populations for exploring the search space. To achieve this goal, we develop the following splitting method by which the core population is first formed and then the two colony populations are constructed.

First, we identify the current optimal individual $Pop_{1, opt}$ in the whole population. Then, we find its closest neighbor $Pop_{1, 1}$, 2nd closest neighbor $Pop_{1, 2}$, 3rd closest neighbor $Pop_{1, 3}$, until we get the $(m' - 1)$th closest neighbor $Pop_{1, m' - 1}$. All these $m'$ individuals form the core population $\{Pop_{1, opt}, Pop_{1, 1}, Pop_{1, 2}, Pop_{1, 3}, \ldots, Pop_{1, m' - 1}\}$. Among all the remaining individuals of the whole population, we identify two individuals that share the fewest overlapping links (may be zero), saying $Ind1$ and $Ind2$. Based on $Ind1$ and $Ind2$, we begin to construct the two colony populations $Pop2$ and
For $Pop_2$, its present optimal individual $Pop_2_{\text{opt}}$ is $Ind_1$. For $Pop_3$, its present optimal individual $Pop_3_{\text{opt}}$ is $Ind_2$. Then, for each remaining individual $Ind_{\text{hair}}$, we check its distance to $Pop_2_{\text{opt}}$ and $Pop_3_{\text{opt}}$, respectively, and put $Ind_{\text{hair}}$ into the population where the distance from $Ind_{\text{hair}}$ is larger, or the overlapping links are fewer. The present optimal individual in the corresponding population will be updated accordingly. This process is repeated until all the remaining individuals have been put into one of the two populations. By this method, intuitively, the two colony populations $Pop_2$ and $Pop_3$ can achieve a good diversity level. However, the size of either population varies.

These three small populations keep evolving independently until the change interval ends. When a new change is detected, in other words, the topology is modified, all of them need to be processed appropriately and then merged together in order to form a single population again. The processing method is the same as the one used in iMPGA1. In iMPGA2, at each generation, we also replace a small number of random immigrants into the population to enhance the diversity. The importation of immigrants follows the same procedure as that used in iMPGA1.

**EXPERIMENTAL STUDY**

We implement iMPGA1, iMPGA2, RIGA, SGA, and RGA for the DSPRP by simulation. In SGA, if the change makes one individual in the current population become infeasible (e.g., one link in the corresponding path is lost after the change), we add penalty value to that individual. By simulation experiments, their performances are evaluated in a continuously changing wireless network. The simulation software for both the dynamic topology generation and the various GAs were 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.

**Experimental Design**

**Common Issues**

Because 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 $200 \times 200$ 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 set to 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 $I$ generations ($I$ is the change interval), the network topology is changed in a way corresponding to the dynamics model used. It can be seen that the variable $I$ 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 mutation probability was set to 0.1. For RIGA, iMPGA1, and iMPGA2, the ratio of the number of immigrants to the population size was set to 0.2. Both the source and destination nodes were randomly selected. The delay upper bound $\Delta$ was set to be two times of the minimum end-to-end delay.

In order to have fair comparisons among GAs, the population size and immigrants ratio were set such that each GA has 120 fitness evaluations per generation as follows:

\[
(1 + r_i) \times n = 120, \tag{4}
\]

where $n$ is the whole population size. Hence, we have $n = 120$ for SGA and RGA, where $r_i = 0$, and $n = 100$ for RIGA, iMPGA1, and iMPGA2, where $r_i = 0.2$. At each generation, for each algorithm, we select the best individual from the current population and output the cost of the shortest path represented by it. For each experiment of an algorithm, 10 independent runs were executed with the same set of random seeds. For each run, 19 environmental changes were allowed under both dynamics models, which are equivalent to 100, 200, and 300 generations, including the initial environment, for $I = 5, 10, \text{ and } 15$, respectively. For each run, the best-of-generation fitness was recorded at every generation. The overall offline performance of a GA on a DOP is defined as:

\[
F_{BOG} = \frac{1}{G} \sum_{i=1}^{G} \left( \frac{1}{N} \sum_{j=1}^{N} F_{BOG_{ij}} \right), \tag{5}
\]

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 $j$. The offline performance is $F_{BOG}$, in other words, the best-of-generation fitness averaged over the 10 runs and then over the data-gathering period.

**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 because some links are lost and other links reappear. The nodes are randomly selected and, thereby, the affected links may belong to the present shortest path or not. Both 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 because each time the change affects only some 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, 3, and 4, respectively. Thus, by the number of nodes changed per time, we have three different series of topologies. When $M$ is set to 2, 3, and 4, we generate the topology series $\#2$, $\#3$, and $\#4$, respectively. Each of these three series has 20 different topologies. All the experiments under the general model were based on these three topology series. We set up experiments to evaluate the appropriate population sizes, the impact of the change interval, and the improvements of iMPGA1 and iMPGA2 over traditional GAs and RIGA.

**Environmental Changes Under the Worst Dynamics Model**

In the worst dynamics model, every $I$ generations, the present best shortest path is first identified. Then, a certain number (say, $U$) of links on the path 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 is ready for the coming change. The population is severely affected by each topology change because the optimal solution and possibly some other good solutions become infeasible suddenly. 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 19 changes and derive 20 different topologies in the whole algorithm run. We set up the experiments to
evaluate the impact of the change interval and the change severity, and the improvements of iMPGA1 and iMPGA2 over traditional GAs and RIGA.

**Experimental Results and Analysis**

*Under the General Dynamics Model*

In iMPGA1, there are one parent population and two child populations. In order to make iMPGA1 work at its best performance, we need to determine the appropriate sizes for all the populations. For simplicity, the two child populations are set to have the same size. We vary the ratio of the child population size to the whole population size from 0.2 to 0.4. Because we expect iMPGA1 to work well in all the circumstances (i.e., whether the environmental changes are significant or not), we test iMPGA1 with different child population sizes over topology series #2 and #4, respectively. The topology series #2 represents the environment where the changes are less significant whereas topology series #4 represents the environment where the changes are significant. Figures 2(a) and (b) show the dynamic performance of iMPGA1 with different child population sizes. From Figures 3(a) and (b), it can be seen that iMPGA1 achieves the best performance when the ratio is set to 0.3. It means that the size of the parent population is 40 and the size of either child population is 30. In the following experiments, this size combination will be used for iMPGA1.

In iMPGA2, there are one core population and two colony populations. Similarly, in order to make iMPGA2 work at its best performance, we need to determine the appropriate population sizes. However, in iMPGA2, the sizes of the two colony populations vary from time to time because of the construction method in use. Therefore, we can determine the size only for the core population. We vary the ratio of the core population size to the whole population size from 0.4 to 0.6. We also test iMPGA2 with different core population sizes over topology series #2 and #4, respectively. Figures 3(a) and (b) show the results. From Figures 3(a) and (b), it can be seen that iMPGA2 achieves the best performance when the ratio of the core population size to the whole population size is set to 0.4. It means that the ideal core population size is 40 and the two colony populations have the total size of 60. This size setting applies to all the experiments on iMPGA2 below.

From Figures 2 and 3, we can see that sometimes when a change occurs, the algorithm is not affected. The reason is 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 shortest path represented by the optimal individual, the optimal individual has a very high probability
to stay in the population unaffected. Therefore, regarding the algorithm performance under different environmental change settings, we investigate the change interval only under the general dynamics model. However, under the worst dynamics model, we will investigate both the change interval and the change severity.

We set $I$ to 5, 10, and 15, respectively, to see the impact of the change interval (i.e., the change frequency) on the performance of algorithms. Topology series #3 was used as the general dynamic environment. 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 a smaller interval does. We compare the quality of solutions obtained by iMPGA1 at different intervals. However, one problem is that the total generations are different for different intervals (i.e., 100, 200, and 300 versus the interval 5, 10, and 15 when there are 20 different topologies). Because the number of change points (i.e., the generation at which a new topology is applied) is

\[ \text{FIGURE 2} \quad \text{Comparison results of the quality of solutions for iMPGA1 with different child population sizes over (a) topology series #2 and (b) topology series #4.} \]
the same for all the intervals, we take the data at each change point and its previous two generations and next two generations. Thus, the three different data sets can be aligned over the three different intervals. Figure 4 shows the comparison results. Because the generation number does not correspond to the actual number when the interval is 10 or 15, we rename it as pseudo generation in Figure 4.

From Figure 4, it can be seen that the solution quality becomes better when the change interval is increased from 5 to 10. However, when the change interval is increased from 10 to 15, the results in both Figure 4(a) and (b) are slightly different. Therefore, it will be a proper choice for both iMPGA1 and iMPGA2 to set the change interval to 10.

The quality of solutions is the most important metric with which to evaluate the algorithm performance. We compare iMPGA1 and iMPGA2 with both traditional GAs (i.e., SGA and RGA) and random immigrants GA (i.e., RIGA). We set the change interval to 10 and test the algorithms over topology series #2 and #4, respectively. Since both iMPGA1 and
iMPGA2 are dynamic GAs, which are specifically designed for the dynamic environment, they should show a better performance than the traditional GAs over the test DSPRP. Figure 5 shows the comparison results. It can be seen that both iMPGA1 and iMPGA2 achieve better solutions than the other three algorithms. RGA shows the worst performance as a result of frequent restarts, which do not give the population enough time to evolve. SGA works better than RGA but worse than RIGA in most of the cases. In Figure 5(a), after 50 generations, iMPGA1, iMPGA2, and RIGA show a similar performance. The reason is that the environmental changes did not affect them during these time slots.

**Under the Worst Dynamics Model**

Under the worst dynamics model, we first investigate the impact of the change interval on the performance of algorithms. Similarly, we set $I$ to 5, 10, and 15, respectively. Here, the number of links removed per change was fixed to 2. We compare the quality of solutions obtained by iMPGA1 and
iMPGA2 at different change intervals. In order to align the three data sets, we employ the same method as that used under the general dynamics model. Figure 6 shows the comparison results.

From Figure 6, it can also be seen that the solution quality becomes better when the change interval is increased from 5 to 10. Here, what is different from the results under the general dynamics model is that when the change interval is increased from 10 to 15, the quality of solutions is also improved significantly. The reason is that under the worst dynamics model, the environmental changes are significant and bring in drastic changes to the network topology. The generations that the whole population evolved at interval 15 are much larger than the generations that the whole population evolved at interval 10. A longer evolution process before a change occurs brings better solutions. Therefore, the capability of the multi-population GAs in searching the optimum has been significantly enhanced. In traditional GAs, the population may converge after evolving for a while. However, in both iMPGA1 and iMPGA2, because of the
introduction of random immigrants, the population can keep evolving and hence avoid being trapped in a local optimum.

Second, in order to evaluate the effect of the change severity on the performance of algorithms, we vary the number of links removed per change from 1 to 3. Meanwhile, the change interval was fixed to 10. With more links removed from the network, the environmental changes become more severe. Furthermore, because all the removed links come from the present best path, some individuals, including the optimal one in the population, become infeasible. It is also possible that the whole population becomes infeasible if each individual contains at least one removed link. The more the links removed, the higher the probability of an individual becoming infeasible.

Figure 7 shows the comparison results in terms of the change severities. It can be seen that the quality of solutions is best when the number of links removed per change was set to 1 and the worst when the number was set to 3. However, the difference between iMPGA*:1 and iMPGA*:2 is

![Figure 6](image-url)
less significant than the difference between iMPGA*C3:2 and iMPGA*C3:3, where iMPGA*C represents either iMPGA1 or iMPGA2. The reason is that the increase in the number of links removed per change is not proportional to the increase in the change severity. Removing one more link will bring in a much higher change severity to the network and therefore affect many more individuals in the population.

Third, we also compare iMPGA1, iMPGA2 with SGA, RGA, and RIGA under the worst dynamics model. We set the change interval to 10 and the number of links removed per change to 2, respectively. Figure 8 shows the comparison results. Because we do not have two topology series to test, we show the performance over generation 0 to 99 and generation 100 to 199, respectively. It can be seen that iMPGA1 and iMPGA2 outperform both of the traditional GAs and RIGA. RGA still shows the worst performance as a result of frequent restarts. Although RIGA is also a dynamic GA, it does not utilize the approach of multiple populations to help search. The results

FIGURE 7 Comparison results of the quality of solution under the worst dynamics model in terms of different change severities for (a) iMPGA1 and (b) iMPGA2.
that both iMPGA1 and iMPGA2 perform better than RIGA verify that the multi-population approach helps improve the capability of GAs in handling dynamic environments.

CONCLUSIONS

The static SP routing problem considers the static network topology only. Intuitively, it is a much more challenging task to deal with the dynamic SP routing problem in a rapidly changing network environment such as MANETs than to solve the static one in a fixed infrastructure. Recently, there has been a growing interest in studying GAs for dynamic optimization problems. Among approaches developed for GAs to deal with DOPs, the multi-population GA aims at handling the problem dynamics by using multiple small populations to perform both exploration and exploitation. The random-immigrants scheme is another approach that maintains the
diversity of the population throughout the run by introducing new individuals into the current population.

In this article, we propose two immigrants-enhanced multi-population GAs (i.e., iMPGA1 and iMPGA2), which combine both the multi-population approach and immigrants scheme for the dynamic SP routing problem in MANETs. We design well the GA components for the SP routing problem and the immigrants-enhanced multi-population GAs. In order to investigate the performance of proposed GAs for the DSPRP in MANETs, two types of dynamic test environments are constructed to generate various dynamic network topologies with different change severities. Simulation experiments are conducted in a large scale MANET of 100 nodes. The experimental results show that both iMPGA1 and iMPGA2 are powerful techniques for solving the DSPRP and have potential to be applied to the real-world telecommunication network.

There are several relevant future works. One interesting work would be to further identify new dynamic optimization problems in mobile wireless networks (e.g., dynamic clustering, etc.) and evaluate GAs studied in this article based on the new network problems. Another future work would be to extend the problem background to the transport network, which also has the inherent characteristics of continuously changing topologies. We believe that these newly developed GAs can benefit the route planning in the transport network.

REFERENCES


