IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 16, Issue 6, Ver. I (Nov – Dec. 2014), PP 14-24
www.iosrjournals.org

Scalable parallel clustering using modified Firefly algorithm

Juby Mathew, Dr.R Vijayakumar
School of Computer Science
Mahatma Gandhi University, Kottayam, Kerala, India

***Abstract*:** *Clustering is the process of assigning data objects into a set of disjoint groups called clusters so that objects in each cluster are more similar to each other than objects from different clusters. We try to exploit computational power from the multicore processors. We need a new design on existing algorithms and software. Firefly algorithm is one of the metaheuristic algorithms which are used for solving optimization problems. The existing clustering algorithms either handle different data types with inefficiency in handling large data or handle large data with limitations in considering numeric attributes. Hence, parallel clustering has come into picture to provide crucial contribution towards clustering large data. In this paper, we have developed a scalable parallel clustering algorithm using FA and genetic algorithm to cluster large data. Modified FA algorithm does not handle the large data effectively. So, our ultimate aim is to design and develops an algorithm in parallel way by considering data. The experimental analysis will be carried out to evaluate the feasibility of the new combined clustering approach. The experimental analysis showed that the proposed approach obtained upper head over existing method in terms of accuracy and time. Most of the programming languages doesn't provide multiprocessing facilities and hence wastage of processing resources. In order to utilize the intrinsic capabilities of a multi-core processor the software application must be able to execute tasks in parallel using all available CPUs. To achieve this we can use fork/join method in java programming. It is the most effective design method for achieve good parallel performance.*

Keywords: *Parallel clustering, large data, Firefly, genetic algorithm.*

I. INTRODUCTION

Optimization problem is one of the most challenging problems in the field of operation research. The goal of the optimization problem is to find the set of variables that results into the optimal value of the objective function, among all those values that satisfy the constraints. Many new types of optimization algorithms have been explored. Firefly algorithm is one of the powerful population–based algorithms inspired by nature which is used for solving optimization problems [1]. The field of nature inspired computing and optimization techniques have evolved to solve the difficult optimization problems in diverse fields of engineering, science and technology. The Firefly algorithm is one of the several nature inspired algorithms that have been developed in the recent past and is inspired from the flashing behaviour of the fireflies. The flashing behaviour of the fireflies is to attract other fireflies in the group for mating. The less bright firefly will be attracted by the brighter one. As all the fireflies are assumed to be unisexual, each firefly is attracted to the other. This process is mimicked in the algorithm to find the solution to objective function. This swarm intelligence optimization technique is based on the assumption that solution of an optimization problem can be shown as a firefly which glows proportionally to its quality in a considered problem setting. Consequently, each brighter firefly attracts its partners, which makes the search space being explored efficiently. Yang used the FA for nonlinear design problems [2] and multimodal optimization problems [3] and showed the efficiency of the FA for finding global optima in two dimensional environments.

Another well-known nature-inspired algorithm is genetic algorithm (GA). GA is inspired by the process of natural evolution. It starts with a population of chromosomes and effects changes by genetic operators. Three key genetic operators are crossover, mutation, and selection operators [4]. Our algorithm proposed in this paper combines attributes of firefly mating and genetic algorithm. GA is used as the core of our algorithm while the attributes mentioned are used to compose a new selection operator.

We implement our proposed approach using Fork/Join method in JAVA. Fork/Join parallelism [5] is a style of parallel programming useful for exploiting the parallelism inherent in divide and conquer algorithms. Fork-join executor framework has been created which is responsible for creating one new task object which is again responsible for creating new sub-task object and waiting for sub-task to be completed. Internally it maintains a thread pool and executor assign pending task to this thread pool to complete when one task is waiting for another task to complete.

The rest of the paper is organized as follows. Section 2 describes related work .Section 3 presents the proposed implementation. Section 4 shows experimental results and evaluations. Finally, the conclusions and future work are presented in Section 5.

II. Related Work

Many swarm intelligence algorithms were developed and enhanced over the time. Particle swarm optimization (PSO) is well-known swarm intelligence algorithm which simulates collective behaviour of fish or birds. It was developed by Eberhard and Kennedy [6] for optimizing nonlinear functions in multidimensional space. Ant colony optimization (ACO) simulates foraging behaviour of ants which use pheromone substance to find the shortest path between their nests and food source [6]. This algorithm, as well its modifications, found wide-spread use in many practical applications [7]. Karaboga was inspired by behaviour of honey bee swarms and devised artificial bee colony (ABC) algorithm [8]. ABC algorithm uses three kinds of artificial bees: employed, onlooker and scout. Employed bees and onlookers are exploiting search space, while scout maintain diversification through exploration. There are also other approaches which model bee's behaviour [9].

Cuckoo search (CS) algorithm is relatively new metaheuristic for continuous optimization problems [10]. CS uses Levy flights search pattern and shows very promising performance. Firefly (FF) algorithm is a novel metaheuristic which mimics flashing behaviour of light bugs [11]. As the primary purpose of firefly's flash is to act as a signal system to attract other fireflies, researches have been attracted to incorporate such behaviour into computation algorithm. Very similar to FF algorithm is glow-worm swarm algorithm which has been used in many practical fields including robotics [12]. Besides above mentioned, there are also other swarm intelligence algorithms are often used with other evolutionary computation (EC) algorithms [13], and applied to wide variety of practical problems [13], [14].To obtain acceptable computational speed on huge datasets, most researchers turn to parallelizing scheme. Md. Mostofa Ali Patwary et al [15] have presented a scalable parallel OPTICS algorithm (POPTICS) designed using graph algorithmic concepts. To break the data access sequentially, POPTICS exploits the similarities between the OPTICS algorithm and PRIM's Minimum Spanning Tree algorithm. Additionally, they used the disjoint-set data structure to achieve a high parallelism for distributed cluster extraction.

Genetic algorithms are randomized search and optimization techniques guided by the principles of evolution and natural genetics, having a large amount of implicit parallelism [3] [7].GA are inspired by Darwin's theory about evolution. An initial population of possible solutions is randomly selected and then operated on by GA operators. Three key GA's operators are crossover, mutation, and selection operators. The crossover operator exchanges genes between a mating pair and creates an offspring. The mutation operator randomly changes the genes of an offspring, and the selection operator selects a group of offspring to be the next generation of population.

GAs perform search in complex, large and multimodal landscapes, and provide near-optimal solutions for objective or fitness function of an optimization problem. In GAs, the parameters of the search space are encoded in the form of strings (called chromosomes). A collection of such strings is called a population. Initially, a random population is created, which represents different points in the search space. An objective and fitnes*s* function is associated with each string that represents the degree of goodness of the string. Based on the principle of survival of the fittest, a few of the strings are selected and each is assigned a number of copies that go into the mating pool. Biologically inspired operators like *crossover* and *mutation* are applied on these strings to yield a new generation of strings. The process of selection, crossover and mutation continues for a fixed number of generations or till a termination condition is satisfied [8].Extensive studies dealing with comparative analysis of different clustering methods [16] suggest that there is no general strategy which works equally well in different problem domains. However, it has been found that it is usually beneficial to run schemes that are simpler, and execute them several times, rather than using schemes that are very complex but need to be run only once [16]. Since our aim is to propose a clustering technique based on GAs, a criterion is required whose optimization would provide the final clusters.

2.1. Firefly algorithm

The firefly algorithm (FA) is a metaheuristic algorithm, inspired by the flashing behaviour of fireflies. The primary purpose for a firefly's flash is to act as a signal system to attract other fireflies. [17]

a. All fireflies are unisexual, so that one firefly will be attracted to all other fireflies regardless of their sex.

b. Attractiveness is proportional to their brightness, and for any two fireflies, the less bright one will be attracted by the brighter one.

c. If there are no fireflies brighter than a given firefly, it will move randomly.

The brightness should be associated with the objective function. Assume that there exists a swarm of n agents (fireflies) and Xi represents a solution for a firefly i, whereas f (Xi) denotes its fitness value [18]. Here the brightness I of a firefly is selected to reflect its current position x of its fitness value f (x).

$I_i = f(X_i),\ 1 \le i \le n.$

(1)

Firefly attractiveness is proportional to the light intensity seen by adjacent fireflies. Each firefly has its distinctive attractiveness β which implies how strong it attracts other members of the swarm. However, the attractiveness β is relative, it will vary with the distance r_{ij} between two fireflies i and j at locations x_i and x_j respectively, is given as

$r_{ij} = \|x_i - xj\|$.

(2)

The degree of attractiveness of a firefly is determined by

$\beta(r) = \beta_0 e^{-\gamma r2}$

(3)

where β_0 is the attractiveness at r = 0 and γ is the light absorption coefficient at the source.

The movement of a firefly i at location x_i attracted to another more attractive (brighter) firefly j at location x_j is determined by

$x_i(t + 1) = xi(t) + \beta_0 e^{-\gamma r2}(x_j - x_i)$.

(4)

2.2. Genetic algorithm

Perhaps the most referenced of the evolutionary algorithms is the genetic algorithm. The Algorithms mimics the natural evolution proposed by Charles Darwin. The basic steps of operation in a genetic algorithm are initialization, selection, crossover and mutation, termination. The algorithm is initiated by generating a random population of chromosomes, in the next step the fitness is evaluated for each of the chromosomes based on the function under evaluation. Based on the fitness value the best chromosomes are selected to reproduce the next generation of chromosomes by the crossover and mutation process. The current population is then replaced by the new population that is generated in the previous steps and the process of calculating the fitness function for the new population is repeated. The algorithm is iterated until a predetermined fitness value is reached or for prefixed number of generations or iterations has been reached.GA has strong global search capability. In the search process, it reoriented the search space to keep the algorithm to find global optimal solutions or prospective global optimal solution easily.

2.3. Optimization using Multi core

Multi-core processors can deliver significant performance benefits for multi-threaded software by adding processing power with minimal latency, given the proximity of the processors. Multicore processors are now widespread across server, desktop, and laptop hardware. They are also making their way into smaller devices, such as smartphones and tablets. They open new possibilities for concurrent programming because the threads of a process can be executed on several cores in parallel. One important technique for achieving maximal performance in applications is the ability to split intensive tasks into chunks that can be performed in parallel to maximize the use of computational power. When designing software to run on a multi-core or multiprocessor system, the main consideration is how to allocate the work that will be done on the available processors. The most common way to allocate this work is by using a threading model where the work can be broken down to separate execution units that can run on different processors in parallel. If the threads are completely independent of one another, their design does not have to consider how they will interact. For example, two programs running on a system as separate processes each on its own core do not have any awareness of each other. Performance of the programs is not affected unless they contend for a shared resource such as system memory or the same I/O device. The Fork/Join Framework enhances multithreaded programming in two important ways. First, it simplifies the creation and use of multiple threads. Second, it automatically makes use of multiple processors.[5]

III. Proposed Work

3.1. Proposed Algorithm

The aim of the proposed method is to cluster a large dataset efficiently. Here a scalable parallel clustering algorithm is used to overcome the problem in clustering large dataset with high dimension. Here we used two different clustering methods. The first clustering method is the Firefly based clustering algorithm which is applied to the each randomly divided set of input data. The second is a genetic algorithm based clustering method called genetic algorithm which is applied to the merged cluster centroid data obtained. Then finally the resultant cluster is obtained at the output. Merging is the process of grouping the distributed group of clusters which is an important process in our proposed method. Also the important purpose of this merging process is to get the efficient data from all the clusters. The proposed approach initiate a genetic algorithm based clustering for obtaining the merging node.

3.2. Genetic firefly algorithm clustering

We developed a new method for clustering the merge data by combining both the Genetic algorithm and firefly algorithm. In general, the genetic firefly base clustering algorithm can reach the optimum point of the function very quickly. In the first step of our genetic firefly method each firefly is randomly generated. For each firefly two data from the input data are taken as initial population and forms the two centroids for the respective firefly. Then the distance is computed between the data in the firefly and the whole data in the data set. Likewise for each data in the firefly the distance between the centroid and the whole data is computed. Finally the minimum distance among the centroids in each firefly is found out which is the fitness for each firefly. Finally for each fitness of the firefly Davies Bouldin index (DBI) is calculated. Then two worst solutions are taken and performed the crossover and mutation process. These steps are repeated until stopping criteria is met.

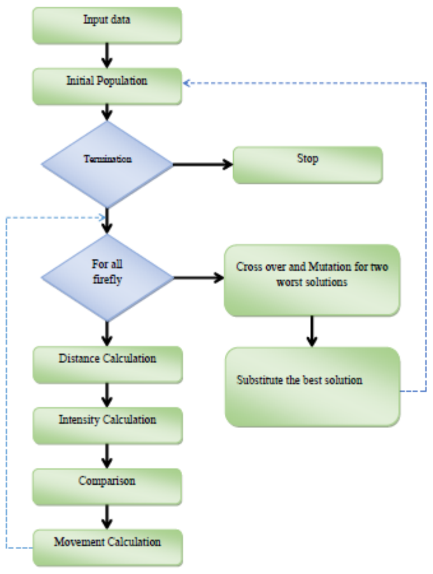

Fig. 1. General Block Diagram of Proposed Technique

3.2.1. Initial population for firefly

The initial population of firefly is created as $F = F_1, F_2, F_3, ...F_N$. Then for each firefly the two input data d_1 and d_2 are taken and has two centroids each C_1^1 *and* C_2^1. Then for the selected data the DBI is calculated to find the minimum distance which calculates the new solution. Initially the minimum distance for each firefly of the population is determined based on the below process. Firstly, the distance between the data in firefly and the whole data in the dataset are computed. The randomly taken centroid data forms the data in the firefly represented by C_i^j, *where* $0 < i \leq 2$ and $0 < j \leq N$.

For the first firefly, two centroids are considered and are represented by C_1^1 *and* C_2^1 . The whole dataset is represented by d_i, *where* $0 < i \leq M$ and M is the total number of data in the dataset. The term $C_i^1 - d_j$ represents the distance between the ith centroid and jth data in the dataset. The distance computation is made by the formula given by:

$$dis = \sqrt{\sum_{n=1}^{M} (z_{in} - z_{jn})^2}$$

Where, z_{in} is the nth attribute of the ith data and z_{jn} is the nth attribute of the jth data.

Once the distance computation is carried out, find out the minimum distance value in each row. That is the minimum distance among the centroids in the firefly with respect to data from the dataset is found out. For example, minimum distance value between the centroids in the first firefly and the first data from the dataset is found using the formula:

$$D\min_1^{\,1} = \begin{cases} C_1^1 - d_1; & if[(C_1^1 - d_1) < (C_2^1 - d_1)] \\ C_2^1 - d_1; & if[(C_2^1 - d_1) < (C_1^1 - d_1)] \\ C_3^1 - d_1; & if[(C_3^1 - d_1) < (C_1^1 - d_1)] \end{cases}$$

Generalizing, minimum distance value between the centroids in the first firefly and the jthdata from the dataset is found using the formula:

$$D\min_j^1 = \begin{cases} C_1^1 - d_j; & if[(C_1^1 - d_j) < (C_2^1 - d_j)] \\ C_2^1 - d_j; & if[(C_2^1 - d_j) < (C_1^1 - d_j)] \\ C_3^1 - d_j; & if[(C_3^1 - d_j) < (C_1^1 - d_j)] \end{cases}$$

Then the DB Index is calculated for the minimum distance D_{min} of each firefly.

3.2.2. DB Index calculation

In this section the DBI for the selected data of each firefly is calculated. The representation of DBI for N number of clusters is given below,

$$DB \equiv \frac{1}{N}\sum_{i=1}^{N} D_i$$

The DB index is completely depends on both the data as well as the algorithm. In the above equation the D_i choose the worst case scenario, and this value is equal to $R_{i,j}$ for the most similar to similar cluster i.

The symmetry condition for D_i is given below,

$$D_i \equiv \max_{j:i \neq j} R_{i,j} \qquad \text{where} \quad R_{i,j} = \frac{S_i + S_j}{M_{i,j}}$$

Here, $R_{i,j}$ is a measure of how good the cluster scheme. This measure, by definition has to account for $M_{i,j}$ the separation between i^{th} and j^{th} clusters, which ideally has to be as large as possible, and S_i the within cluster scatter for cluster i which has to be as low as possible. The DB index calculation has various properties such a,

(i) $R_{i,j} \geq 0$

(ii) $R_{i,j} = R_{j,i}$

(iii)If $S_j \geq S_k$ and $M_{i,j} = M_{i,k}$ then $R_{i,j} = R_{i,k}$ and

(iv)If $S_j = S_k$ and $M_{i,j} \leq M_{i,k}$ then $R_{i,j} > R_{i,k}$

Let C_i be a cluster of vectors. Let X_j be an n dimensional feature vector assigned to cluster C_i.

The within cluster scatter S_i for cluster I is represented as

$$S_i = \left(\frac{1}{T_i} \sum_{j=1}^{T_i} \left| X_j - A_i \right|^q \right)^{1/q}$$

Here A_i is the centroid of C_i and T_i is the size of the cluster $_i$. Usually the value of q is 2, which makes this a Euclidian distance function between the centroid of the cluster, and the individual feature vectors. For meaning full result the distance metrics has to match with the metrics used in the clustering.

$$M_{i,j} = \left\| A_i - A_j \right\|_p = \left(\sum_{k=1}^{n} \left| a_{k,i} - a_{k,j} \right|^p \right)^{\frac{1}{p}}$$

Where, $M_{i,j}$ is a measure of separation between cluster C_i and the cluster C_j. $a_{k,i}$ is the kth element of A_i and there are n such elements in A for it is an n dimensional centroid. Hence, the DB Index of all fireflies are computed which is subsequently given for movement computation.

3.2.3. Movement Computations

The movement computation is calculated based on the calculated DB Index value. Subsequently, the movement computation is made on the firefly that has high DB Index value. For example, compare the first firefly having DB Index value D_1 and the second firefly having DB Index value D_2 , then if $D_2 > D_1$, that is if the second firefly has better DB Index value, then apply the movement calculation in the second firefly and substitute the data in the first firefly by the solution acquired after movement computation. Similarly, if $D_1 > D_2$, then substitute the data in the second firefly by the solution obtained after movement computation. Movement computation is out by the formula:

$$z_i = z_i + \beta_0 e^{-\gamma r_{ij}^2} + \alpha\left(random - \frac{1}{2} \right)$$

where, z_i is the attribute data value of firefly having higher DB Index and β_0, γ and α are constants. *randomn* represents a value that hold the values in range of 0 to 1 and r_{ij} is the difference between z_i and z_j. The process of comparison is carried out for all firefly and movement computation and substitution is made accordingly.

3.2.4. Crossover and Mutation

In this crossover two parent solutions are used for crossover. By performing the cross over operation on the selected parent solutions two new offspring solutions are generated. Each off spring solution has a part of their parent solutions. Here the cross over point is set to be as one. After the crossover we attain alternative n number of new children clusters. Then the mutation is applied over the children of each parent clusters in order to make diversity among the solutions. So, the parent solutions through the help of genetic operators are able to generate offsprings that are superior to their parents. After crossover and mutation, a completely new population of clusters is obtained. Then the fitness value of new population is calculated. After that, the population of clusters with maximum fitness value is selected as the new parent clusters. These steps are repetitive till it reaches the maximum iteration limit. After the termination criteria, we have got new set of clusters that represents the better clusters.

The implementation model of our approach adopts the form of Fork-Join (Fork means to create a new thread or awake the existing thread, Join means the joint of multi-thread).If the master thread is required for parallel computing while it is running, it may produce derived thread to execute parallel task, in which the master thread work together with the derived thread. When the executing of parallel task is finished, the derived thread will be quit or hung, and the control of the flow returns to the separate master thread.
Fork/Join Framework adds two main classes to the java.util.concurrent package:

- ForkJoinPool
- ForkJoinTask

The execution of ForkJoinTask takes place within a ForkJoinPool, which manages the execution of the tasks. ForkJoinTask objects support the creation of subtasks and waiting for the subtasks to complete. Advantage of the ForkJoinPool, is that it can *'steal'* work. It means that it allows one thread that has finished a task to immediately execute another task with much less overhead than the ExecutorService. [19].
The RecursiveAction and RecursiveTask are the only two direct, known subclasses of ForkJoinTask. The only difference between these two classes is that the RecursiveAction does not return a value while RecursiveTask does have a return value and returns an object of specified type.
ForkJoinTask objects feature two specific methods:
The fork () method allows a new ForkJoinTask to be launched from an existing one. In turn, the join () method allows a ForkJoinTask to wait for the completion of another one. A ForkJoinTask instance is very light weight when compared to a normal Java thread. [5]
Following code can be used to perform parallel operations.

```
Public class FJoin extends RecursiveTask<Integer>
{
public static ArrayList<ArrayList<String>> input=new ArrayList<ArrayList<String>();
private static final int size=130;
public FJoin(int data,int start,int end)
{
this.data=data;
this.start=start;
this.end=end;
}
public FJoin(int data)
{
this(data,0,data);
}
@override
protected Integer compute()
{
final int length=end-start;
if(length<size)
{
return computeDirectly();
}
final int split=length/2;
final FJoin first=new FJoin(data,start,start+split);
first.fork();
final FJoin second=new FJoin(data,start+split,end);
return Math.max(second.compute(),first.join());
}
```

Fork/join parallelism is implemented by means of a fixed pool of worker threads. We set the number of worker threads in the fork/join pool to a maximum of four, which is the number of cores in a node in our system. Each worker thread can execute one task at a time. Tasks waiting to be executed are stored in a queue, which is owned by a particular worker thread. Currently executing tasks can dynamically generate (i.e. fork) new tasks, which are then enqueued for subsequent execution

IV. EXPERIMENTAL RESULTS

We implemented the proposed algorithm using JAVA language. The code is executed on dell Inspiron N4030 Laptop, Intel(R) Core(TM) i5 Processor 2.67 GHz, 2 MB cache memory, 3GB RAM, 64-bit Windows 7 Home and NetBeans IDE 8.0.

4.1 Dataset Description

In our proposed method two kinds of datasets are used for evaluation. The first dataset is the skin segmentation dataset taken from UCI machine learning repository. It consists of totally 245057 instances and 4 attributes. It is constructed over B, G and R colour space. Skin and Non-skin dataset is generated using skin textures from face images of diversity of age, gender, and race people. Also it collected by sampling B,G,R values from face images of various age groups such as young, middle, and old, race groups such as white, black, and asian, and genders obtained from FERET database and PAL database. Total learning sample size is 245057.Among this 50859 is the skin samples and 194198 is non-skin samples. Colour FERET Image Database, PAL Face Database from Productive Aging Laboratory. The University of Texas at Dallas. The information about the attribute is as below. This skin segment data set is of the dimension 245057 * 4 where first three columns are B, G, R values and fourth column is of the class labels [20].

The next dataset is the poker hand dataset taken from the UCI machine learning repository. The poker hand data set is multivariate characteristics which consist of 1025010 instances and 11 numbers of attributes. The dataset description is each record is an example of a hand consisting of five playing cards drawn from a standard deck of 52. Each card is described using two attributes, for a total of 10 predictive attributes. [21].

4.2. Evaluation metrics

The proposed scalable parallel clustering algorithm uses mainly two evaluation matrices, the clustering accuracy and computation time. The clustering accuracy is measured by counting the number of correctly assigned documents and dividing by N, which is given the equation below

$$Accuracy = \frac{1}{N}\sum_k \max_j |B_k \cap C_j|$$

where, B_k is the set of clusters $B = \{B_1, B_2, B_3..........B_k\}$and C_j is the set of classes $C = \{C_1, C_2.C_3,......C_k\}$.We interpret B_k as the set of documents in B_k and C_j as the set of documents in C_j.

The other metrics used here is the computation time which is measured based on the starting and ending time of the program.

4.3. Performance based on the final resultant cluster

The initial analysis is based on the skin dataset with respect to the proposed approach. The skin dataset is initially selected for the analysis. The analysis conducted for varying parameters data size in the parallel and initial number of clusters in the system in the following section, the accuracy and computation time based on final cluster is subjected.

a) Accuracy and computation time based on final cluster

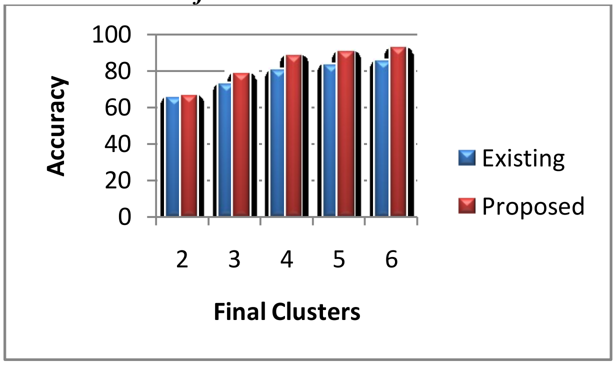

Fig.2: Accuracy of skin data set on final cluster

The existing method we used here to compare the performance of the proposed method is k-means and genetic. In the above figure, we have presented the accuracy of the proposed approach by varying the final clusters of the parallel clustering system. The final clusters are varied from 2 to 6. The analysis from the figure shows that the proposed approach has upper hand over the existing method. In the case of proposed approach, the accuracy increases as the number of final cluster increases. The maximum accuracy attained by the proposed approach is 90%, which is very better figure as compared to the existing method, for which the maximum accuracy obtained is 70%. All these experiments are conducted in same environment.

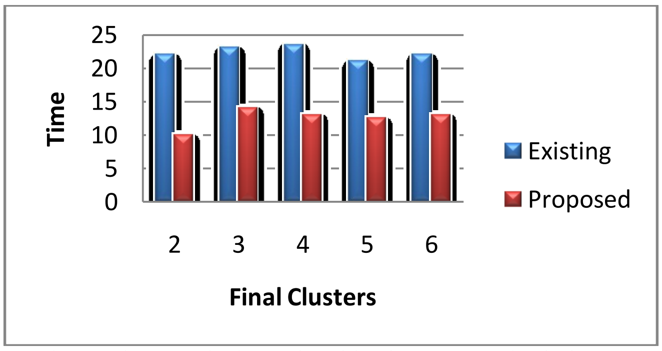

Fig.3: Time analysis of the skin data set on final cluster

The above figure represents the time analysis of the proposed approach based on varying number of clusters. The response regarding time for execution is different from that of accuracy analysis. The response is clearly irregular in manner for all the cases. Both the proposed approach and existing approach is clearly different from each other. As considering all scenarios, we assess that the proposed approach is efficient in the time analysis as compared to the existing approach.

b) Accuracy and computation time based on data size

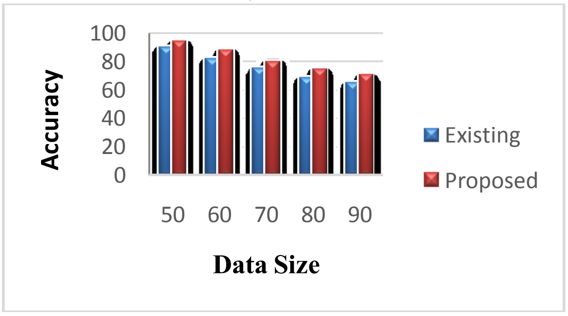

Fig.4: Accuracy and computation time based on data size in the skin data set

In the above figure, we have presented the accuracy of the proposed based on the data size of the parallel clustering system. The data size is varied from 50 to 90. For each data size the accuracy is plotted. From this we analysis the proposed approach has upper hand over the existing method. The maximum accuracy attained by the proposed approach is 93%, which is very better figure as compared to the existing method, for which the maximum accuracy obtained is 90%. But, in the case of proposed approach, the accuracy decreases as the data size increase.

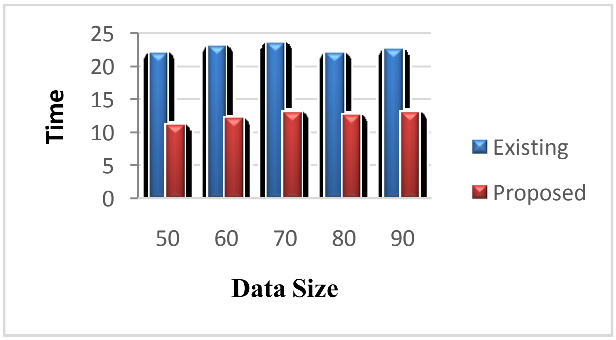

Fig.5: Time analysis of the skin data set based on data size

The time analysis based on the data size is shown in the above figure. From this analysis we found that the proposed approach and existing approach is clearly different from each other. As considering all scenarios, we assess that the proposed approach is efficient in the time analysis as compared to the existing approach.

4.4. Performance evaluation based on poker hand dataset

The analysis is based on the poker hand dataset with respect to the proposed approach. The poker hand dataset is initially selected for the analysis. The analysis conducted for varying parameters data size in the parallel and initial number of clusters in the system. In the following section, the accuracy and computation time based on final cluster is subjected.

a) Accuracy and computation time based on final cluster

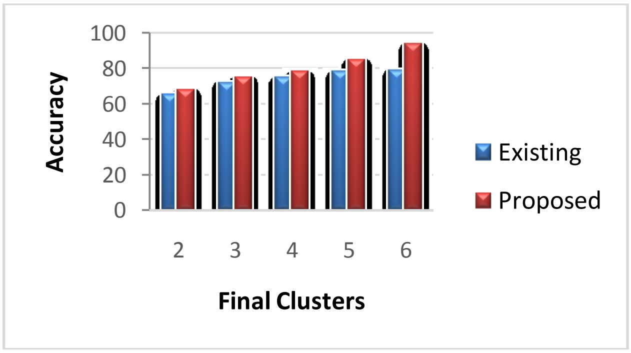

Fig.6: Accuracy of poker hand data set on final cluster

In the above figure, we have presented the accuracy of the proposed approach by varying the final clusters of the parallel clustering system in poker hand dataset. The final clusters are varied from 2 to 6. The analysis from the figure shows that the proposed approach has upper hand over the existing method. In the case of proposed approach, the accuracy attained by the proposed approach is very better as compared to the existing method.

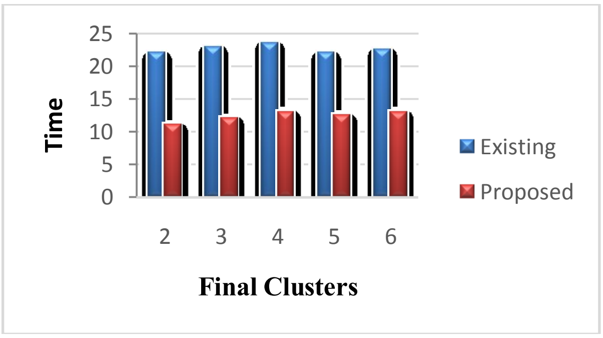

Fig.7: Time analysis of the poker hand data set based on final clusters

The above figure represents the time analysis of the proposed approach based on varying number of clusters in poker hand dataset. The response regarding time for execution is different from that of accuracy analysis. Both the proposed approach and existing approach is clearly different from each other. As considering all scenarios, we assess that the proposed approach is efficient in the time analysis as compared to the existing approach

b) Accuracy and computation time based on data size

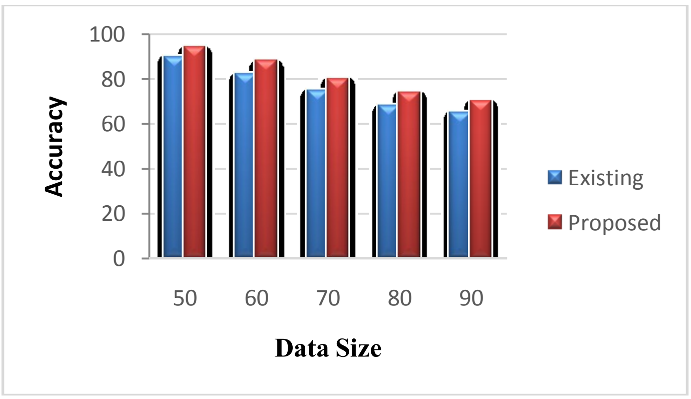

Fig.8: Accuracy and computation time based on data size in the poker hand data set

The accuracy plotted based on the varying data size from 50 to 90. For each data size the accuracy is plotted. Here the accuracy of our proposed method increased with increasing data size which shows that our proposed provide better accuracy than the existing method. Similarly the time analysis plot based on the varying data size is shown in the figure below.

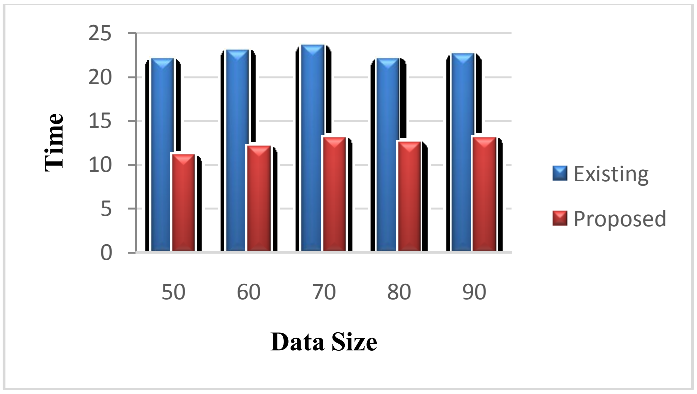

Fig.9: Time analysis of the poker hand data set based on data size

From this analysis we found that the proposed approach and existing approach is clearly different from each other. As considering all scenarios, we assess that the proposed approach is efficient in the time analysis as compared to the existing approach.

V. Conclusion

Large data clustering plays crucial and related processes in various domains. However, most of the clustering algorithms are compatible only with small data. The solution to address large scale clustering problem is exploiting parallel algorithm. Finally, based on the assumption, in this paper, we have presented a method for scalable parallel clustering based on series of methods. The proposed approach is designed to address mainly for the difficulty to cluster large data bases. A hybrid genetic firefly clustering method is applied to merged clusters. Finally the analysis was made by two types of datasets the skin and the poker hand data set from UCI machine learning repository. Our proposed method is compared with the performance of the existing genetic and k-means clustering algorithm. We have used fork and join model for the Java programming concurrently on multi-cores machine. Fork/join method overcomes deficiencies of multithreaded execution. The performance analysis and experimental result showed that our proposed method provide better result. Also the experimental analysis showed that the proposed approach obtained upper head over existing method in terms of accuracy and time. The highest accuracy achieved by the proposed approach is 92%.

References

[1] X. S. Yang, “Nature-Inspired metaheuristic Algorithms”. Luniver Press, 2008.

[2] X. S. Yang, “Firefly algorithm‹ stochastic Test Functions and Design optimization”.Int. J. bio-inspired computation .2010.

[3] X. S. Yang, “Firefly algorithm for multimodal optimization.” In: Stochastic Algorithms: foundations and applications ‹SAGA ‹lecture notes in computer sciences, pp. 169-178, 2009.

[4] S. M. Elsayed, R. A. Sarker, and D. L. Essam, “A new genetic algorithm for solving optimization problems,” Engineering Applications of Artificial Intelligence, vol. 27, pp. 57-69, 2014.

[5]. Doug Lea, A Java Fork/Join Framework, State University of New York at Oswego,www.developer.com

[6] Kennedy J., Eberhart R., *Particle Swarm Optimization*, Proceedings of IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.

[7] Kwee L., Yew-Soon O., Meng L., Xianshun C., Agarwal A., *Hybrid ant colony algorithms for path planning in sparse graphs*, Soft Computing, Vol. 12, Issue 10, 2008, pp. 981- 994.

[8] D. Karaboga, *An idea based on honey bee swarm for numerical optimization*, Technical Report TR06, Computer Engineering, Department, Erciyes University, Turkey, 2005.

[9] Jiann-Horng L., Li-Ren H., *Chaotic bee swarm optimization algorithm for path planning of mobile robots*, Proceedings of the 10th WSEAS international conference on evolutionary computing , 2009, pp. 84-89.

[10] Yang, X. S. and Deb, S., *Cuckoo search via Lévy flights*, in: Proc. of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), 2009, pp. 210-214.

[11] Lukasik S., Zak S., *Firefly Algorithm for Continuous Constrained Optimization Tasks* Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent, LNCS, Vol. 5796, 2009, pp. 97–106.

[12] Krishnanand K., Ghose D., *Glowworm swarm based optimization algorithm for multimodal optimization functions with collective robotics applications*, Multiagent and Grid Systems, Vol. 2, Issue 3, 2006, pp.209–222.

[13] Cecilia R., Tenreiro Machado J. A., *Crossing Genetic and Swarm Intelligence Algorithms to Generate Logic Circuits*, WSEAS Transactions on computers, Vol. 8, Issue 9, 2009, pp. 1419- 1428.

[14] Zhuang X., Mastorakis N.E., *Edge Detection Based on the Collective Intelligence of Artificial Swarms*, in: Proceedings of the 4th WSEAS International Conference on Electronic, Signal Processing and Control, 2005, pp. 25-27.

[15] d. Mostofa Ali Patwary,Diana Palsetia1, Ankit Agrawal1,Wei-keng Liao1, Fredrik Manne2, AlokChoudhary, " Scalable Parallel OPTICS Data Clustering Using Graph Algorithmic Techniques", International Conference for High Performance Computing, Networking, Storage and Analysis, ACM, No. 49, 2013

[16] R.C. Dubes, A.K. Jain, Clustering techniques: the user's dilemma, Pattern Recognition 8 (1976) 247}260.

[17] M. R. Senapati. "Local linear wavelet neural network based breast tumor classification using firefly algorithm", Neural Computing and Applications, 2012

[18]. J. Senthilnath, S.N. Omkar, V. Mani, Clustering using firefly algorithm: Performance study, Swarm and Evolutionary Computation 1 (2011) 164–171
[19]. Robert D Blumofe, The University of Texas at Austin, Scheduling Multithreaded Computations by Work stealing
[20] Data set from UCI machine learning repository, "https://archive.ics.uci.edu/ml/datasets/Skin+Segmentation#".
[21] Data set from UCI machine learning repository "https://archive.ics.uci.edu/ml/datasets/Poker+Hand".