Multi-population Genetic Algorithms with Immigrants Scheme for Dynamic Shortest Path Routing Problems in Mobile Ad Hoc Networks

Hui Cheng and Shengxiang Yang

Department of Computer Science, University of Leicester
University Road, Leicester LE1 7RH, United Kingdom
{hc118,s.yang}@mcs.le.ac.uk

Abstract. The static shortest path (SP) problem has been well addressed using intelligent optimization techniques, e.g., artificial neural networks, genetic algorithms (GAs), particle swarm optimization, etc. However, with the advancement in wireless communications, more and more mobile wireless networks appear, e.g., mobile ad hoc network (MANET), wireless mesh network, etc. One of the most important characteristics in mobile wireless networks is the topology dynamics, that is, the network topology changes over time due to energy conservation or node mobility. Therefore, the SP problem turns out to be a dynamic optimization problem in mobile wireless networks. In this paper, we propose to use multi-population GAs with immigrants scheme to solve the dynamic SP problem in MANETs which is the representative of new generation wireless networks. The experimental results show that the proposed GAs can quickly adapt to the environmental changes (i.e., the network topology change) and produce good solutions after each change.

1 Introduction

A mobile ad hoc network (MANET) [11] is a self-organizing and self-configuring multi-hop wireless network, comprised of a set of mobile hosts (MHs) that can move around freely and cooperate in relaying packets on behalf of each other. In this paper, we investigate the shortest path (SP) routing, which concerns 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 problem has been investigated extensively. It involves a classical combinatorial optimization problem arising in many design and planning contexts [1,2].

There are several search algorithms for the SP problem: the Dijkstra's algorithm, the breadth-first search algorithm and the Bellman-Ford algorithm, etc. All these algorithms have polynomial time complexity. Therefore, they will be effective in fixed infrastructure wireless or wired networks. But, they exhibit unacceptably high computational complexity for real-time communications involving rapidly changing network topologies [2,3]. Since the algorithms with polynomial time complexity are not suitable for the real-time computation of shortest paths, quite a few research work have been conducted to solve SP problems using artificial intelligence techniques, e.g., artificial neural networks (ANNs) [2], genetic algorithms (GAs) [3], and particle swarm optimization (PSO) [7].

C. Di Chio et al. (Eds.): EvoApplications 2010, Part I, LNCS 6024, pp. 562–571, 2010.

However, so far all these algorithms mainly address the static SP problem. When the network topology changes, they will regard it as a new network and restart the solving process over the new topology. As is well known that the topology changes rapidly in MANETs due to the characteristics of wireless networks, e.g., battery exhaustion and node mobility. Therefore, for the dynamic SP problem in MANETs, these algorithms are not good choices since they require frequent restart and cannot meet the real-time requirement. Therefore, for the dynamic SP problem in a changing network environment, we need to employ new appropriate approaches.

In recent years, studying EAs for DOPs has attracted a growing interest due to its importance in EA's real world applications [14]. 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 [13], 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 diversity during the run of EAs, i.e., the immigrants schemes [15]. Multi-population approach [4] is also an effective technique for DOPs. In the multi-population GA (MPGA), some populations are responsible for exploiting and others for exploring. By both exploiting and exploring the solution space, MPGA can well adapt to the environmental changes.

In this paper, the multi-population GA with immigrants scheme is implemented and applied to solve the dynamic SP problem. The algorithm is denoted as iMPGA. 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 then merge together. At each generation, to enhance the diversity a small number of random immigrants are added into the single population or the small populations which are responsible for exploring. This process is repeated for each change interval. Since end-to-end delay [10] is a pretty important quality-of-service (QoS) metric to guarantee the real-time data delivery, we also require the routing path to satisfy the delay constraint. For comparison purposes, we also implement the Standard GA (SGA), the Restart GA, and the random immigrants GA (RIGA). By simulation experiments, we evaluate their performance on the dynamic SP problem. The results show that iMPGA significantly outperforms the other three GA methods.

2 Model

In this section, we first present our network model and then formulate the problem of dynamic SP routing. We consider a MONET operating within a fixed geographical region. We model it 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) can not 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 ith 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 on 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 problem of the dynamic SP routing 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, due to energy conservation or some other issues, some nodes are scheduled to sleep or 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 mobile ad hoc network $G(V, E)$ and a unicast communication request from the source node s to the destination node r with the delay upper bound Δ. The *dynamic delay-constrained shortest path problem* is to find a series of paths $\{P_i | i \in \{0, 1, ...\}\}$ over a series of graphs $\{G_i | i \in \{0, 1, ...\}\}$, which satisfy the delay constraint as shown in (1) and have the least path cost as shown in (2).

$$\Delta(P_i) = \sum_{l \in P_i(s,r)} d_l \leq \Delta \,. \tag{1}$$

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

3 Design of GA for SP Problem

This section describes the design of the GA for the SP 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 [3]. For the selection scheme, the pair-wise tournament selection without replacement [6] is employed and the tournament size is 2.

3.1 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 one 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 so as to satisfy the delay constraint.

3.2 Population Initialization

In GA, each chromosome corresponds to a potential solution. The initial population Q is composed of a certain number, denoted as n, of chromosomes. To explore the genetic diversity, in our algorithm, for each chromosome, 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 $P(s, r)=\{s, v_1, v_2, ..., r\}$. Since the path should be loop-free, the nodes that are already included in the current path are excluded, thereby avoiding reentry of the same node. The initial population is generated as follows.

Step 1: Start(j=0).

Step 2: Generate chromosome Ch_j: search a random loop-free path $P(s, r)$;

Step 3: j=j+1. If $j < n$, go to *Step 2*, otherwise, stop.

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

3.3 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 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_j (representing the path P), denoted as $F(Ch_j)$, is given by:

$$F(Ch_j) = [\sum_{l \in P(s,r)} c_l]^{-1} . \tag{3}$$

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

3.4 Crossover and Mutation

GA relies on two basic genetic operators - crossover and mutation. Crossover processes the current solutions so as to find better ones. Mutation helps GA keep away from local optima [3]. The performance of GA depends on them greatly. The type and implementation of operators depend on problem-specific encoding.

In our algorithm, since chromosomes are expressed by the path structure, we adopt single point crossover to exchange partial chromosomes (subpath) at positionally independent crossing sites between two chromosomes [3]. With the crossover probability, each time we select two chromosomes Ch_i and Ch_j for crossover. 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. In Ch_i, there is a path consisting of two parts: ($s \xrightarrow{Ch_i} v$) and ($v \xrightarrow{Ch_i} r$). In Ch_j, there is a path consisting of two parts: ($s \xrightarrow{Ch_j} v$) and ($v \xrightarrow{Ch_j} r$). The crossover operation exchanges the subpaths ($v \xrightarrow{Ch_i} r$) and ($v \xrightarrow{Ch_j} r$).

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 ($v \xrightarrow{Ch_i} r$) by a new random subpath.

Both crossover and mutation may produce new chromosomes which are infeasible solutions. Therefore, we check if the paths represented by the new chromosomes are acyclic. If not, repair functions [8] will be applied to eliminate the loops. Here the detail is omitted due to the space limit. All the new chromosomes produced by crossover or mutation satisfy the delay constraint since it has already been considered.

4 iMPGA: Multi-population GAs with Immigrants Scheme

The random immigrants approach was proposed by Grefenstette [5] with the inspiration from the flux of immigrants that wander in and out of a population between two generations in nature. It maintains the population diversity level by 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. In order to avoid that random immigrants disrupt the ongoing search progress too much, especially during the static period between two environmental changes, the ratio of the number of random immigrants to the population size, r_i, is usually set to a small value.

The traditional genetic algorithm has a single population searching through the entire search space. 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 genetic algorithms (FGAs) [12], the parent population continuously searches for new optimum, while a number of child populations try to exploit previously detected promising areas. In the Shifting Balance GA [9], the core population is used to exploit the best solution found, while the colony populations are responsible for exploring different areas in the solution space.

In this paper, we generally follow the idea of the FGAs. However, to address the dynamic SP problem, we still need to make specific design in our algorithm. To measure the similarity degree between two individuals, we define the distance between any two individuals by counting the same links shared by them. The more same links they share, the closer they are. For the parent population which is responsible for exploring, we expect that the individuals in it are kept far away from each other in the distance. Thus, the population can search a wide area. For a child population which is responsible for exploiting, we expect that the individuals in it stay close to an optimum and perform lots of local search.

In iMPGA, initially we randomly generate a large single population. For each given change interval I, the whole population will evolve together for $\lfloor I/2 \rfloor$ generations. Then the single population is split into three small populations. Of them, one small population will act as the parent population for exploring and the other two will act as the child populations for exploiting. To achieve this goal, we develop the following splitting method. First, we identify the present optimal individual $Pop1_{opt}$ in the whole population. Then we find its closest neighbor $Pop1_1$, 2nd closest neighbor $Pop1_2$, 3rd closest neighbor $Pop1_3$, till the (m-1)th closest neighbor $Pop1_{m-1}$. All these m individuals form the first

child population $\{Pop1_{opt}, Pop1_1, Pop1_2, Pop1_3, ..., Pop1_{m-1}\}$. Among all the remaining individuals of the whole population, the optimal one is identified again, denoted as $Pop2_{opt}$. Similarly, among the remaining individuals, we determine its closest neighbor $Pop2_1$, 2nd closest neighbor $Pop2_2$, 3rd closest neighbor $Pop2_3$, till the (m-1)th closest neighbor $Pop2_{m-1}$. All these m individuals form the second child population $\{Pop2_{opt}, Pop2_1, Pop2_2, Pop2_3, ..., Pop2_{m-1}\}$. All the remaining individuals form the third population, i.e., the parent population.

These three small populations keep evolving independently till 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 also want to keep other individuals which 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 our algorithm, at each generation, a small number of random immigrants are added 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 only imported into the parent population since it is responsible for exploring.

5 Experimental Study

We implement iMPGA, RIGA, SGA, and Restart GA for the dynamic SP problem by simulation. For RIGA and 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, we evaluate their performance in a continuously changing mobile ad hoc network.

5.1 Experimental Design

All the algorithms start from the initial network topology of 100 nodes. Then every I generations, the present best path is identified and 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. However, just before the next change occurs, the network topology will be recovered to its original state and ready for the oncoming change. The population is severely affected by each topology change since the optimal solution and possibly some other good solutions become infeasible suddenly. Considering that the optimal path length could not be a large number, we let U range from 1 to 3 to see the effect of the change severity. Under this network dynamics model, the

topology series cannot be generated in advance because every change is correlated with the running of the algorithm. We allow 10 changes in each run of the algorithms. We set up experiments to evaluate the impact of the change interval and the change severity, and the improvements over traditional GAs and RIGA.

In all the experiments, the whole population size n is set to 100, the child population size m is set to 20, and the mutation probability is set to 0.1. For the random immigrants scheme, r_i is set to 0.2. In addition, we set the number of changes to 19 and therefore the algorithms will work over 20 different but highly-correlated network topologies (the initial topology plus the 19 changed topologies). Both the source and destination nodes are randomly selected and they are not allowed to be scheduled in any change. The delay upper bound Δ is set to be 2 times of the minimum end-to-end delay.

5.2 Experimental Results and Analysis

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. We repeat each experiment 10 times and get the average values of the best solutions at each generation. First, we investigate the impact of the change interval on the algorithm performance. We set I to 5, 10, and 15 separately to see the impact of change interval (i.e., change frequency) on the algorithm performance. Here the number of links removed per change is fixed to 2.

When the change interval is 5, the population evolves only 5 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 compare the quality of solutions obtained by iMPGA 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. 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 left two and right two generations. Thus, the three different data sets can be aligned over the three different intervals. Fig. 1 shows the comparison results in terms of the change intervals.

Since the generation number does not correspond to the actual number when the interval is 10 or 15, we rename it as pseudo generation. From the two subfigures, 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 subfigures are slightly different. In Fig. 1(a), the iMPGA shows competing performance for both intervals. For five times, the performance at interval 10 is better and for the other five times, the performance at interval 15 is better. In Fig. 1(b), the performance at interval 15 is better than the performance at interval 10 for all the times. The reason is that in Fig. 1(b), the generations that the whole population has evolved at interval 15 are much larger than the generations that the whole population has evolved at interval 10. Longer evolution brings better solutions. Therefore, the capability of the multi-population genetic algorithm in searching the optimum has been significantly enhanced. In traditional GA, the population may converge after evolving for a while. However, in iMPGA, due to the introduction of random immigrants, the population can keep evolving and get out of the trap in the local optimum.

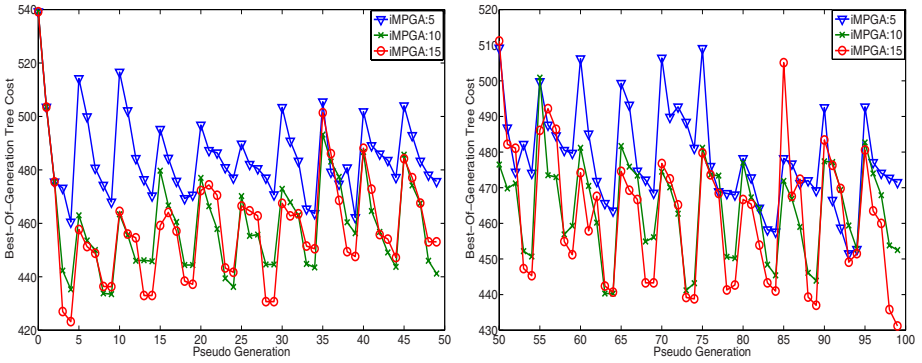

Fig. 1. Comparison of the solution quality of iMPGA with different change intervals from (a) generation 0-49 and (b) generation 50-99

To evaluate the effect of the change severity on the algorithm performance, we vary the number of links removed per change from 1 to 3. Meanwhile, the change interval is fixed to 10 since it is a reasonably good change frequency as shown in the above experiments. With more links removed from the network, the environmental changes become more severe. Furthermore, since 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 being infeasible.

Fig. 2 shows the comparison results in terms of the change severities. It can be seen that the quality of solution is the best when the number of links removed per change is 1 and the worst when the number is 3. However, the difference between iMPGA:1 and iMPGA:2 is less significant than the difference between iMPGA:2 and iMPGA:3. The reason is that the increase in the number of links removed per change is not proportional to the increase in the change severity. To remove one more link will bring a much higher change severity to the network and therefore affect much more individuals in the population. Another interesting point is that in Fig. 2(b), the performance differences between the algorithms with different change severities become less than the differences in Fig. 2(a). It is also due to the enhanced search capability of the multi-population algorithm after long time evolution as explained above.

The quality of solution is the most important metric to evaluate the algorithm performance. We compare iMPGA with both traditional GAs and random immigrants GA. The two traditional GAs are Standard GA and Restart GA. We set the change interval to 10 and the number of links removed per change to 2, respectively. Since iMPGA is a dynamic GA which is specifically designed for the dynamic environment, it should show better performance than the traditional GAs over our dynamic shortest-path problem. Fig. 3(a) shows the comparison results between iMPGA and traditional GAs. It can be seen that iMPGA achieves better solutions than both of the traditional GAs. Restart GA shows the worst performance due to frequent restart which does not give the

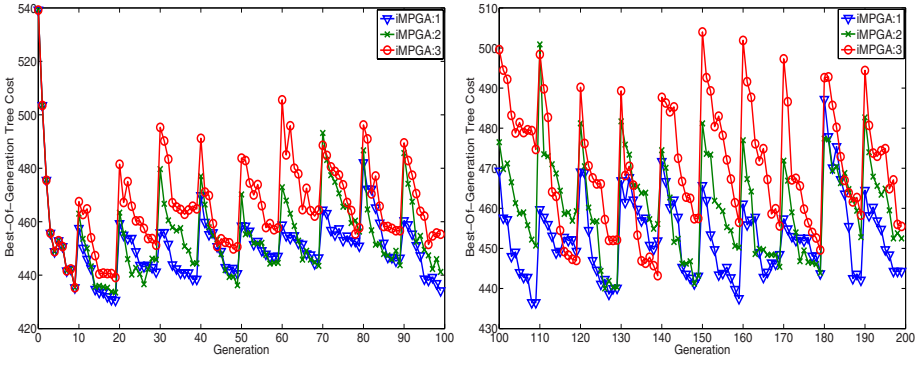

Fig. 2. Comparison of the solution quality of iMPGA with different change severities from (a) generation 0-99 and (b) generation 100-199

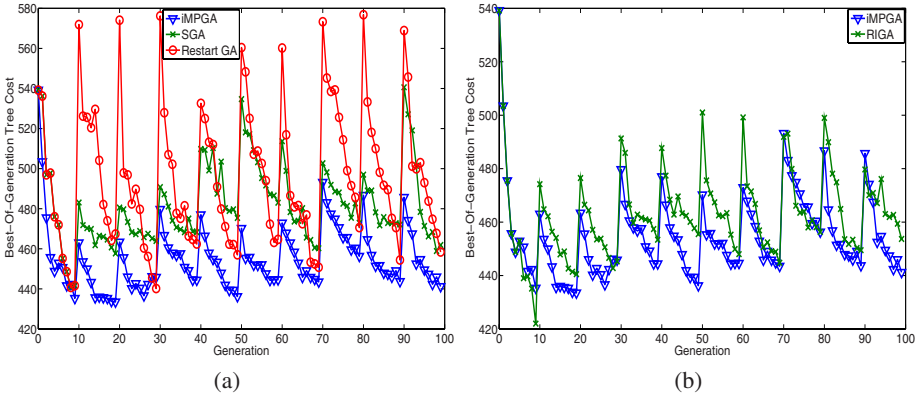

Fig. 3. Comparison of the solution quality of iMPGA against (a) traditional GAs and (b) RIGA

population enough time to evolve. Although RIGA is also a dynamic GA, it does not utilize the approach of multiple populations to help search. Fig. 3(b) shows the comparison results between iMPGA and RIGA. It shows that iMPGA performs better than RIGA. This verifies that the multi-population approach helps improve the capability of GA in handling dynamic environment.

6 Conclusions

The static SP problem considers the static network topology only. Intuitively, it is a much more challenging task to deal with the dynamic SP 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. Random immigrants scheme is another approach which maintains the diversity of the population throughout the run via introducing new individuals into the current population. In this paper, we propose iMPGA which combines both the multi-population approach and immigrants. We well design the GA components for the SP problem and the multi-population GA with immigrants scheme. Simulation experiments are conducted in a large scale MANET. The results show that iMPGA is a powerful technique for solving the dynamic SP problem and has potential to be applied to the real-world telecommunication network. In the future work, we will further investigate the robustness of the solutions provided by us.

Acknowledgement

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of UK under Grant EP/E060722/1.

References

1. Ali, M.K., Kamoun, F.: Neural networks for shortest path computation and routing in computer networks. IEEE Trans. on Neural Networks 4(6), 941–954 (1993)
2. Ahn, C.W., Ramakrishna, R.S., Kang, C.G., Choi, I.C.: Shortest path routing algorithm using hopfield neural network. Electronics Letters 37(19), 1176–1178 (2001)
3. Ahn, C.W., Ramakrishna, R.S.: A genetic algorithm for shortest path routing problem and the sizing of populations. IEEE Trans. on Evol. Comput. 6(6), 566–579 (2002)
4. Branke, J., Kaußler, T., Schmidt, C., Schmeck, H.: A multi-population approach to dynamic optimization problems. In: Proc. 4th Int. Conf. on Adaptive Computing in Design and Manufacture, pp. 299–308 (2000)
5. Grefenstette, J.J.: Genetic algorithms for changing environments. In: Proc. 2nd Int. Conf. on Parallel Problem Solving from Nature, pp. 137–144 (1992)
6. Lee, S., Soak, S., Kim, K., Park, H., Jeon, M.: Statistical properties analysis of real world tournament selection in genetic algorithms. Applied Intell. 28(2), 195–205 (2008)
7. Mohemmed, A.W., Sahoo, N.C., Geok, T.K.: Solving shortest path problem using particle swarm optimization. Applied Soft Comput. 8(4), 1643–1653 (2008)
8. Oh, S., Ahn, C., Ramakrishna, R.: A genetic-inspired multicast routing optimization algorithm with bandwidth and end-to-end delay constraints. In: Proc. 13th Int. Conf. on Neural Information Processing, pp. 807–816 (2006)
9. Oppacher, F., Wineberg, M.: The shifting balance genetic algorithm: improving the GA in a dynamic environment. In: Proc. Genetic and Evol. Comput. Conf., pp. 504–510 (1999)
10. Parsa, M., Zhu, Q., Garcia-Luna-Aceves, J.: An iterative algorithm for delay-constrained minimum-cost multicasting. IEEE/ACM Trans. on Networking 6(4), 461–474 (1998)
11. Perkins, C.E. (ed.): Ad Hoc Networking. Addison-Wesley, London (2001)
12. Tsutsui, S., Fujimoto, Y., Ghosh, A.: Forking genetic algorithms: GAs with search space division schemes. Evol. Comput. 5(1), 61–80 (1997)
13. Yang, S., Yao, X.: Experimental study on population-based incremental learning algorithms for dynamic optimization problems. Soft Comput. 9(11), 815–834 (2005)
14. Yang, S., Yao, X.: Population-based incremental learning with associative memory for dynamic environments. IEEE Trans. on Evol. Comput. 12(5), 542–561 (2008)
15. Yang, S.: Genetic algorithms with memory- and elitism-based immigrants in dynamic environments. Evol. Comput. 16(3), 385–416 (2008)