Genetic Algorithms with Elitism-based Immigrants for Dynamic Load Balanced Clustering Problem in Mobile Ad Hoc Networks

Hui Cheng
Department of Computer Science and Technology
University of Bedfordshire
Luton LU1 3JU
United Kingdom
Email: hui.cheng@beds.ac.uk

Shengxiang Yang
Department of Information Systems and Computing
Brunel University
Uxbridge, Middlesex UB8 3PH
United Kingdom
Email: shengxiang.yang@brunel.ac.uk

***Abstract*—Clustering can help aggregate the topology information and reduce the size of routing tables in a mobile ad hoc network (MANET). To achieve fairness and even energy consumption, each clusterhead should ideally support the same number of cluster members. Moreover, one of the most important characteristics in MANETs is the topology dynamics, that is, the network topology changes over time due to energy conservation or node mobility. Therefore, for a dynamic and complex system like MANET, an effective clustering algorithm should efficiently adapt to each topology change and produce the new load balanced solution quickly. The maintenance of the cluster structure should be as stable as possible to reduce overhead. It requires that the new solution should try to keep most of the good parts in the previous solution. In this paper, we propose to use elitism-based immigrants genetic algorithm (EIGA) to solve the dynamic load balanced clustering problem in MANETs. Each individual represents a feasible clustering structure and its fitness is evaluated based on the load balance metric. Immigrants are introduced to help the population to handle the topology dynamics and produce new and closely related solutions. The experimental results show that EIGA can quickly adapt to the environmental changes (i.e., the network topology change) and produce high-quality solutions after each change.**

***Index Terms*—Mobile ad hoc networks, elitism-based immigrants, dynamic clustering**

I. Introduction

A mobile ad hoc network (MANET) [8], [11], [7] is a self-organizing wireless local area network without infrastructure and central administration. It has the advantages of low cost, plug-and-play convenience, and flexibility. Just like the Internet, the flat network infrastructure of MANETs encounters the scalability problem when the network size increases. Scalability is more challenging in MANETs due to node mobility. Therefore, efficient network management is extremely important. Analogous to the IP subnet concept, a MANET can also be organized into a hierarchical architecture by dividing nodes into clusters. Each cluster maintains and aggregates the information of the nodes within it. Each cluster can thus be seen as a logical node at the cluster level. The network layer only needs to maintain and manage the information of these logical nodes. Clearly, the control overhead will be reduced with the aid of clustering.

A clustering algorithm [19] is to find a feasible interconnected set of clusters covering the entire set of nodes in a MANET. At any instant, one mobile node can only belong to one cluster. A cluster may have a clusterhead or not. Since the recruiting of clusterheads brings the advantage of easy management, most of the prior research work is on clustering with clusterhead. In this paper, our algorithm also generates the clusters with clusterheads assigned. Furthermore, clustering must be associated with at least one metric such as node ID, node degree, and energy (battery power). The metric is specified based on the application requirements. For example, in the highest degree heuristic [4], the node with the maximum number of neighbors (highest degree) is chosen as the clusterhead.

In this paper, we consider the load balance as the clustering metric since it is an important issue. The load balance means that every clusterhead should ideally support the same number of clustermembers. It can guarantee the fairness for all the clusterheads in term of the load. Moreover, the load balanced clustering can help prolong the lifetime of the clustering structure since each clusterhead will evenly consume the battery energy. It has been proved that finding an optimal set of clusterheads with one or more clustering metrics is NP-hard [2]. Conventional search techniques, such as hill climbing [9], are often incapable of optimizing non-linear multimodal functions. In such a case, a random search method might be required. Evolutionary algorithms (EAs), e.g., genetic algorithms (GAs), are well-known guided random search and optimization techniques. They are based on the basic principles of evolution: survival of the fittest and inheritance. Generally, EAs are applied to find approximate optimal solutions with respect to a fitness function for NP-hard problems.

However, since we consider the load balanced clustering problem in a continuously changing network, it turns out to be one of the dynamic optimization problems (DOPs). In recent years, studying EAs for DOPs has attracted a growing interest due to its importance in EA's real world applications [16].

978-1-4244-9929-8/11/$26.00 ©2011 IEEE

Due to continuous topology changes, MANET is a typical dynamic network. Therefore, a few traditional static network optimization problems, e.g., routing and multicast, have become dynamic network optimization problems in MANETs. These problems urgently call for specialized algorithms which are able to provide high-quality solutions to them. In previous works, we have successfully applied a few dynamic genetic algorithms to solve a couple of representative DOPs in MANETs, i.e., dynamic shortest path routing problem [18] and dynamic multicast problem [3]. In this paper, we investigate the dynamic clustering problem in MANETs.

The simplest way of addressing DOPs is to restart EAs from scratch whenever an environmental 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 re-introduce diversity during the run of EAs, i.e., the immigrants schemes [14], [10], [17]. The random immigrants scheme is the simplest one where randomly generated new individuals are introduced into the population. The elitism-based immigrants scheme [15] is a representative one among immigrants schemes for EAs in dynamic environments. In the scheme, the elite from previous generation is used as the base to create immigrants via mutation to replace the worst individuals in the current population. This way, the introduced immigrants are more adapted to the changing environment.

In this paper, we implement and apply the GA with elitism-based immigrants to solve the dynamic load balanced clustering problem. First, we design the specific genetic algorithm for the dynamic clustering problem. Then, at each generation, a certain number of elitism-based immigrants are generated and added into the population to maintain the diversity. Once the topology is changed, the new immigrants can help guide the search of good solutions in the new environment. For comparison purposes, we also implement a GA with random immigrants (RIGA), standard GA (SGA), and the Restart GA (RGA). By simulation experiments, we evaluate their performance on the dynamic load balanced clustering problem. The results show that the EIGA significantly outperforms the other three GA methods. It is verified that EIGA works really well in the dynamic real-world networks.

II. Related Work

A typical cluster structure in a MANET is shown in Fig. 1. Within one cluster, mobile nodes may play different roles, such as clusterhead, clustergateway, or clustermember. A clusterhead normally serves as a local coordinator for its cluster, performing intracluster transmission control, data forwarding, and so on. A clustergateway is a non-clusterhead node with inter-cluster links, so it can access neighboring clusters and forward data between clusters. A clustermember is an ordinary node, which is a non-clusterhead node without any inter-cluster links.

The primary step in clustering is the selection of clusterheads. The clusterhead can be the leader node, for example,

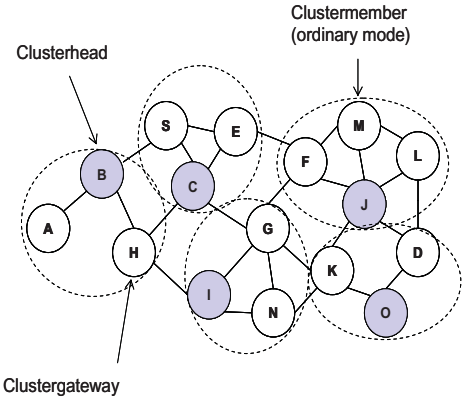


Fig. 1. Illustration of a cluster structure in a MANET.

the node with the maximum power. The selection is based on different criterion derived from specific communication requirements. For one-hop clustering, the cluster structure is determined once the clusterheads are determined. In the following, we formalize the clusterhead selection problem. A MANET is represented as an undirected graph $G(V, E)$, where V represents the set of mobile nodes and E represents the set of links between nodes. Let $N(v)$ be the neighborhood of node v, defined as:

$$N(v) = \bigcup_{v' \in V, v' \neq v} \{v' | dist(v, v') < r\} \ . \tag{1}$$

where r is the transmission range of node v. The generalized procedure for selecting the clusterhead is as follows.

Step 1: From G, select one mobile node v as a clusterhead according to a certain rule.

Step 2: Delete node v and all its neighbors (i.e., all nodes in $N(v)$) from G.

Step 3: Repeat Steps 1 to 2 for the remaining nodes in G until G is empty.

The above three steps generate a set of clusterheads. In Step 1, the rule determines which node is selected as the clusterhead. Different clustering algorithms define different rules, such as the lowest node-ID, the highest node-degree, the least node-weight, etc.

III. Model

In this section, we first present our network model and then formulate the problem of dynamic load balanced clustering. We consider a MANET that operates 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 that fall into the radio transmission range.

The dynamic load balanced clustering (DLBC) problem can be informally described as follows. Initially, given a network of wireless nodes, we wish to find a set of clusterheads from the network and each clusterhead serves the same number of clustermembers. 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 set of clusterheads after each topology change.

More formally, consider a MANET $G(V, E)$. The DLBC problem is to find a series of clusterhead sets $\{CH_i | i \in \{0, 1, ...\}\}$ over a series of graphs $\{G_i | i \in \{0, 1, ...\}\}$. Assume that the clusterhead set $CH_i = \{C_1, C_2, ..., C_m\}$, the clusterhead degree of C_j (i.e., the number of clustermembers served by clusterhead C_j) is d_j, and the average number of clustermembers served by each clusterhead is $\overline{d_{CH_i}}$. We aim to minimize the standard deviation of $\{d_j | j \in \{1, 2, ..., m\}\}$ as shown in Eq. (2).

$$\sigma_{CH_i} = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(d_j - \overline{d_{CH_i}})^2}\ . \qquad (2)$$

IV. Design of the GA for the DLBC Problem

This section describes the design of the GA for the DLBC problem. The GA operations consist of several key components: genetic representation, population initialization, fitness function, selection scheme, crossover, and mutation.

A. Genetic Representation

Each solution produced by our algorithm stands for a set of clusterheads, which are selected from all the nodes in the network. Hence, a random permutation of node IDs will result in a random set of clusterheads. In this algorithm, we use random permutation of node IDs to represent a chromosome. It is important to guarantee that there is no duplicate node ID in each chromosome. Each node ID in the chromosome is called a gene. For example, in a MANET consisting of eight nodes with IDs ranging from 1 to 8, a random permutation (4 3 8 7 1 6 2 5) represents a chromosome.

We need to derive a set of clusterheads from each chromosome. Let us explain this method with an example. Assume that the chromosome is (4 3 8 7 1 6 2 5). First, we add the first gene 4 into the clusterhead set. Then, all the 1-hop neighbors of node 4 are no longer allowed to be clusterheads. Assume that the neighbors of node 4 are nodes 5 and 6. We continue to check the next gene and add node 3 into the clusterhead set. The 1-hop neighbors of node 3 are nodes 1 and 2. So nodes 1, 2, 5, and 6 are not considered as clusterheads any more. Then, we add node 8 into the clusterhead set. The available neighbors of node 8 are node 7. Hence, node 7 is also forbidden to be a clusterhead. Until now, all the nodes have been checked and a clusterhead set {4, 3, 8} is generated. Table I illustrates the procedure of clusterhead selection.

TABLE I
The Procedure for deriving a set of clusterheads from a chromosome

Step	Candidate genes for clusterheads	Set of clusterheads
1	(43871625)	$\{\}$
2	$(-3871-2-)$	$\{4\}$
3	$(--87----)$	$\{4, 3\}$
4	$(--------)$	$\{4, 3, 8\}$

B. Population Initialization

In the GA, each chromosome corresponds to a potential solution. The initial population Q is composed of a certain number, denoted as q, of chromosomes. To explore the genetic diversity, in our algorithm, for each chromosome, the corresponding permutation of node IDs is randomly generated. The initial population is generated as follows.

Step 1: Start ($k = 0$).
Step 2: Generate chromosome Chr_k: create a random permutation of all the node IDs and derive the corresponding clusterhead set CH_k;
Step 3: $k = k + 1$. If $k < q$, go to *Step 2*, otherwise, stop.

Thus, the initial population $Q = \{Chr_0, Chr_1, ..., Chr_{q-1}\}$ is obtained.

C. Fitness Function

Given a solution, we should accurately evaluate its quality (i.e., the fitness value), which is determined by the fitness function. In our algorithm, we aim to find the clusterhead set which can build up the load balanced cluster structure, that is, each clusterhead has the same clusterhead degree (i.e., serving the same number of clustermembers). Our primary criterion of solution quality is the standard deviation of the clusterhead degrees. Therefore, among a set of candidate solutions, we choose the one with the least standard deviation. The fitness value of chromosome Chr_i (representing the clusterhead CH_i), denoted as $F(Chr_i)$, is given by:

$$F(Chr_i) = (\sigma_{CH_i})^{-1} = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(d_j - \overline{d_{CH_i}})^2}^{-1}\ . \qquad (3)$$

D. 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 [6] as it is simple and efficient. The tournament size is 2.

E. Crossover and Mutation

Crossover and mutation are two important genetic operators. Crossover helps generate two offspring chromosomes from two parent chromosomes. All the genes in each offspring chromosome are inherited from different parts of the two parent chromosomes. In this algorithm, we employ the well-known X-Order1 method [12]. Mutation generates an offspring

chromosome from only one parent chromosome by changing the values of some genes. We employ the simple and efficient gene swapping method for mutation.

V. Elitism-based Immigrants GA

In stationary environments, convergence at a proper pace is really 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 [5], [1]. 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 [20]. 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 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 [15], is proposed for GAs to address DOPs.

The pseudo-code for the EIGA is shown below. Within EIGA, for each generation t, after the normal genetic operations (i.e., selection and recombination), the elite $E(t-1)$ from previous generation is used as the base to create immigrants. From $E(t-1)$, a set of $r_{ei} \times q$ individuals are iteratively generated by mutating $E(t-1)$ with a probability p_m^i, where q 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 previous population to guide the immigrants toward the current environment, which is expected to improve GA's performance in dynamic environments.

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

    // perform elitism-based immigration
    denote the elite in $Q(t-1)$ by $E(t-1)$
    generate $r_{ei} \times n$ immigrants by mutating $E(t-1)$ with p_m^i
    evaluate these elitism-based immigrants

    replace the worst individuals in $Q'(t)$ with the generated immigrants
    $Q(t+1) := Q'(t)$
  until the termination condition is met // e.g., $t > t_{max}$
end

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

VI. Experimental Study

We implement EIGA, RIGA, SGA, and RGA for the DLBC problem. By simulation experiments, we evaluate their performance in a continuously changing MANET.

A. Experimental Design

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 they are 1-hop neighbors. 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.

All the algorithms start from the initial network topology. Then, after a certain number (saying, R) of generations (i.e., the change interval), a certain number (saying, 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. By this means, we create a series of network topologies corresponding to the continuous network changes. Furthermore, these adjacent topologies are highly related since each time the changes affect only part of the nodes. We can see that R and M determine the change frequency and severity, respectively. The larger the value of R, the slower the changes. The larger the value of M, the more severe the changes.

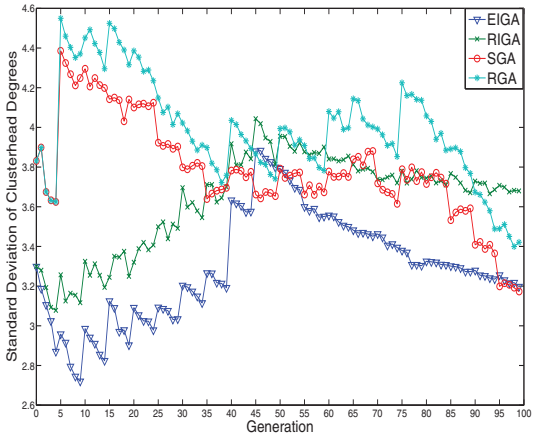


Fig. 2. Comparison of EIGA, RIGA, SGA, and RGA when *M* is set to 2 and *R* is set to 5.

As described in Section IV.D, the GA adopts pair-wise tournament selection without replacement. In all the experiments, the mutation probability is set to 0.1. For the random immigrants scheme, r_i is set to 0.2. For the elitism-based immigrants scheme, r_{ei} is set to 0.2 and p_m^i is set to 0.8. In addition, we set the number of changes to 19 and therefore the algorithms will work over 20 different but highly-related network topologies (the initial topology plus the 19 changed topologies).

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

$$(1 + r_i) * q = 120, \tag{4}$$

where q is the whole population size, which was set to 100 in the experiments. Hence, we have $q = 120$ for SGA and RGA, and $q = 100$ for RIGA and EIGA.

B. Experimental Results and Analysis

At each generation, for each algorithm, we select the best individual from the current population and output the standard deviation of the clusterhead degrees calculated from it. We repeat each experiment 10 times and get the average values of the best solutions at each generation. We vary M from 2 to 4 to see the effect of change severity on the algorithm performance. We also set R to 5, 10, and 15, respectively, to see the effect of the change frequency on the algorithm performance.

Fig. 2 is the comparison results when the change severity parameter M is set to 2 and the change interval R is 5. In Fig. 3, we change R to 10 and in Fig. 4, we change R to 15. Therefore, Fig. 2 shows a rapidly changing environment, Fig. 3 shows a relatively slowly changing environment, and Fig. 4 shows a much slower changing environment than the two others. In Fig. 5, the change interval R is 10 while the change severity parameter M is increased to 4.

From Fig. 2, we can see that the first nine changes affect the algorithms much more significantly than all the next changes. This is due to that in the random dynamic topology

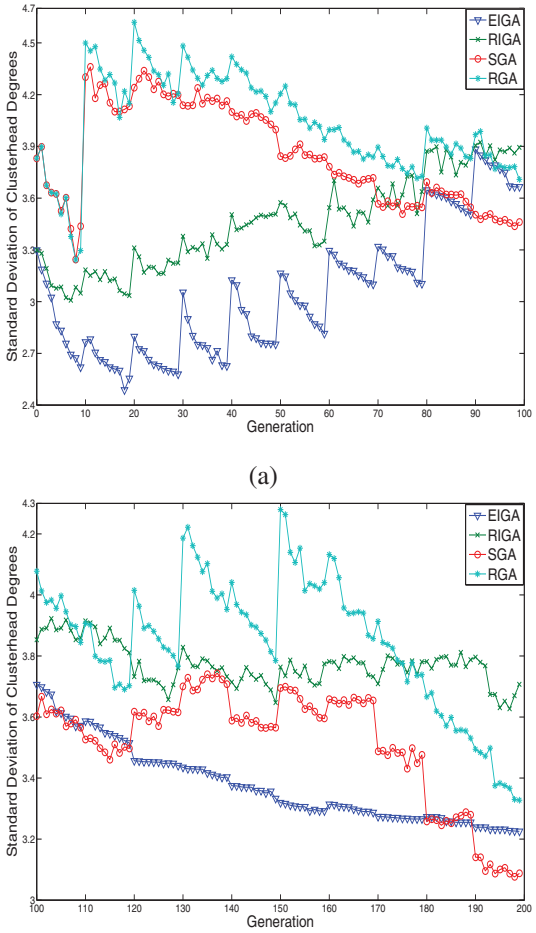
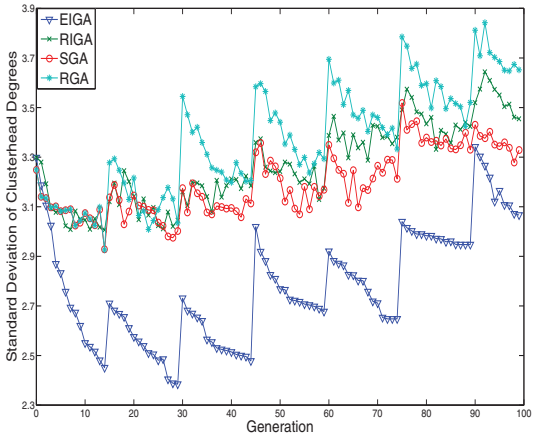


Fig. 3. Comparison of EIGA, RIGA, SGA, and RGA when *M* is set to 2 and *R* is set to 10: (a) generation 0 to 99; (b) generation 100 to 199.

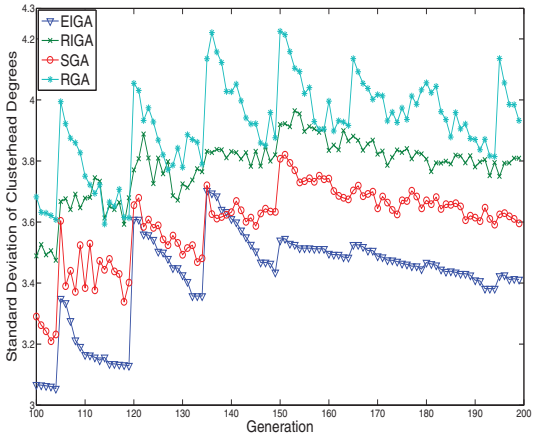
generation, the first nine changes all schedule the nodes to sleep. Therefore, each of these changes will result in re-search the clusterheads. However, in the later changes, some nodes are scheduled again to wake up and work. These changes affect the current population in a less significant way. There is a high probability that the corresponding good solutions already exist in the population. Therefore, EIGA can handle them well and no significant impact to the population is observed. From Fig. 3 and Fig. 4, we can see the similar behaviors of EIGA.

In Fig. 2, the standard deviation of clusterhead degrees achieved by EIGA ranges from 2.7 to 3.9. In Fig. 3, the corresponding value ranges from 2.45 to 3.95. In Fig. 4, the value ranges from 2.35 to 3.7. We can see that in a slowly changing environment, EIGA shows a better performance than in a rapidly changing environment since it has more time to search good solutions before the next change occurs.

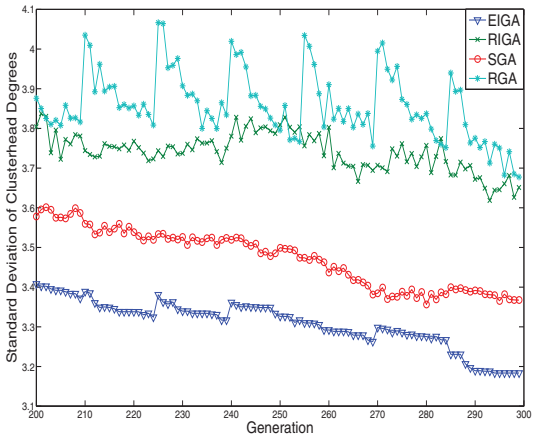
From the above three figures, another interesting point observed is that when the changes have significant impact to the population, RIGA performs better than SGA. While when the changes show slight impact to the population, SGA performs better than RIGA. In Fig. 5, the change severity



(a)



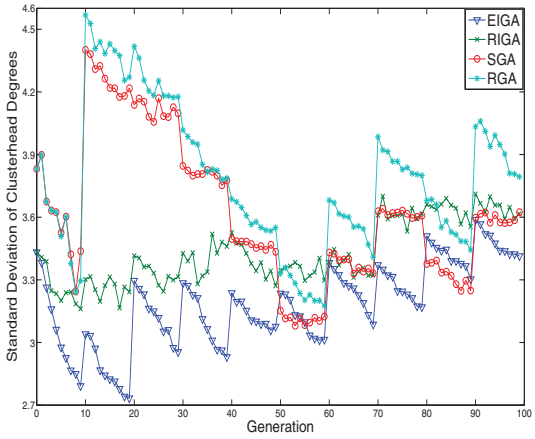
(b)



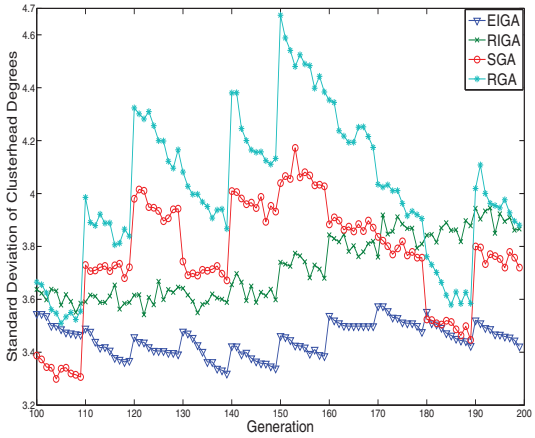
(c)

Fig. 4. Comparison of EIGA, RIGA, SGA, and RGA when M is set to 2 and R is set to 15: (a) generation 0 to 99; (b) generation 100 to 199; (c) generation 200 to 299.

parameter M is set to 4, which means that each time four nodes will be scheduled to sleep or wake up. More nodes involved in the change will bring a higher change severity. We can see that RIGA performs better when M is 4 than when M is 2. This shows that random immigrants can help preserve the diversity of the population and thereby increase the capability of the population in handling the dynamic environment. Furthermore, in the environment with a higher change severity, the random immigrants scheme is more powerful. However, if the environment is relatively peaceful, random immigrants adversely affect the population.



(a)



(b)

Fig. 5. Comparison of EIGA, RIGA, SGA, and RGA when M is set to 4 and R is set to 10: (a) generation 0 to 99; (b) generation 100 to 199.

In all the figures, EIGA shows the best performance. The reason is that EIGA exploits both the population diversity brought by the immigrants scheme and the advantages of the elitism scheme to enhance its search capability. On the other hand, we can see that RGA always exhibits the worst performance even when the changes have trivial impacts on the current population. The reason is that RGA does not exploit any useful information in the old environment and that the frequent restart sacrifices its evolving capability. Therefore, for a dynamic optimization problem where the problem dynamics is at a reasonable level, to restart the whole population of GA is not a good choice.

VII. Conclusions

The clustering problem has been extensively addressed in MANETs. However, previous works do not pay much attention to the continuous topology changes, which are actually the inherent characteristics of MANETs. Intuitively, it is much more challenging to deal with the dynamic clustering problem in a continuously changing MANET than to solve the quasi-static one in a quasi-static infrastructure where only local small modification will be introduced after clustering.

In recent years, there has been a growing interest in studying GAs for dynamic optimization problems. Among approaches developed for GAs to deal with DOPs, immigrants schemes for GAs on DOPs aim at maintaining the diversity of the population throughout the run via introducing new individuals into the current population. In this paper, we apply the elitism-based immigrants scheme for the GA to solve the dynamic load balanced clustering problem in a large scale MANET. We well design the GA components for the clustering problem and the elitism-based immigrants scheme. To encourage load balance, we use the standard deviation of clusterhead degrees to evaluate the performance of the clustering results. Simulation experiments show that EIGA is a powerful technique for solving the dynamic load balanced clustering problem and that the elitism-based immigrants scheme outperforms the random immigrants scheme, standard GA, and restart scheme for the tested dynamic clustering problems.

Acknowledgment

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of UK under Grant EP/E060722/1 and Grant EP/E060722/2.

References

[1] H. G. Cobb and J. J. Grefenstette, "Genetic algorithms for tracking changing environments," *Proc. 5th Int. Conf. Genetic Algorithms*, Urbana-Champaign, IL, Jun. 1993, pp. 523–530.

[2] M. Chatterjee, S. K. Das and D. Turgut, "WCA: a weighted clustering algorithm for mobile ad hoc networks," *Cluster Comput.*, vol. 5, no. 2, pp. 193–204, Apr. 2002.

[3] H. Cheng and S. Yang, "Genetic algorithms with immigrants schemes for dynamic multicast problems in mobile ad hoc networks," *Eng. Appl. AI*, vol. 23, no. 5, pp. 806–819, Aug. 2010.

[4] M. Gerla and J. T.-C. Tsai, "Multicluster, mobile, multimedia radio network," *Wireless Netw.*, vol. 1, no. 3, pp. 255–265, Sep. 1995.

[5] J. J. Grefenstette, "Genetic algorithms for changing environments," *Proc. 2nd Int. Conf. Parallel Problem Solving from Nature*, 1992, pp. 137–144.

[6] S. Lee, S. Soak, K. Kim, H. Park and M. Jeon, "Statistical properties analysis of real world tournament selection in genetic algorithms," *Appl. Intell.*, vol. 28, no. 2, pp. 195–205, Apr. 2008.

[7] C. Siva Ram Murthy and B. S. Manoj, *Ad Hoc Wireless Networks: Architectures and Protocols*, Prentice Hall PTR, 2004.

[8] C. E. Perkins, Editors, *Ad Hoc Networking*, London: Addison-Wesley, 2001.

[9] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, NJ: Prentice Hall, 2003.

[10] R. Tinos and S. Yang, "A self-organizing random immigrants genetic algorithm for dynamic optimization problems," *Genetic Programming Evolvable Mach.*, vol. 8, no. 3, pp. 255–286, Sept. 2007.

[11] C.-K. Toh, *Ad Hoc Mobile Wireless Networks: Protocols and Systems*, Prentice Hall PTR, 2002.

[12] D. Turgut, S. K. Das, R. Elmasri and B. Turgut, "Optimizing clustering algorithm in mobile ad hoc networks using genetic algorithmic approach," *Proc. IEEE Global Telecommun. Conf.*, Taipei, Taiwan, 2002, pp. 62–66.

[13] S. Yang and X. Yao, "Experimental study on population-based incremental learning algorithms for dynamic optimization problems," *Soft Comput.*, vol. 9, no. 11, pp. 815–834, Nov. 2005.

[14] S. Yang and R. Tinos, "A hybrid immigrants scheme for genetic algorithms in dynamic environments," *Int. J. Autom. Comput.*, vol. 4, no. 3, pp. 243–254, Jul. 2007.

[15] S. Yang, "Genetic algorithms with elitism-based immigrants for changing optimization problems," *Proc. EvoWorkshops 2007: Appl. Evol. Comput.*, 2007, pp. 627–636.

[16] S. Yang and X. Yao, "Population-based incremental learning with associative memory for dynamic environments," *IEEE Trans. Evol. Comput.*, vol. 12, no. 5, pp. 542–561, Oct. 2008.

[17] S. Yang, "Genetic algorithms with memory- and elitism-based immigrants in dynamic environments," *Evol. Comput.*, vol. 16, no. 3, pp. 385–416, Sept. 2008.

[18] S. Yang, H. Cheng and F. Wang, "Genetic algorithms with immigrants and memory schemes for dynamic shortest path routing problems in mobile ad hoc networks," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 40, no. 1, pp. 52–63, Jan. 2010.

[19] Y. Yu and H. J. Chong, "A survey of clustering schemes for mobile ad hoc networks," *IEEE Commun. Surveys Tuts.*, vol. 7, no. 1, pp. 32–48, Mar. 2005.

[20] F. Vavak and T. C. Fogarty, "A comparative study of steady state and generational genetic algorithms for use in nonstationary environments," *Proc. AISB Workshop Evol. Comput.*, Brighton, UK, Apr. 1996, pp. 297–304.