Genetic Algorithm with Random and Memory Immigrant Strategies for Solving Dynamic Load Balanced Clustering Problem in Wireless Sensor Networks

Mohaideen Pitchai
Department of Computer Science and Engineering,
National Engineering College, India
kmpcse@nec.edu.in

Abstract: *In Wireless Sensor Networks (WSNs), clustering is an effective method to distribute the load equally among all the nodes as compared to flat network architecture. Due to the dynamic nature of the network, the clustering process can be viewed as a dynamic optimization problem and the conventional computational intelligence techniques are not enough to solve these problems. The Dynamic Genetic Algorithm (DGA) addresses these problems with the help of searching optimal solutions in new environments. Therefore the dynamic load-balanced clustering process is modeled using the basic components of standard genetic algorithm and then the model is enhanced is using immigrants and memory-based schemes to elect suitable cluster heads. The metrics nodes' residual energy level, node centrality, and mobility speed of the nodes are considered to elect the load-balanced cluster heads and the optimal number of cluster members are assigned to each cluster head using the proposed DGA schemes such as Random Immigrants Genetic Approach (RIGA), Memory Immigrants Genetic Approach (MIGA), and Memory and Random Immigrants Genetic Approach (MRIGA). The simulation results show that the proposed DGA scheme MRIGA outperforms well as compared with RIGA and MIGA in terms of various performance metrics such as the number of nodes alive, residual energy level, packet delivery ratio, end-to-end delay, and overhead for the formation of clusters.*

Keywords: *Wireless sensor networks, genetic algorithm, load balanced clustering, random immigrants, memory immigrants.*

Received November 24, 2021; accepted December 15, 2022
https://doi.org/10.34028/iajit/20/4/3

1. Introduction

The sensors in Wireless Sensor Network (WSN) are powered by batteries and recharging these batteries is very difficult in large-scale deployments. It necessitates the research community to design and implement energy-efficient approaches to prolong the lifetime of the network. Clustering is an effective method to distribute the load equally among the sensor nodes which minimizes the energy consumption of nodes as compared to the flat network architecture [9]. The network is divided into different groups or clusters based on suitable network parameters where each cluster has a cluster leader known as cluster head, and all other nodes are cluster member nodes. The number of cluster member nodes assigned to each cluster is an issue to balance the load of the cluster head [2]. Moreover, finding the optimal number of cluster member nodes in a cluster with one or more metrics is an NP-hard problem [5].

A Genetic Algorithm (GA) is an appropriate optimization technique to find optimal clustering by considering various network parameters since it uses the biological principles of natural selection, cross over and mutation. Hence GA finds the near-optimal solution for load-balanced clustering. But the clustering problem can be viewed as a Dynamic Optimization Problem (DOP) due to the dynamic nature of the network and the solutions proposed so far for clustering are not adaptable to this dynamic environment. Various genetic-based clustering algorithms are proposed and those approaches fail to maintain the stability of the network due to frequent movements of nodes in the network [4].

GA addresses the DOP through dynamic genetic operators such as random immigrants to maintain diversity [14] and memory immigrants to use the knowledge obtained from old environments into the new environments [17]. Thus various dynamic GA-based schemes are proposed such as random immigrants, memory immigrants, and memory-random immigrants to find the load balanced cluster heads in dynamic environments of the sensor network. The novelty and main contributions of this paper can be summarized as follows:

1. The basic components of standard GA for the dynamic load balanced clustering problem are

designed.
2. The immigrant and memory schemes are incorporated in the standard GA to enhance the searching process for finding suitable cluster heads in dynamic environments of the sensor network.
3. The hybrid scheme of memory-random immigrants is also applied to optimize the load-balanced clustering process, which uses nodes' residual energy level, node centrality, and mobility speed of the nodes.

The rest of this paper is organized as follows. Section 2 surveys some related works on the Dynamic Load Balanced Clustering Problem (DLBCP) in sensor networks. The proposed dynamic GA-based scheme for DLBCP is explained in detail in section 3. In section 4, the simulation results are presented. Finally, section 5 gives a conclusion.

2. Related Works

This section describes various GA-based strategies developed so far for solving dynamic clustering and related problems in sensor networks. In the last few years, immigrant schemes of GA are employed to adapt to the dynamic environments of the sensor network. To achieve this, an energy-efficient clustering scheme is needed to choose the load-balanced cluster head node which is adaptable to the topology dynamics of the network.

Yuan *et al*., [18] a method named 'Genetic Algorithm based Self-Organizing Network Clustering (GASONeC)' is proposed which optimizes the clusters in the network dynamically. The parameters such as the residual energy, the distance to the base station, the expected energy consumption, and the number of nodes in the locality are considered to form a dynamic and optimal network structure. Here more clusters are formed when the base station is placed far from the field of the network. In [6], a two-stage genetic algorithm is proposed which is used to select the optimal set of clusters in wireless sensor networks. The optimal cluster heads are identified in the first stage and assign appropriate cluster members to these cluster heads in the second stage. To achieve load balancing, the optimized intra-cluster distance is considered which minimizes the energy consumption of the network.

Tianshu *et al*., [13] a genetic algorithm-based energy-efficient clustering and the routing mechanism are presented for wireless sensor networks. This approach improves the search efficiency by adding an optimal solution obtained in the previous round to the initial population of the current round. They constructed the fitness function by considering the load balancing and the total energy consumption of the network. Also, the energy consumption among the nodes is balanced. In [3], used an elitism-based immigrant scheme of genetic algorithm is used to solve the dynamic load balanced clustering problem in ad-hoc networks. Here the fitness value is evaluated based on the load metric and the immigrant helps to handle the topology dynamics which produces new and near closer solutions.

Kheireddine *et al*., [8] a dynamic centralized GA-based clustering approach is proposed to optimize the clustering (cluster heads and cluster members) to reduce the energy consumption of nodes. They presented a novel clustering algorithm, denoted as Genetic Centralized Dynamic Clustering (GCDC), which uses a GA to optimize the number and the corresponding locations of CHs. In [11], a dynamic clustering-based approach is developed where the relay nodes in a cluster select the most suitable sensor node as a cluster head for that cluster. The fitness function used here chooses the cluster head node based on the metrics like distance to the base station, the total coverage area of the cluster, energy consumption to send the message to the base station, and the number of transmissions. The availability and the total efficiency of the network are improved using this GA-based approach.

Sirbu and Alecsandrescu [12], user-specific genetic operators are designed for clustering to improve the energy efficiency of ad-hoc sensor networks. Their proposed algorithm achieves convergence through fine-tuning the genetic algorithm parameters. This method produces near-optimal solutions in terms of the convergence speed rate and minimizes the distance for communication to nearby cluster heads. In [10], a weighted clustering algorithm with the use of a genetic algorithm is developed called 'Genetic Algorithm Based Optimized Clustering (GABOC)'. Here the cluster heads are chosen based on the fitness function parameters such as battery power, degree of distance difference, and degree of mobility. The performance of this algorithm is better in terms of maintaining connectivity among the cluster heads as compared to other deterministic algorithms. In [1], a Distributed Efficient Multi hop Clustering (DEMC) protocol for mobile wireless sensor networks is presented. DEMC is distributed, works well with mobile nodes, and has a recovery mechanism that is used to reduce the packet loss during inter cluster communication. The recovery mechanism also improves the connectivity between cluster heads during inter cluster communication. On average, each node sends less than one message during clustering, and does not rely on periodic hello messages.

Yang *et al*., [16], a genetic algorithm with immigrants and memory schemes is used to solve dynamic shortest path routing problems in ad-hoc networks. The results show that the immigrants and memory-based genetic algorithms can easily and quickly adaptable to dynamic environments and generates good quality solutions for these environments. That scheme maintains the diversity in

the current population and stores useful information through random and memory immigrants. In [15], energy dynamic distribution and the optimization algorithm is proposed which uses the genetic algorithm with elitism-based immigrants approach for optimizing the poor individuals when the nodes' position is changed for maritime search and rescue applications. In [7], the dynamic shortest path problem using elitism-based immigrants genetic algorithm is solved in ad-hoc networks. Here the immigrants are generated and added in each generation which helps to find good solutions in the changing environment.

3. Proposed Work

The proposed work focuses on finding optimal cluster heads, which adapt the network's load balancing using various genetic approach operations. The nodes are represented as genes, and new populations are generated using permutations of chromosomes. The cluster heads are selected based on residual energy level, node centrality, and weight value. The fitness values of nodes are calculated using those parameters. The selection operation is performed by choosing the chromosomes with higher fitness value since those chromosomes have a higher probability of mating. The roulette wheel selection method is applied for selection operation. Then the genetic operations crossover and mutation are used.

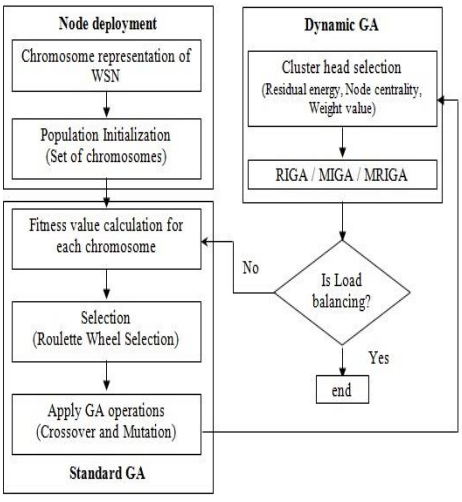

Figure 1. Proposed load balanced clustering in WSN.

The crossover selects the best individual from the current population, which produces the new population. The mutation alters one or more gene values in a chromosome in the current population from its initial state, and if needed, the new gene values are added to the gene pool. The best solution may be obtained from the new gene values in the current population. The best solution produces the new child chromosomes whose corresponding cluster headsets act as the finalized cluster heads. If the network environment changes, then the clustering structure also changes. Hence the proposed method uses various dynamic genetic approaches such as random immigrants, memory immigrants, and memory-random immigrants. These approaches replace the worst individuals in the chromosome with the best individuals who are generated using those immigrants. Thus the load-balanced cluster head selection process is achieved. Figure 1 shows the proposed dynamic load-balanced clustering process in WSNs

3.1. Chromosome Representation

The nodes in the network are assumed as genes. The number of nodes in the WSN is taken as the size of the population. Assume that the size of the population is 'm'. The random selection of nodes is considered a chromosome. The size of the chromosome is 'r'. Thus the selection of chromosomes from the population is considered as permutation process ($_mP_r$). Each node in the network has been identified by a node identifier (n_ID). For example, in a WSN consisting of nine nodes with IDs ranging from 1 to 9, a random permutation (3 4 7 8 1 6 9 2 5) represents a chromosome. Here there is a guarantee that no duplicate ID is generated in each chromosome.

3.2. Population Initialization

The initial population P_{init} is composed of 'q' number of chromosomes since each chromosome corresponds to a potential solution. The cluster heads are generated randomly from each chromosome to explore generic diversity. Thus the initial population is denoted as

$$P_{init} = \{chr_1, chr_2, \ldots\ldots\ldots\ldots chr_q\} \quad (1)$$

The corresponding cluster head set ch_i ($1 \le i \le q$) of each chromosome chr_i ($1 \le i \le q$) is derived.

3.3. Fitness Function

The quality of the solution is determined accurately using the fitness function. The objective of the proposed algorithm is to find the cluster head. The criteria are to select the cluster head with maximum residual energy level and node centrality. Among a set of cluster head candidates, choose the one which satisfies the criterion. Through this way of selecting cluster heads, the load is balanced among the nodes in the network. The fitness value of each chromosome (chr_i) is given by

$$\mathrm{Fit}(chr_i) = (\alpha\alpha_i + \beta D_i + \gamma W_i)^{-1} \quad (2)$$

Here α, β and γ are weight factors and $\alpha+\beta+\gamma=1$. R_i is the residual energy level of the i^{th} node. D_i is the distance between the central location of the cluster and the location of the i^{th} node. W_i is the weight value of the i^{th} node. The weight value (W_i) includes the mobility speed of the node (M_i) and the density of the neighbors (N_i) around it. The weight value of the i^{th}

node is calculated using the following equation

$$W_i = w_1 M_i + w_2 N_i \quad (3)$$

Here w_1 and w_2 are weight factors and $w_1+w_2=1$. The mobility speed of a node is calculated using Equation (4). Here T denotes the time interval of the rotation of nodes as cluster heads. Also, (x_t,y_t) and (x_{t+1},y_{t+1}) are the coordinates of node i at time t and t+1 respectively. The density of a node is equal to the number of neighbor nodes. The neighborhood of a node is the set of nodes within its transmission range. After evaluating the fitness function of chromosomes, the optimal cluster heads are identified.

$$M_i = \frac{\sum_{t=0}^{T} \sqrt{(x_{t+1} - x_t)^2 + (y_{t+1} - y_t)^2}}{T} \quad (4)$$

3.4. Selection Scheme

The proposed algorithm uses the Roulette Wheel Selection (RWS) method to improve the population's quality by selecting high-quality chromosomes to the next successive generations. This method also ensures quality by choosing high fitness value chromosomes, which have a higher probability of selecting the mating process. The sum of all individuals' fitness values in the chromosomes is considered the circumference of the roulette wheel. A fixed point on the roulette wheel is chosen, and then the wheel is rotated. The region of the wheel which comes on the fixed point is taken as a parent. This process is repeated to obtain a second parent. Thus the two parents are chosen randomly using the RWS method to produce their offspring. The same chromosome has not been selected as a second parent in this method.

3.5. Crossover and Mutation

These operations are used to generate new populations from the current population. The crossover generates children's chromosomes from two-parent chromosomes. The mutation operator takes the population generated by crossover operation and generates children's chromosomes through the gene swapping method. The proposed scheme uses a single-point crossover. After selecting the parent chromosomes, the genes in the chromosomes are ordered in descending order based on the residual energy level of the nodes. Then, choose the swath of consecutive genes randomly from the two-parent chromosomes. The first parent chromosome drops the swath down into the first child chromosome, and the genes from the second chromosome are filled in the remaining area of the first child chromosome. The left side and right side of the second chromosome's swath are filled in the left side and right side of the child 1 chromosome. Also, this process is used for generating child 2 chromosomes. Figure 2 shows the illustration of the crossover process.

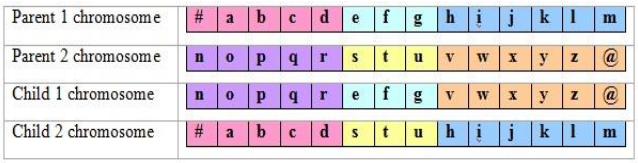

Figure 2. Illustration of crossover.

The mutation operation is performed using the gene swapping method, which generates a child's chromosome from a parent chromosome. It allows changing the values of one or more genes in the parent chromosome. Also, it randomly selects two different positions and interchanges the genes of those positions. Thus the generated new genes will give better solutions. Figure 3 shows the illustration of the mutation process. The generated child chromosome is finalized as cluster heads for the given network. The dynamic genetic operations will be used to adapt the network environmental conditions due to frequent changes in the network's topology. The dynamic conditions are adapted in genetic operations using various approaches like Random Immigrants Genetic Approach (RIGA), Memory Immigrants Genetic Approach **(**MIGA), and Memory and Random Immigrants Genetic Approach (MRIGA).

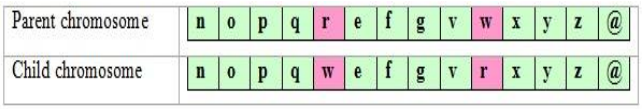

Figure 3. Illustration of mutation.

3.6. Random Immigrants Genetic Approach (RIGA)

Since the cluster head selection problem is considered in a dynamic network environment, the standard genetic algorithm is not well suited. This dynamic optimization problem is solved using a random immigrant genetic approach. The standard genetic operations are combined with immigrant schemes to introduce the diversity level in the current population. It can be achieved by replacing the worst individuals with randomly generated immigrants in the population. The chromosome having the least fitness value in the current generation is replaced with higher fitness value chromosomes in the previous generation. Then crossover and mutation operations will be performed. These processes are repeated until the termination conditions will be met. The termination condition states that until the maximum number of generations is reached. The pseudo-code for RIGA is shown in Algorithm (1).

Algorithm (1) RIGA based Cluster Head selection

 # Initialize m, r, g, Rc, Rm, Ri, Di, Wi
#m is the number of nodes *#r is the*

```
size of chromosome                          #g is the
generation number (g=0)
   #Rc is the crossover rate                #Rm
is the mutation rate
   #Ri is the coefficient of residual energy of the ith node
   #Di is the coefficient of node centrality of the ith node
   #Wi is the coefficient of weight value of the ith node
   for (i = 1 to r)
     {
       chromosome [i]= getChromosome(m,r);
     }
   for (i= 1 to length(chromosome))
     {
       fitness_value=getFitness(chromosome);

       repeat
       {
          fit_chr[i]=Roulette_Wheel_Selection(i,Fit(chi))
         #perform crossover operation on fit_chr[i] with
         Rc crossover rate
         #perform mutation operation on fit_chr[i] with
         Rm mutation rate
         #evaluate the fit_chr[i]
         #perform random immigration
         #generate immigrants randomly
         #replace the worst individuals in fit_chr[i] with
         random immigrants
        #evaluate these immigrants
        chromosome[i]=fit_chr[i]
      } until (g>gmax)
    }
```

3.7. Memory Immigrants Genetic Approach (MIGA)

This approach uses the good solutions of the old population into the new populations when its characteristics match the old population. The best solution is stored in the current population in the extra memory space explicitly or through redundant representation implicitly, and the stored information is reused in the new generations. This approach is more suitable when the environment changes frequently. Since when the old generations will be regenerated exactly, and the corresponding solution of this generation in the memory will move the genetic operations to the reappeared generations.

Algorithm (2)MIGA based Cluster Head selection

```
# Initialize m, r, g, Rc, Rm, Ri, Di, Wi, M
#m is the number of nodes                    #r is the
size of chromosome                        #g is the
generation number (g=0)
#Rc is the crossover rate                    #Rm is the
mutation rate
#Ri is the coefficient of residual energy of the ith node
#Di is the coefficient of node centrality of the ith node
#Wi is the coefficient of weight value of the ith node
#M is memory which is initialized (M[0]) randomly
for (i = 1 to r)
  {
      chromosome [i]= getChromosome(m,r);
  }
for (i= 1 to length(chromosome))
  {
    fitness_value=getFitness(chromosome);
    Fit(ch_i) = (αR_i + βD_i + γW_i)^-1
    repeat
    {
       #evaluate memory M[g]
       #denote the best individual in chromosome[i]
         by B[i]
       #find the memory point C[i] nearest to B[i]
       if (C[i]>B[i])
       {
          B[i]=C[i]
       }
       #/perform standard genetic operations
      fit_chr[i]=Roulette_Wheel_Selection(g,Fit(ch_i))
      #perform crossover operation on fit_chr[i]
      with Rc crossover rate
      #perform mutation operation on fit_chr[i] with
      Rm mutation rate
      #evaluate the fit_chr[i]
      #perform memory-based immigration
      #denote best memory point in M[g] by Bm[i]
      P[i]=mutateBestMemoryPoint(Bm[i])
      #replace the worst individuals in fit_chr[i]
        with the memory based immigrant in P[i]
      #evaluate these memory-based immigrants
      chromosome[i]=fit_chro[i]
      g=g+1
    }
   until (g > gmax)
  }
```

Thus the best solutions are stored and retrieved during the environmental changes. The retrieved solutions will be reactivated for the new generations. Since the memory space is a limited one, the best solutions stored in the memory should be updated periodically to provide space for new good solutions. There are several replacement strategies. The solution with the least fitness value is replaced with the new good solutions in the current generation. The pseudo-code for MIGA is shown in Algorithm (2).

The best individual from chromosome[i] is assigned to B[i]. The random memory point C[i] is equalized to B[i] if the memory point C[i] has higher fitness than the best individual. The best memory point is generated in every generation, and it is mutated to get the individuals that act as immigrants. These immigrants are evaluated and then replace with the worst individuals in the current population. Here the immigrants are distributed over the best memory point.

3.8. Memory and Random Immigrants Genetic Approach (MRIGA)

This algorithm coincides with the memory updating time with the environment change time. When the environmental changes are detected, the memory is re-evaluated in every generation. The best solutions are retrieved from M[g] when the environmental change is detected that assigns to Fit_chro[i].

The key idea behind MRIGA is that memory is used to guide immigrants as a biased one for the current population. In contrast, random is used to select the immigrants in a search space around the best memory point in the current population. These two approaches are combined in the MRIGA approach. The pseudo-code for MRIGA is shown in Algorithm (3).

Algorithm (3) MRIGA based Cluster Head selection

```
# Initialize m, r, g, Rc, Rm, Ri, Di, Wi, M
#m is the number of nodes                    #r is the
size of chromosome                          #g is the
generation number (g=0)
#Rc is the crossover rate                    #Rm is the
mutation rate
#Ri is the coefficient of residual energy of the ith node
#Di is the coefficient of node centrality of the ith node
#Wi is the coefficient of weight value of the ith node
#M is memory which is initialized (M[0]) randomly for (i = 1 to
r)
  {
      chromosome [i]= getChromosome(m,r);
  }

for (i= 1 to length(chromosome))
 {
    Fitness_value=getFitness(chromosomes);
    Fit(ch_i) = (αR_i + βD_i + γW_i)^-1
    repeat
    {
      #evaluate memory M[g]
      if changes_ in_environment_detected
      {
         fit_chr[i]=retrieveBestsolutions(M[g], g)
      }
      #denote best individual in fit_chr[i] by B[i]
      #find the memory point C[i] nearest to B[i]
      if (C[i]>B[i])
      {
        B[i]=C[i]
      }
      //perform standard genetic operations
      fit_chr[i]=Roulette_Wheel_Selection(g,Fit(chi))
      #perform crossover operation on fit_chr[i]
        with Rc crossover rate
      #perform mutation operation on fit_chr[i]
        with Rm mutation rate
      #perform random immigration
      #evaluate the fit_chr[i]
      #generate immigrants randomly
      #evaluate these immigrants
      #replace the worst individuals in fit_chr[i]
      #with random immigrants
      chromosome[i]=fit_chr[i]
      g=g+1
    }
    until(g > gmax)
 }
```

4. Results and Discussion

The performance of the proposed schemes is evaluated using network simulator 2.35 (NS2.35). Table 1 shows the set of relevant simulation parameters. The simulation results analyzed the effectiveness of the three DGA schemes such as random immigrants (RIGA), memory immigrants (MIGA), and memory and random immigrants (MRIGA) to form load-balanced clustering structures in the network. Its performance is compared with the competing scheme, namely, "Genetic algorithm based optimization of clustering (GABOC)". This scheme is mainly designed to enhance the performance of the cluster head selection process using a weighted clustering mechanism. The network's remaining energy, number of nodes alive, packet delivery ratio, end-to-end delay, and the clustering overhead are considered as the performance metrics to evaluate the performance of the proposed GA schemes.

Table 1. Simulation parameters.

Parameter/Scheme	Value/Method
Network area size	100x100m^2
Number of nodes	100, 200
Mobility speed of nodes	[5-20]m/s
Mobility Model	Random way point
Transmission range	1m
Initial energy of each node	3J
Transmission energy	0.6J
Receiving energy	0.2J
Packet size	500 bits
Population size	20
Number of generations	100
Selection method	Roulette–Wheel Selection
Crossover rate (R_c)	0.6
Mutation rate (R_m)	0.03
Ratio of random immigrants	0.2
Ratio of memory immigrants	0.4
Replacement rate	5%

Figure 4 shows the number of nodes alive by increasing the number of rounds for the network of size 200 nodes. The simulation results show that the MRIGA scheme performs better than the other schemes as it has a higher number of nodes alive by 6%, 12%, and 23% against MIGA, RIGA, and GABOC, respectively. In MIGA, the nodes with the least fitness value are replaced with new good solutions. Also, the new solutions are stored in the memory and the same has been updated in a predefined interval for storing new good solutions. In RIGA, the same can be processed through the replacement of the worst individuals by randomly generated immigrants in the current generation. But in MRIGA, uses a randomized approach to select the immigrants in a search space around a memory location in the current population. Thus it replaces the worst candidates for cluster head election with these suitable random immigrants, which leads to increasing the number of nodes in the alive state.

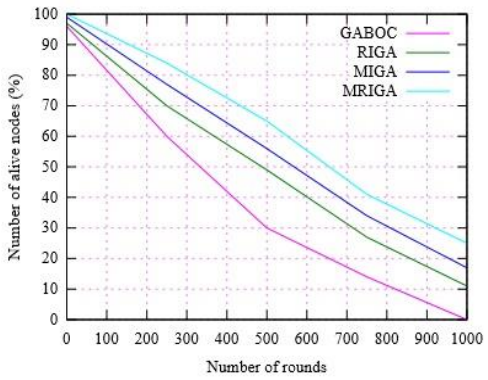

Figure 4. Number of alive nodes.

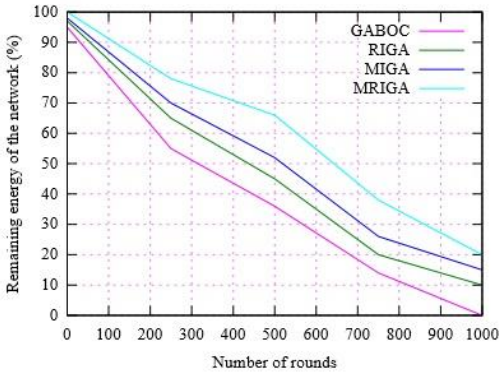

Figure 5. Residual energy level of the network.

Figure 5 shows the network's residual energy for an increasing number of rounds for the network of size 200 nodes. It is observed that the MRIGA has the highest residual energy about 8%, 13%, and 20% as compared with MIGA, RIGA, and GABOC respectively since it stores the nodes having the highest fitness values in the memory. It finds suitable nodes in every generation of the population. In addition, MRIGA adapts the combined approach of RIGA and MIGA which makes the one-hop neighbors of a cluster head become the cluster head in a sequence of iterations. Also, the selection of cluster heads in MRIGA uses energy as a metric to balance the load of clustering formation. Moreover, randomly identifies the nodes which are suitable for cluster heads is selected in a search space around the best memory point in the current population of the network.

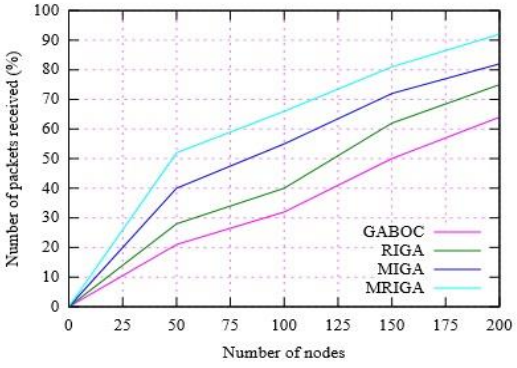

Figure 6. Number of packets received.

Figure 6 shows the number of packets received by varying the number of nodes in the network. As shown in the figure, the number of packets received gets increased when the number of nodes increases. MRIGA achieves a higher number of packets received by 8%, 17%, and 25% as compared with MIGA, RIGA, and GABOC respectively since the GABOC scheme achieves less number of packets received due to its less focus on dynamic environments and multi-metric formation of clusters. MRIGA achieves a higher number of packets delivered because of storing the random best immigrants in the memory to reduce the complexity of the clustering process.

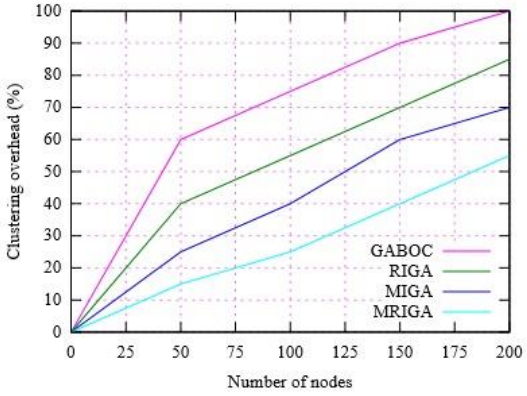

Figure 7. Clustering overhead.

Figure 7 shows the clustering overhead of various schemes while varying the number of nodes. MRIGA creates less clustering overhead by 27% as compared with other schemes since MRIGA stores the current environment in the memory for the future generation of the population. RIGA uses the previous generation as the future generation of the population. It does not have memory storage for storing the current environments. Also, the worst individuals are replaced by suitable best individuals who are generated by random in the previous population. In MIGA, the generations stored in the memory are used as the future generation of the population. Thus MRIGA uses the old environment stored in the memory to reduce the overhead of the clustering process.

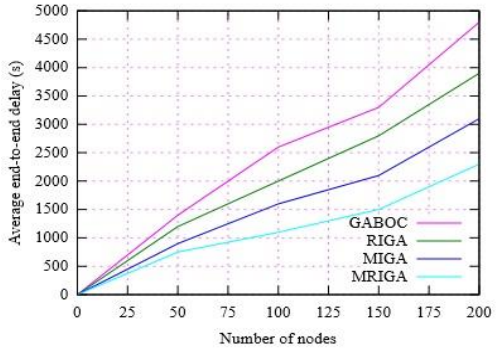

Figure 8. Average end-to-end delay.

Figure 8 shows the average end-to-end delay as the

number of nodes increases. MRIGA achieves a minimum end-to-end delay of the 2300s as compared with other schemes. It selects the energy-efficient cluster heads through the exchange of packets with minimum delay. It leads to the load-balanced formation of clusters whereas MIGA and RIGA increase the end-to-end delay of 3100s and 3900s respectively. MIGA finds the suitable memory point which has a higher fitness value than the best individual which takes a higher end-to-end delay than MRIGA. Thus the three GA schemes have a closer end-to-end delay as compared with GABOC which takes more delay for load balancing in the dynamic environments.

5. Conclusions

The topology dynamics due to node movement and energy preservation of sensor networks result in a dynamic load-balanced clustering problem. The dynamic genetic algorithm schemes such as RIGA. MIGA and MRIGA are proposed to address the issue of this clustering problem. The parameters such as node's residual energy level, distance from the centric of the cluster, mobility speed, and density of the neighbors are considered to form clusters along with the DGA schemes. The proposed DGA schemes RIGA, MIGA, and MRIGA elect suitable nodes as cluster heads to form load-balanced clusters when topology changes. The fitness function is calculated using the residual energy level of nodes and the node centrality. Then the genetic operations such as chromosome selection, crossover, and mutation are applied to find suitable cluster heads and form load-balanced clusters. The proposed DGA scheme MIGA and MRIGA store the previous generation information and utilizes this information for future generations. In RIGA, the randomly based approach is used to identify the best individuals for candidates as cluster heads. The simulation experiments have been carried out to evaluate the performance of the proposed DGA schemes. The simulation results show that the proposed DGA scheme MRIGA performs well in terms of various performance metrics such as the number of nodes alive, residual energy level, packet delivery ratio, end-to-end delay, and overhead for the formation of clusters.

References

[1] Ali S. and Madani S., "Distributed Efficient Multi Hop Clustering Protocol for Mobile Sensor Networks," *The International Arab Journal of Information Technology*, vol. 8, no. 3, pp. 302-309, 2011. https://iajit.org/PDF/vol.8,no.3/1410.pdf.

[2] Chandra A. and Parvin M., "Quasi-dynamic Load Balanced Clustering Protocol for Energy Efficient Wireless Sensor Networks," *Wireless Personal Communications*, vol. 111, pp.1589-1605, 2020. https://link.springer.com/article/10.1007/s11277-019-06942-6

[3] Cheng H. and Yang S., "Genetic Algorithms with Elitism-Based Immigrants for Dynamic Load Balanced Clustering Problem in Mobile Ad Hoc Networks," *in Proceeding of IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments*, Paris, pp. 1-7, 2011. DOI: 10.1109/CIDUE.2011.5948486

[4] Cheng H. and Yang S., "Genetic Algorithms with Elitism-Based Immigrants for Dynamic Shortest Path Problem in Ad Hoc Networks," *in Proceeding of IEEE Congress on Evolutionary Computation*, Trondheim, pp. 3135-3140, 2009. DOI: 10.1109/CEC.2009.4983340

[5] Elhabyan R. and Yagoub M., "Particle Swarm Optimization Protocol for Clustering in Wireless Sensor Networks: A Realistic Approach," *in Proceeding of IEEE International Conference on Information Reuse and Integration*, Redwood, pp. 345-350, 2014. DOI: 10.1109/IRI.2014.7051910

[6] Farahmand E., Sheikhpour S., Mahani A., and Taheri N., "Load Balanced Energy-Aware Genetic Algorithm Clustering in Wireless Sensor Networks," *in Proceeding of IEEE International Conference on Swarm Intelligence and Evolutionary Computation*, Bam, pp. 119-124, 2016.

[7] Hussain S., Matin A., and Islam O., "Genetic Algorithm for Hierarchical Wireless Sensor Networks," *Journal of Networks*, vol. 2, no. 5, pp. 87-97, 2007. DOI:10.4304/jnw.2.5.87-97

[8] Kheireddine M., Abdellatif R., and Ferrari G., "Genetic Centralized Dynamic Clustering in Wireless Sensor Networks," *in Proceeding of 5th International Conference on Computer Science and its Applications*, Algeria, pp. 503-511, 2015. https://link.springer.com/chapter/10.1007/978-3-319-19578-0_41

[9] Kuila P. and Jana P., "Energy Efficient Load-Balanced Clustering Algorithm for Wireless Sensor Networks," *in Proceeding of International Conference on Communication, Computing and Security*, New Delhi, pp. 771-777, 2011. https://doi.org/10.1016/j.protcy.2012.10.093

[10] Nandi B., Barman S., and Paul S., "Genetic Algorithm Based Optimization of Clustering in Ad Hoc Networks," *International Journal of Computer Science and Information Security*, vol. 7, no. 1, pp. 165-169, 2010. https://arxiv.org/ftp/arxiv/papers/1002/1002.2194.pdf

[11] Rani S., Ahmed S., and Rastogi R., "Dynamic Clustering Approach Based on Wireless Sensor Networks Genetic Algorithm for Iot Applications," *Journal of Wireless Networks*, vol. 26, pp. 2307-2316, 2020. https://link.springer.com/article/10.1007/s11276-019-02083-7

[12] Sirbu A. and Alecsandrescu I., "Enhanced Genetic Algorithm for Energy Efficient Dynamic Ad Hoc Wireless Sensor Networks," *in Proceeding of IEEE International Symposium on Signals, Circuits and Systems*, Iasi, pp.1-4, 2017. DOI: 10.1109/ISSCS.2017.8034920

[13] Tianshu W., Gongxuan Z., Xichen Y., and Ahmadreza V., "Genetic Algorithm for Energy-Efficient Clustering and Routing in Wireless Sensor Networks," Journal of Systems and Software, vol. 146, pp. 196-214, 2018. https://doi.org/10.1016/j.jss.2018.09.067

[14] Vavak F. and Fogarty T., "A Comparative Study of Steady State and Generational Genetic Algorithms for Use in Non-Stationary Environments," *in Proceeding of AISB Workshop on Evolutionary Computing*, UK, pp. 297-304, 1996. https://link.springer.com/chapter/10.1007/BFb0032791

[15] Wu H., Zhang Q., Nie S., Sun W., and Guan X., "An Energy Distribution and Optimization Algorithm in Wireless Sensor Networks for Maritime Search and Rescue," *International Journal of Distributed Sensor Networks*, vol. 2, pp. 1-8, 2013. DOI:10.1155/2013/725869

[16] Yang S., Cheng H., and Wang F., "Genetic Algorithms with Immigrants and Memory Schemes for Dynamic Shortest Path Routing Problems in Mobile Ad Hoc Networks," *IEEE Transactions on Systems Man and Cybernetics*, vol. 40, no. 1, pp. 52-63, 2010. DOI: 10.1109/TSMCC.2009.2023676

[17] Yang S., "Memory-Based Immigrants for Genetic Algorithms in Dynamic Environments," *in Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, USA, pp. 1115-1122, 2005. https://doi.org/10.1145/1068009.1068196

[18] Yuan X., Elhoseny M., El-Minir H., and Riad A., "A Genetic Algorithm-Based, Dynamic Clustering Method towards Improved WSN Longevity," *Journal of Network and Systems Management*, vol. 25, pp. 21-46, 2017. https://doi.org/10.1007/s10922-016-9379-7

Mohaideen Pitchai is currently working as a Professor in the Department of Computer Science and Engineering at National Engineering College, Kovilpatti, Tamilnadu, India. He received his Ph.D. degree in Computer Science and Engineering from Anna University, Chennai, India in 2015. He has published more than 20 referred research papers, 1 book, 2 patents filed and 1 patent published. He has completed 5 funded projects sponsored by DRDO, AICTE, and IE(I). His research interests include Adhoc and Sensor Networks, Cryptography, and Blockchain Technology.