# Dynamic Load Balanced Clustering using Elitism based Random Immigrant Genetic Approach for Wireless Sensor Networks

**K. Mohaideen Pitchai[1], B. Paramasivan[1], S. Anitha[1]**

[1] Department of Computer Science and Engineering, National Engineering College, Kovilpatti–628503, Tamilnadu, India

## Abstract

Wireless Sensor Network (WSN) consists of a large number of small sensors with restricted energy. Prolonged network lifespan, scalability, node mobility and load balancing are important needs for several WSN applications. Clustering the sensor nodes is an efficient technique to reach these goals. WSN have the characteristics of topology dynamics because of factors like energy conservation and node movement that leads to Dynamic Load Balanced Clustering Problem (DLBCP). In this paper, Elitism based Random Immigrant Genetic Approach (ERIGA) is proposed to solve DLBCP which adapts to topology dynamics. ERIGA uses the dynamic Genetic Algorithm (GA) components for solving the DLBCP. The performance of load balanced clustering process is enhanced with the help of this dynamic GA. As a result, the ERIGA achieves to elect suitable cluster heads which balances the network load and increases the lifespan of the network.

**Keywords:** Genetic Algorithm, Elitism, Load Balanced Clustering, Random Immigrants, Wireless Sensor Network

## 1. Introduction

In recent years, an energy efficient design of a Wireless Sensor Network (WSN) has become a leading area of research. WSN consists of base stations and numbers of nodes (wireless sensors). Energy efficiency is the essential criteria to increase the lifespan of the network. Clustering is the key technique used for load balancing to extend the lifetime of a sensor network by reducing energy consumption. Cluster structure can prolong the lifetime of the sensor network by making the cluster head aggregate information from the nodes in the cluster and send it to the base station.

The clustering concept offers tremendous benefits for WSN. The importance of the clustering problem can be summarized in two aspects. First, it plays an essential role in effective network management. The flat network infrastructure of WSN encounters the scalability problem when the network size keeps rising. In WSNs node mobility, scalability is more challenging factor. Therefore, effective network management is extremely important. So far, clustering is the most efficient way to manage WSN. Second, clustering serves as the foundation for many other key problems in WSN (e.g., routing, intrusion detection, topology control, and backbone construction). All these problems are solved based on a well clustered network structure. In this paper, the load balance is used a clustering metric since it is a vital application requirement. It can guarantee the fairness for all the cluster heads in terms of the workload. Moreover, the load balanced clustering will prolong the lifespan of the cluster structure since every cluster head can evenly consume its battery energy.

The objective of this paper is to solve the Dynamic Load Balanced Clustering Problem (DLBCP) in WSN and also to maximize the network lifetime and minimizes the network energy consumption. In this paper, a novel load balanced clustering approach using Genetic Algorithm (GA) is proposed. This proposed work forms clusters with optimal cluster heads. Here the genetic algorithm is applied along with the weighted clustering algorithm and this improves the performance of the network and also balances the load in the network.

## 2. Related work

Various works has been surveyed which provides solutions for the DLBCP problem. Some of the important works has been reviewed which are described as follows. Huicheng et al [1] formulates the DLBCP into a dynamic optimization problem. They proposed to use a series of dynamic Genetic Algorithms (GA) to solve the DLBCP in MANETs. In this dynamic GA, each individual represents a feasible clustering structure and its fitness is evaluated based on the load balance metric. Various dynamics handling techniques are introduced to help the population to deal with the topology changes and produce closely related solutions in good quality. Their experimental results show that these GA can work well for the DLBCP and outperforms traditional GA that do not consider dynamic network optimization requirements. Xiaohui Yuan et al [2] proposed a genetic algorithm based, self-organizing network clustering (GASONeC) technique that has a framework to dynamically optimize wireless sensor node clusters. In GASONeC, the residual energy, the expected energy expenditure, the distance to the base station and the also the range of nodes in the vicinity are employed in search for a best, dynamic network structure. Reconciliation these factors is that the key of organizing nodes into applicable clusters and designating a surrogate node as cluster head. Here the authors contributed the work into two folds. First, a GA based clustering method is proposed that maintains the dynamic cluster structure. Second, it balances the energy consumption of every node and improves the network lifetime.

Kaliappan **e**t al [3] used energy metric in genetic algorithm (GA) to solve the DLBCP. They used genetic algorithms such as elitism based immigrants genetic algorithm (EIGA) and memory enhanced genetic algorithm (MEGA) to solve DLBCP. These schemes select an optimal cluster head by considering the distance and energy parameters. They used EIGA to maintain the diversity level of the population and MEGA to store the old environments into the memory. It promises the load balancing in cluster structure to increase the lifetime of the network. Their experimental result shows that their proposed schemes increases the network lifetime and reduces the total energy consumption.

Vipin pal et al [4] presented a genetic algorithm based cluster head selection for centralized clustering algorithms to have a better load balanced network than the traditional clustering algorithm. Here the cluster head is selected according to their residual energy and takes care of trade-off of inter and intra cluster communication distance. Also their proposed scheme optimizes the number of cluster head for a round. Their simulation shows that their proposed solution finds the optimal cluster heads and has prolonged network lifetime than the traditional clustering algorithms.

Pratyay Kuila [5] proposed the load balanced clustering problem in WSN. In this paper, the author used the genetic algorithm approach to solve the load balancing problem in WSN. This proposed algorithm performs well for both equal as well as unequal load of the sensor nodes. The author performs in depth simulation of the proposed method and compares the results with some evolutionary based approaches and other related clustering algorithms. The results demonstrate that the proposed algorithm performs better than all such algorithms in terms of varied performance metrics like load balancing, execution time, energy consumption, number of active sensor nodes, and number of active cluster heads and also the rate of convergence.

Sheng Xiang Yang et al [6] proposed the random immigrants and memory scheme. The random immigrant scheme addresses dynamic environments by maintaining the population diversity while the memory scheme aims to adapt genetic algorithms quickly to new environments by reusing historical information. Experimental results show that the memory-based and elitism-based immigrant schemes efficiently improve the performance of genetic algorithms in dynamic environments.

## 3. Dynamic Load Balanced Clustering Problem

The DLBCP problem is described as follows. For a given sensor nodes, the proposed algorithm is to find a set of cluster heads from the network and each cluster head serves the same number of cluster members. Due to some environmental changes the topology of the network also changes from time to time. Therefore the objective of this paper is to find suitable cluster heads in WSN with dynamic environment.

The proposed algorithm consists of various phases like genetic representation, population initialization, fitness function calculation, selection, crossover, and mutation. Initially the deployed sensor nodes are represented using genes. Then populations are generated using random permutation of chromosomes. In genetic algorithm, each chromosome represents a potential solution. In the next phase, the weight value of each node presented in a chromosome is calculated by using weighted clustering process [7,8]. Then the procedure for cluster head selection is applied to select suitable cluster heads from each chromosome. The calculated weight values for each node in the chromosome is used to find out the fitness value of that chromosome by taking the sum of weight values of all cluster heads in this particular chromosome. Next the selection operator is formulated to ensure that the better chromosomes of the population with higher fitness value have greater probability of being selected for mating. In order to achieve the same, roulette wheel selection method is used. Finally the important GA operations namely cross over and mutation is applied.

The crossover helps to select suitable best individual from current population and to produce the new population. Mutation alters one or more gene values in a chromosome from its initial state and the new gene values are added to the gene pool. From the new gene values the better solution may be obtained. As a result a new child chromosome is generated whose corresponding cluster head sets are the finalized cluster heads of the given network topology. Whenever the environment change s, the network topology also changed which leads to change the cluster structure and also the cluster heads will be changed. Hence this proposed method uses the elitism based random immigrant method to adapt dynamic topology changes. Finally the load balanced cluster structure is obtained. Figure 1 shows the flowchart of the proposed ERIGA algorithm.

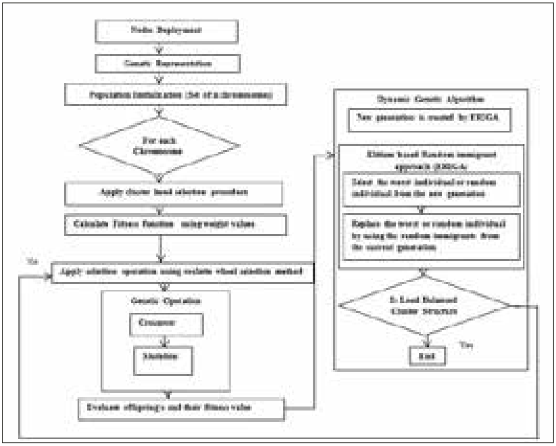

Node Deployment
Genetic Representation
Population Initialization (Set of n chromosomes)
For each Chromosome
Apply cluster head selection procedure
Calculate Fitness function using weight values
Apply selection operation using roulette wheel selection method
Genetic Operation
Crossover
Mutation
Evaluate offsprings and their fitness value
Dynamic Genetic Algorithm
New generation is created by ERIGA
Elitism based Random Immigrant approach (ERIGA)
Load Balanced Cluster Structure
End
Yes

Fig.1 Flowchart of ERIGA Algorithm.

### 3.1 Genetic Representation

In WSN, the nodes which are presented initially are considered as population. It is denoted as

$$P = \{n_1, n_2, \ldots, n_m\} \tag{1}$$

Each node in the network has a node ID. Each node ID represents a gene. The random permutation of node IDs are used to represent a chromosome.

### 3.2 Population Initialization

The initial population is represented as $P_{GA}$ and it consists of 'q' (=m!) chromosomes. It is denoted as

$$P = \{CHr_o, CHr_1, \ldots, CHr_{q-1}\} \tag{2}$$

In GA, each chromosome represents a potential solution. If 20 numbers of nodes are presented in a network then there are 20! Chromosomes can be obtained. It guarantees that each chromosome has no duplicate ID and it explores the population diversity for each chromosome. Hence a random set of cluster heads can be derived by using random permutation of node IDs.

### 3.3 Weighted Clustering

After the formation of the initial population, weight $w_v$ of each node was calculated by using weighted clustering procedure. Genetic algorithm uses those values to sum up for all the cluster heads for each chromosome. $W_v$ is defined as

$$W_v = W_1\Delta_v + W_2 m_v \tag{3}$$

Where $\Delta_v$ is the degree difference and $m_v$ is the average speed of the nodes. The corresponding weighing factors are such that.

$$\sum_{i=1}^{2} W_i \tag{4}$$

This procedure consists of the following steps.

1. Find the degree of each node $d_v$. It is defined as the number of neighbor nodes N(v). It can be calculated as

$$d_v = N(v) = \sum_{v' \in V, v' \neq v} (dist(v, v') < tx_{range}) \tag{5}$$

Where $tx_{range}$ is the transmission range of v and V represents the set of nodes $v_i$. The neighborhood of a cluster head is the set of nodes within its transmission range.

2. Degree difference ($\Delta_v$) is one of the performance parameter for load balancing. It is defined as the difference of ideal node degree (δ) and actual degree (connectivity) of that node. Degree of node ($d_v$) is the number of neighbors of node v that are in the transmission range. Ideal degree is the number of neighbors that a cluster head can handle effectively. The degree difference for every node v can be calculated as

$$\Delta_v = |d_v - \delta| \tag{6}$$

3. Mobility is an important factor to decide efficient cluster heads and it is used to avoid frequent cluster head changes. It is computed using the average running speed for every node till the current time T. The mobility of a node $m_v$ is calculated as

$$m_v = (1/T) \cdot \sum_{t=1}^{T} \sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2} \tag{7}$$

Where $(x_t, y_t)$ and $(x_{t-1}, y_{t-1})$ are the coordinates of node v at time t and t-1 respectively.

4. Weighing factors are chosen in such a way that $w_1 + w_2 = 1$. Calculate the combined weight $w_v$ for each node v, where

$$w_v = w_1\Delta_v + w_2 m_v \tag{8}$$

where $w_1$ and $w_2$ are weighting factors. Repeat the steps 2-4 for remaining nodes.

## 3.4 An illustrative scenario

Consider a network consists of 20 nodes with IDs ranging from 0 to 19. The transmission range for each node is represented by a circle with equal to radius. The weighted clustering algorithm is demonstrated with figures 2 to 6. Figure 2 shows that the initial configuration of 20 nodes with node IDs

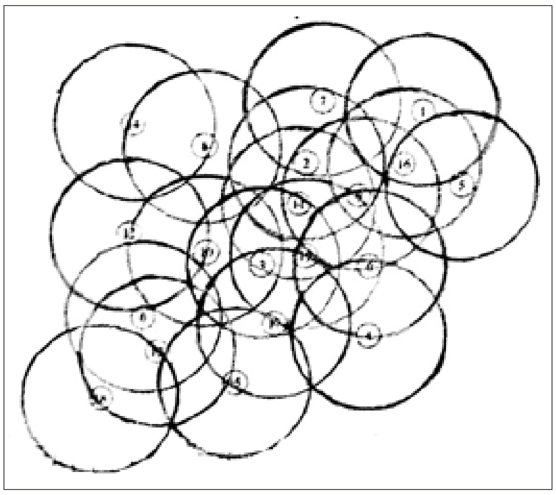


Fig.2. Initial configuration consists of 20 nodes.

Figure 3 shows the neighbors of each node. The neighbors of the each node v are defined by its degree $d_v$. Step 1 implies that degree, $d_v$, which is the total number of neighbors of a node. The degree difference, $\Delta_v$, of each node is calculated in step 2 with ideal node degree δ=2.The mobility of node is calculated in step 3 and the $m_v$ values are chosen randomly. If $m_v$=0 means the node does not move ie static node.

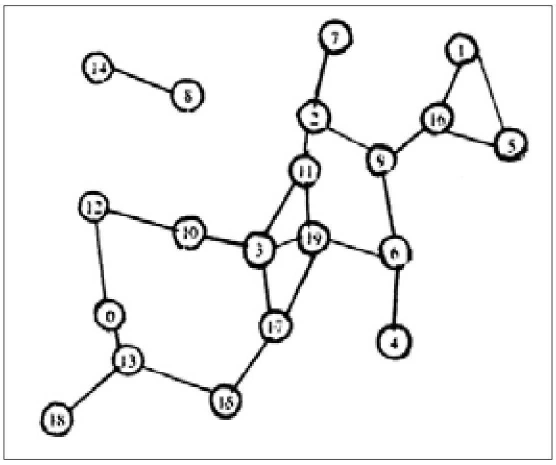


Fig.3. Neighbors identified for each node.

In Figure 4, the arrows indicate the speed and direction of movement of every node. The weighted metric $w_v$ for each node is computed in step 4.The values of weights are considered as $w_1$=0.5 and $w_2$=0.5. So these weighting factors are chosen randomly according to $w_1+w_2=1$.

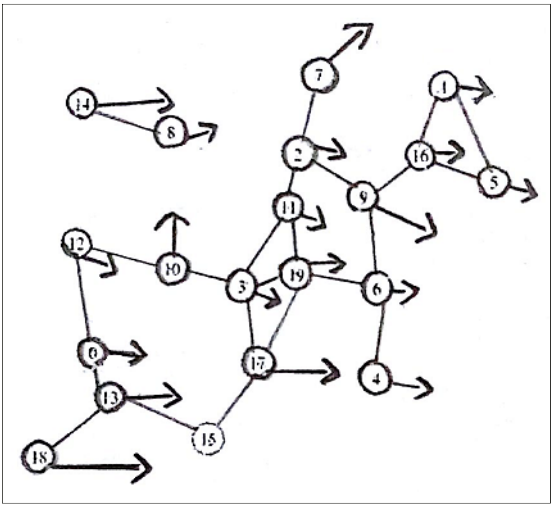


**Fig.4. Mobility of the nodes.**

Table 1 represents the execution of weighted clustering algorithm. The cluster head results are shown in the form of table. The first column in the table represents the node ID. The values in the second column of the table represent the degree of the each node and the degree difference values are represented in third column of the table. The randomly chosen mobility values are represented in fourth column of the table. Finally the fifth column shows the weighted values of each node.

**Table 1. Execution of weighted clustering**

| NODE ID | $d_v$ | $\Delta_v$ | $m_v$ | $W_v$ |
|---|---|---|---|---|
| 0 | 2 | 0 | 2 | 1 |
| 1 | 2 | 0 | 1 | 0.5 |
| 2 | 3 | 1 | 1 | 1 |
| 3 | 4 | 2 | 1 | 1.5 |
| 4 | 1 | 1 | 2 | 1.5 |
| 5 | 2 | 0 | 1 | 0.5 |
| 6 | 3 | 1 | 1 | 1 |
| 7 | 1 | 1 | 2 | 1.5 |
| 8 | 1 | 1 | 1 | 1 |
| 9 | 3 | 1 | 3 | 2 |
| 10 | 2 | 0 | 2 | 1 |
| 11 | 3 | 1 | 1 | 1 |
| 12 | 2 | 0 | 1 | 0.5 |
| 13 | 3 | 1 | 2 | 1.5 |
| 14 | 1 | 1 | 3 | 2 |
| 15 | 2 | 0 | 0 | 0 |
| 16 | 3 | 1 | 1 | 1 |
| 17 | 3 | 1 | 3 | 2 |
| 18 | 1 | 1 | 4 | 2.5 |
| 19 | 4 | 2 | 1 | 1.5 |

This is stored in a list where each node is pointing to its neighbor's list as it is next position that is used to compute the object function. Table 2 represents the neighbor lists of each node.

**Table 2. Neighbor list of each node**

| $w_v$ values | Node id of all nodes | Neighbor's list |
|---|---|---|
| 1 | 0 | → 12 → 13 → |
| 0.5 | 1 | → 5 → 16 → |
| 1 | 2 | → 7 → 11 → |
| 1.5 | 3 | → 17 → 10 → 11 → 19 |
| 1.5 | 4 | → 6 → |
| 0.5 | 5 | → 16 → 1 → |
| 1 | 6 | → 4 → 19 → 9 |
| 1.5 | 7 | → 2 → |
| 1 | 8 | → 14 → |
| 2 | 9 | → 2 → 16 → 6 → |
| 1 | 10 | → 3 → 12 → |
| 1 | 11 | → 2 → 3 → 19 → |
| 0.5 | 12 | → 0 → 10 → |
| 1.5 | 13 | → 0 → 18 → 15 → |
| 2 | 14 | → 8 → |
| 0 | 15 | → 13 → 17 → |
| 1 | 16 | → 9 → 1 → 5 → |
| 2 | 17 | → 3 → 15 → 19 → |
| 2.5 | 18 | → 13 → |
| 1.5 | 19 | → 11 → 6 → 17 → 3 → |

## 3.5 Cluster Head Selection

The algorithm goes through each node in this list and checks three conditions in order to select the current node as a cluster head.

1. If the node under consideration is not already a cluster head.
2. The node is not a member of any cluster heads.
3. The actual number of neighbor is less than threshold value (T=4).

If these three conditions are satisfied, then the nodes are selected as a cluster heads and inserted into the set of cluster heads for that particular chromosome. The selected cluster heads are shown in the figure 5.

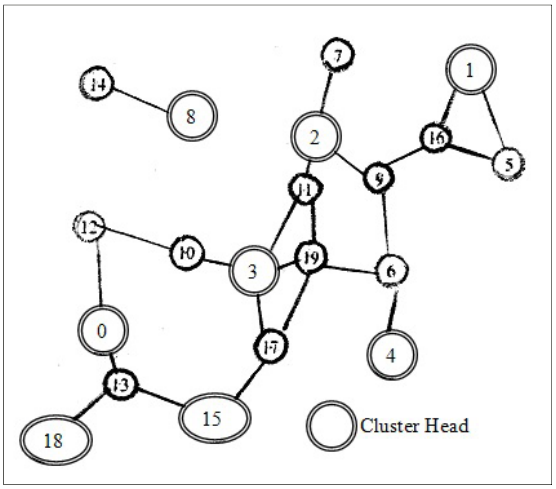
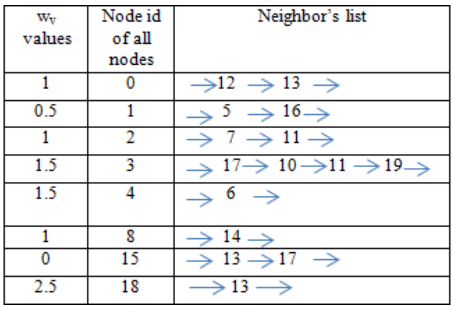


**Fig. 5. Cluster heads selection**

**Table 3.** Cluster heads set for a single chromosome.



| $w_V$ values | Node id of all nodes | Neighbor's list |
|---|---|---|
| 1 | 0 | →12 → 13 → |
| 0.5 | 1 | → 5 → 16→ |
| 1 | 2 | → 7 → 11 → |
| 1.5 | 3 | → 17→ 10→11 →19→ |
| 1.5 | 4 | → 6 → |
| 1 | 8 | → 14 → |
| 0 | 15 | → 13 →17 → |
| 2.5 | 18 | →13 → |

## 3.6 Fitness Function Calculation

After the cluster heads are chosen, the calculated $w_v$ values for each node is used to find out the fitness value of the chromosome by taking the summation of $w_v$ values of all cluster heads in this particular chromosome. Since the orders of appearance of node IDs of the chromosome are different, each chromosome will have a different set of cluster heads which in return will have the different fitness value. This fitness function is used to estimate the quality of the chromosome. The fitness function of this chromosome is 9. So for each chromosome the fitness function is calculated and after that it chooses the highest fitness value chromosomes. These fitness chromosomes are then served as parents for the next generation.

## 3.7 Selection

Selection plays a vital role in improving the average quality of the population by passing the prime quality chromosome to the next generation. The selection operator is formulated to ensure that better chromosomes of the population with higher fitness vale have greater probability of being selected for mating. Roulette Wheel selection is used to select individuals based on their fitness values. Therefore the probability of choosing an individual depends directly on its fitness value. This probability is defined as

$$P_i = \frac{F_i}{\sum_{j=1}^{n} F_j} \tag{9}$$

where $F_i$ and n are the fitness chromosome and size of the population respectively. The size of each individual corresponds to fitness value of associated individuals. The circumference of the roulette wheel is the sum of all fitness values of individuals. The fittest chromosome occupies the largest interval, whereas the least fit has small interval within the roulette wheel. In this method, circular wheel is divided and a fixed point is chosen on the wheel circumference and the wheel is rotated. The region of the wheel which comes in front of the fixed point is chosen as the parent and for the second parent same process is repeated. Two individuals are then chosen randomly based on the probabilities and produces offspring.

## 3.8 Crossover

Crossover is a genetic operator that mixes two chromosomes to provide a new Chromosome. The crossover is used to choose the new chromosome that may be better than both of the parents, if it takes the best characteristics from each of the parents. They help to select suitable best individual from current population and to produce the new population. Here order 1 crossover method is used. In this method, initially selects two parent chromosomes and calculate the residual energy for all the nodes and then arrange the parent chromosomes by value of residual energy (i.e.) from high remaining energy to low remaining energy and then selects a random swath of consecutive gene from two parent chromosomes. After that from parent chromosome 1 and it drops the swath down to child 1 and remaining genes are taken from the second parent chromosome. From the second parent chromosome starting on the right side of the swath, grab genes from parent 2 and then insert them in child 1 at the right edge of the swath. This procedure is repeated for the second child chromosome. For the second child chromosome swap area is selected from the parent chromosome and then remaining genes are genes are grabbing from the parent chromosome 1. Figure 6 shows the illustration of the crossover.

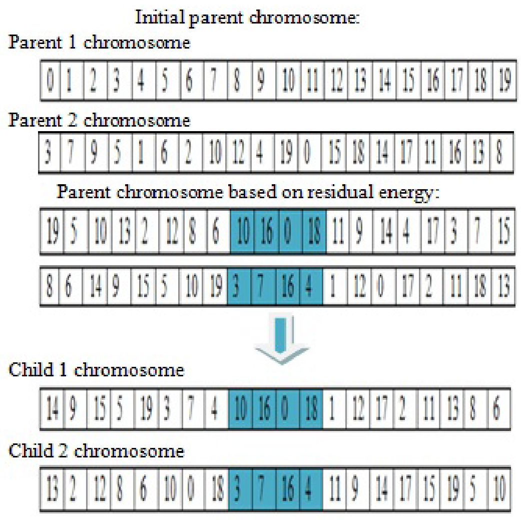


**Fig. 6. Illustration of crossover.**

## 3.9 Mutation

Mutation is a genetic operator used to preserve genetic diversity from one generation of a population of a chromosome to the consequent generation. Mutation alters one or more gene values in a chromosome from its initial state and the new gene values are added to the gene pool. From the new gene values the better solution may be able to obtain. Here gene swapping method is used for generating a children chromosome from a parent chromosome by changing the value of some genes.

In Mutation, initially the residual energy for all the nodes are calculated and selects a parent chromosomes and then arrange the parent chromosome by value of residual energy (i.e.) from high remaining energy to low remaining energy. Then selects two positions on the chromosome at random, and interchanges the values. This is common in permutation based encodings. In the example, genes 9 and 0 are swapped and this is shown in the figure 7.

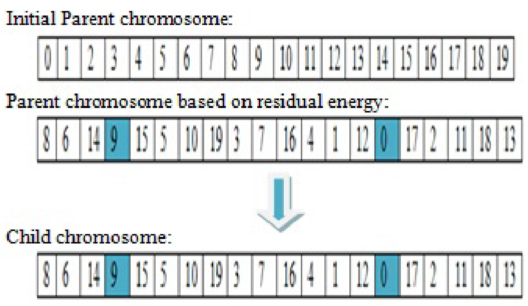


**Fig.7. Illustration of mutation**

As a result of mutation, a new child chromosome is generated whose corresponding cluster head sets are the finalized cluster heads of the given network topology. This procedure is repeated until the new population is equal to the initial population. These are the standard genetic algorithm operations. If the topology changes, then the dynamic genetic operations such as immigrant schemes or memory based schemes can be used. This proposed method uses the elitism based random immigrant approach for the dynamic load balanced clustering problem.

## 3.10 Elitism based Random Immigrant Approach

Standard genetic algorithm is not suite well for the dynamic environments. If the load balanced clustering problem is considered in a dynamic environment, then it is termed as dynamic optimization problem. There are various methods are used to solve dynamic optimization problem. The best method is to enhance the performance of the dynamic optimization problem through immigrant schemes. The basic idea behind the immigrant scheme is to introduce new individuals into the current population by replacing the worst individuals. It can remove the diversity level of the population. The above said conventional genetic operations are combined with immigrant schemes to solve dynamic load balanced clustering problem.

Here an elitism based random immigrant approach is used. After the crossover and mutation operations, every chromosome is evaluated and checks their fitness value. Due to some environment conditions, the network structure may change and that affects the fitness value of the chromosomes (i.e.) if the topology changes then the cluster structure are changed and also their corresponding cluster heads will changed. The fitness value is calculated based on the cluster heads of each chromosome. So the fitness value is affected after the environment changes. So this affects the current population and for that this work uses the Elitism based Random Immigrant Genetic Algorithmic approach (ERIGA).

In ERIGA, new immigrants which include the random and elite immigrants are generated first and then they replace the worst individual in the current population. The random immigrants are feasible solutions generated randomly which aims to increase the diversity level of the population. The elite immigrants are used to reserve the better individuals in current generation to the next generation. From the previous generation, the chromosomes with the higher fitness value (elites) are stored and this elites can use for the replacement strategy.

After the execution of standard GA operations, the current population is evaluated. If the chromosome with the least fitness value is found, then it is replaced with the higher fitness value chromosomes from the previous generation and then performs the standard genetic operations. Finally update the current generation with new individuals. This process is repeated until the maximum population is reached. The pseudo-code for the ERIGA is shown below.

```
Algorithm ERIGA
Evaluate the current population
//Perform elitism based immigrant mechanism
Generate elite immigrant from p(t-1)
Evaluate elite immigrants
// (i.e. immigrant with high fitness value)
If environment change is detected
Then
Replace the worst individual (i.e. individual with low
fitness value) in p (t) with generated immigrant from
p(t-1)
Else
Perform basic genetic operations
Update population until the termination criteria is met
End
```

## 4. Simulation and Results

The simulation experiments are conducted to evaluate the performance of the proposed ERIGA algorithm for the concerned Dynamic load balanced clustering problem. Simulation has been carried out using MATLAB tool. The simulation parameter settings are tabulated in Table 4. The EIRGA algorithm is executed by considering an initial population of 100 chromosomes. For crossover operation, ERIGA selects the best 10% chromosomes using Roulette Wheel selection. The crossover rate is taken as 0.6 and mutation rate as 0.03. The algorithm is run for 20 iterations. The ERIGA algorithm is compared with Standard GA (SGA) algorithm.

**Table 4. Simulation Parameters Settings**

| Parameter | Value/Method |
|---|---|
| Deployment Area | 100 x 100 $m^2$ |
| Base Station Location | (50,50) |
| Network Size | 20-100 |
| Initial Energy of Nodes | 0.5J |
| Population Initialization | Random |
| Number of Generations | 10-100 |
| Selection Schema | Roulette Wheel Selection(RWS) |
| Cross Over Rate | 0.6 |
| Mutation Rate | 0.03 |
| Replacement Rate | 0.1-0.6 |
| Selection Ratio of Random Immigrants | 0.2 |
| Selection Ratio of Elite Immigrants | 0.4 |

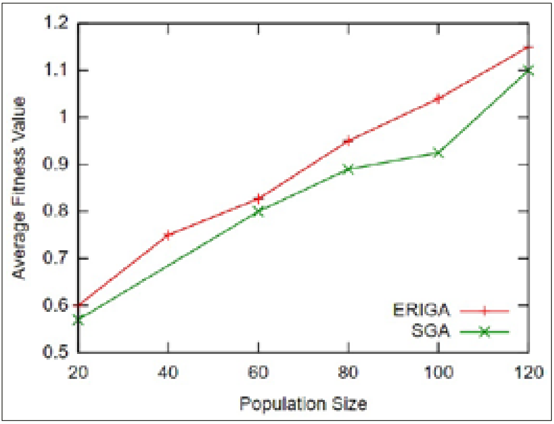


**Fig. 8. Population Size Vs Average Fitness Value.**

Figure 8 shows the comparison of average fitness value with SGA and ERIGA. The average fitness value is increased by increasing the size of the population. During the population size of 100, the average fitness value of SGA is 0.9. But in the case of ERIGA, the value is 1.1. This is due to the replacement of worst individuals in the identified chromosome with elite chromosomes. When the replacement rate is increased, the difference of average fitness value will be increased.

Figure 9 shows the comparision among number of cluster heads with the varying size of population. It is observed that during the population size of 60 and 80, the number of cluster head of ERIGA is 12 and 11. This variation is due to the variation in mobility speed of the nodes. There is a linear increase in the number of cluster heads after the population size is varied from 80 to 140. It is also observed that during the population size of 60, both SGA and ERIGA has same number of cluster heads. In that scenario, there is no mobilty is considered in ERIGA.

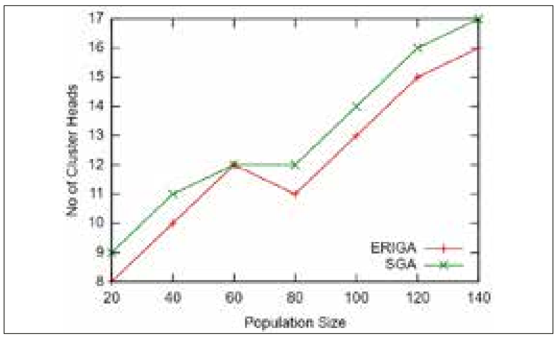


**Fig. 9. Population Size Vs Number of Cluster Heads**

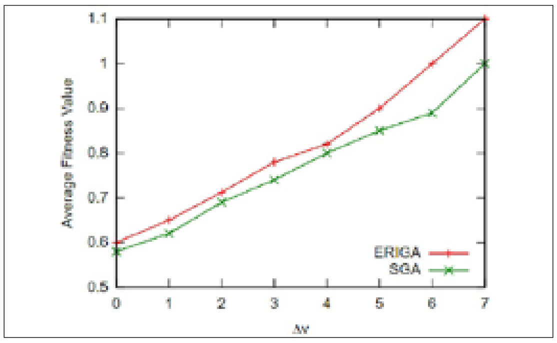


**Fig. 10. Average Fitness Value by varying the value of $\Delta_v$**

Figure 10 shows that the average fitness value of ERIGA and SGA for different values of $\Delta_v$. In ERIGA, the average fitness value varies from 0.6 to 1.1 and for SGA it varies from 0.55 to 1. When the value of $\Delta_v$ is 6, the average fitness value of ERIGA and SGA are 1 and 0.85 respectively. The degree difference $\Delta_v$ value is due to the mobility consideration. Hence the ERIGA achieves highest fitness value as compared with SGA. Also whenever a $\Delta_v$ value varies, the corresponding $W_v$ values are varied and thus it changes the fitness value of that particular chromosome.

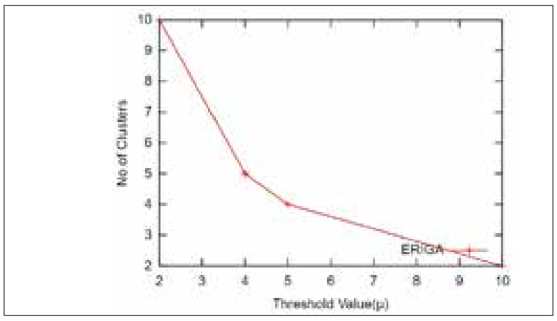


**Fig. 11. Threshold Value Vs Number of clusters**

Figure 11 shows that the number of clusters formed based on the threshold values for a network of size 20 nodes. Here the threshold value is set to 4. The threshold value represents the number of members within a cluster. According to that number of clusters is formed corresponding to the threshold values.

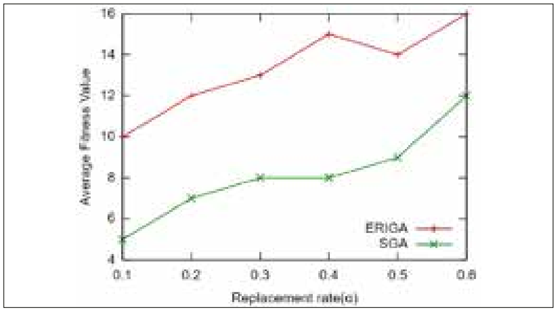


**Fig. 12. Replacement Rate Vs Average Fitness Value**

Figure 12 depicts that the average fitness value of ERIGA and SGA with different values of replacement rate. Replacement rate is defined as how much of chromosomes in the current generation are replaced with the previous generation. When the replacement rate is 0.4, the average fitness value of SGA is 8. But this value is increased up to 15 for ERIGA. Since the replacement of worst individuals in the previous generation with elites in the current generation. These results will improve the performance of the proposed ERIGA algorithm.

## 5. Conclusion

The characteristic such as topology dynamics due to energy conservation and node movement in WSN leads to dynamic load balanced clustering problem. Load balancing and reliable information transfers among all the nodes are essential to prolong the lifespan of the network. In this paper, genetic algorithm is used to solve the dynamic load balanced clustering problem. Initially a standard genetic approach was applied to solve the load balanced clustering problem. In order to adapt a dynamic environment, the standard approach is enhanced to a dynamic genetic algorithmic approach called elitism based random immigrant approach. This is applied to find suitable cluster heads whenever topology changes. The simulation results show that the dynamic approach is more suitable than a static method to solve the dynamic load balanced clustering problem. In future, heterogeneous WSN may be considered for testing the applicability of the proposed load balanced algorithms SGA and ERIGA.

## References

[1] Hui Cheng, Sheng Xiang Yang and Jiannong Cao, "Dynamic genetic algorithms for the dynamic load balanced clustering problem in mobile ad hoc networks", Journal of Expert Systems with Applications, Elsevier Publications, Vol.40, pp.1381-1392, 2013.

[2] Xiaohui Yuan, Mohamed Elhoseny, Hamdy K El-Minir and Alaa M. Riad, "A genetic algorithm based dynamic clustering method towards improved WSN longevity", Journal of Network and System Management, Springer Publications, Vol.24, pp.1-26, 2016.

[3] Kaliappan, Mariappan, Viju Prakash and Paramasivan, "Load Balanced Clustering Technique in MANET using Genetic Algorithms", Defence Science Journal, Vol. 66, No. 3, pp. 251-258, 2016.

[4] Pratyay kuila, Suneet K Gupta and Prasanta K Jana "A novel evolutionary approach for load balanced cluster problem for wireless sensor networks", Journal of Swarm and Evolutionary Computation, Vol.12, pp.48-56, 2013.

[5] Sheng Xiang Yang, "Genetic Algorithms with Memory and Elitism based Immigrants in Dynamic Environments", Journal of Evolutionary Computation, Vol. 16(3), pp.385–416, 2008.

[6] Dipak Wajgi, Nilesh Singh, "Load balancing based approach to improve lifetime of wireless sensor network", International Journal of Wireless & Mobile Networks, Vol. 4, No. 4, pp.155-167, 2012.

[7] Vaibhav Deshpande, Arvind Bhagat Patil, "Clustering for improving lifetime of wireless sensor network: A Survey", International Journal of Engineering Science Invention, Vol. 2, pp.1-6, 2013.

[8] Mainak Chatterjee, Sajal Das and Damla Turgut, "WCA: A Weighted Clustering Algorithm for Mobile Ad Hoc Networks", Journal of Cluster Computing, Vol.5, No.2, pp.193–204, 2002.