Biclustering using Parallel Fuzzy Approach for Analysis of Microarray Gene Expression Data

Dwitiya Tyagi-Tiwari *dwitiya.sr@gmail.com*
Department of Mathematics & Computer Applications
Maulana Azad National Institute of Technology
Bhopal-462051, India

Sujoy Das *sujdas@gmail.com*
Department of Mathematics & Computer Applications
Maulana Azad National Institute of Technology
Bhopal-462051, India

Manoj Jha *m-jha28@rediffmail.com*
Department of Mathematics & Computer Applications
Maulana Azad National Institute of Technology
Bhopal-462051, India

Namita Srivastava *sri.namita@gmail.com*
Department of Mathematics & Computer Applications
Maulana Azad National Institute of Technology
Bhopal-462051, India

Abstract

Biclusters are required to analyzing gene expression patterns of genes comparing rows in expression profiles and analyzing expression profiles of samples by comparing columns in gene expression matrix. In the process of biclustering we need to cluster genes and samples. The algorithm presented in this paper is based upon the two-way clustering approach in which the genes and samples are clustered using parallel fuzzy C-means clustering using message passing interface, we call it MFCM. MFCM applied for clustering on genes and samples which maximize membership function values of the data set. It is a parallelized rework of a parallel fuzzy two-way clustering algorithm for microarray gene expression data [9], to study the efficiency and parallelization improvement of the algorithm. The algorithm uses gene entropy measure to filter the clustered data to find biclusters. The method is able to get highly correlated biclusters of the gene expression dataset.

We have implemented the algorithm of fuzzy c-means in MATLAB parallel computing platform using MATLABMPI (Message Passing Version of MATLAB). This approach is used to find biclusters of gene expression matrices. The biclustering method is also parallelized to reduce the gene centers with lower entropy filter function. By this function we choose the gene cluster centers with minimum entropy. The algorithm is tested on well-known cell cycle of the budding yeast S. cerevisiae by Cho et al. and Tavazoi et.al data sets, breast cancer subtypes Basal A, Basal B and Leukemia from Golub et al.

Keywords: Biclustering Analysis, Gene Expression, Parallel Computing Toolbox, Fuzzy, MATLABMPI.

1. INTRODUCTION

Large amount of gene expression data demands the need for methods that are effective in analyzing informations that are present in it. One of the primary data analysis tasks is to cluster data which intended to help a user to understand the natural grouping or structure of dataset.

Therefore, the development of improved clustering algorithms have received direct attention. Clustering of genes/samples according to their expression values is an important approach in extracting knowledge from microarray gene expression data [9, 17]. Gene expression data is arranged in a matrix form, where rows are represented as genes and columns are represented the conditions or samples. The main goal of a clustering algorithm is to group the data objects of into a set of meaningful subclasses. A clustering algorithm is often applied on microarray data to find the similar group of genes or samples. These similar expression patterns suggest the co-regulation of the genes or samples. Co-regulation of the genes or samples may possibly be involved in a similar biological functions [7]. Therefore in this paper we have used fuzzy c-means clustering method that can assign single gene/sample to several groups. Further FCM clustering is applied because microarray gene expression data contains noisy components due to biological and experimental factors. Sometimes the gene activity shows large variations under smaller changes of the experimental conditions [9,14]. Clustering algorithms generally follows hierarchical or partitional approaches [15] like k-means, fuzzy c-means algorithm [14], hierarchical clustering etc. However, these clustering approaches used to form clusters in one-way only, i.e. clusters of either gene or sample. Hartigen in 1972 [10] first applied two way clustering or biclustering to describe simultaneous grouping of both row and column subsets in a data matrix. Cheng and Church [19] has used two-way clustering approach for analyze gene expression data. These two-way clustering is called biclusters which are particularly valuable or demanding to mine important information from large databases in both rows and columns dimensions. The literature says that biclustering is NP-hard problem [17], and it takes long time to compute the biclusters, so, it is important to solve the problem in timely manner. In this regard parallel computing technology are the essential solution to minimize execution time of biclustering approach. Parallel approach has also been developed to solve the problem of biclustering in timely manner [2][8][14] Liu and Chen [13] have implemented a parallel algorithm for biclustering of gene expression data is based on anti-monotones property of the quality of data sizes. Parallel identification of gene based biclustering with coherent evolution is identified based on additive modelling by [2].

In this paper, we present fuzzy c-means algorithm for clustering gene and sample dimensions for biclustering using message passing model in MATLAB environment. Message passing interface (MPI) is the real standard for implementing programs on multiple processors in parallel computing. Parallel programs are defines in C, C++ and FORTRAN language functions for doing point to point communication with MPI [1]. We have used MatlabMPI to implement the parallel algorithms. MatlabMPI can run on any combination of computers in which MATLAB can run. Another advantage of using MatlabMPI is that it provides the facility to use MATLAB implicit functions and toolbox [21]. This approach is based upon the previous work of [8]. It is a parallelized rework of a parallel fuzzy two way clustering algorithm for microarray gene expression data, to study the efficiency and parallelization improvement of the algorithm. The outline of this paper is as follows: section 2 gives an overview of parallel biclustering algorithm using fuzzy approach with MatlabMPI; Section 3 describe parallel fuzzy two-way clustering algorithm and discusses parallelization aspects of algorithm; Section 4 presents experimental results; Section 5 discuses conclusion and future work.

2. MATERIALS AND METHODS

2.1 Fuzzy c-means Algorithm

Fuzzy c-means algorithm is originally developed by Bezdek 1981[3], which allows one data point to be present two or more clusters. The algorithm assigns membership values to each data point corresponding to each cluster center on the basis of distance between data point and cluster. Summation of membership values of each data point should be equal to one. The original algorithm is based on minimization of the following objective function:

$$J_m = \sum_{k=1}^{K} \sum_{i=0}^{N} (u_{ki})^m \, \|x_i - c_k\|^2 \qquad \ldots\ldots\ldots\ldots\ldots (1)$$

Where *K* is number of cluster and *N* is number of data objects, *m* is constant real-valued number which controls the 'fuzziness', u_{ki} is the degree of membership of data object x_i in cluster *k*.

The steps are as follows [3]:

1. **Inputs**: select K i.e. number of cluster and randomly select initial centroids of each cluster.
2. Initialize $U = [u_{ki}]$, $k = 1, \ldots, K$ and $i = 1, \ldots, N$.
   Calculate membership values using formula as given by Bezdek [3]:
   $$u_{ki} = \left\{ \frac{1}{\sum_{s=1}^{N} \left[\frac{\|x_i - c_k\|}{\|x_i - c_s\|} \right]^{\frac{2}{(m-1)}}} \right. \quad (2)$$
3. At *l* step compute the matrix of centroids $C^{(l)} = c_k$:
   $$c_k = \frac{\sum_{i=1}^{N} (u_{ki})^m x_i}{\sum_{i=1}^{N} (u_{ki})^m} \text{ k=1,2-------K} \quad (3)$$
4. Update U^l to U^{l+1} by Equation 2.
5. If $U^l = U^{l+1}$ the minimum J is achieved, then terminate; otherwise return to Step 2.

2.2 Message Passing Interface

High performance parallel programming environment uses message passing to communicate and exchanging the data among the processors. MPI is a language independent communications protocol which provide high performance scalability and portability [1][25]. MPICH is a standard for message passing used in parallel computing. MATLAB provides the facility of parallelism through MatlabMPI. MatlabMPI developed at Lincoln Laboratory at Massachusetts Institute of Technology (MIT) designed by Dr. Jeremy Kepner, which is a set of MATLAB scripts or programs that implements a subset of MPI [13]. It allows MATLAB program to be run on a parallel computer. MatlabMPI will run on any combination of computers that MATLAB supports [13][24]. Table1 shows six basic MPI function of MatlabMPI:

TABLE 1: MATLABMPI Uses Six Basic MPI Functions.

MPI Functions	Function Description
MPI_Init	To initialize MPI
MPI_Comm_rank	Access Id current processor within a communicator
MPI_Comm_size	Access the number of processors in a communication
MPI_Send	Sends the data or message to processor
MPI_Recv	Receives the data or message from a processor
MPI_Finalize	To terminate or Finalizes MPI

All of the MPI programs establish the communication between the concurrent execution parts. It initializes and starts with MPI-Init(), then establish the processes and new communicator functions and applications to be used for each process, and in last it terminates with MPI_Finalize(). The flow of parallel program design with message passing is shown in figure 1 [25].

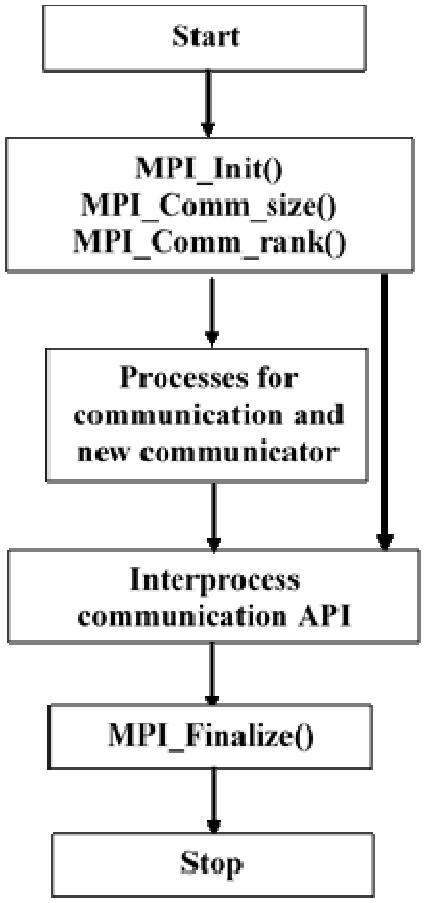

FIGURE 1: Message Passing Interface Process Flow.

MatlabMPI is also used basic MPI function to write a script on MATLAB these are defined in Table1. The basic steps for executing MatlabMPI is given in [13][24]. These steps are included as follows [13]:

1. To set the path for MatlabMPI source in MATLAB current directory.
2. Initialize MPI library.
3. Create the communicator.
4. Define the size and rank of the communicator.
5. Computation using MATLAB.
6. Communication between the processes using MPI for exchanging data.
7. Terminate MPI application.

2.3 Proposed Parallel Fuzzy c-means Algorithm

The fuzzy clustering method assigns one gene/sample to multiple clusters according to their membership values. So that this method is more appropriate for analyzing gene expression profiles of gene expression data, because a single gene might be involved in multiple functions. Main objective of this algorithm is to improve the performance of fuzzy c-means approach for biclustering of gene expression data by distributing computation and main memory usage. The algorithm is based on data parallelism and communicated between the processes using MatlabMPI [13] with SPMD (Single Program Multiple Data) parallel computing model for data distribution.

Based on above original algorithm we have implemented parallel version of FCM using MATLABMPI, as shown in figure 1. The parallel algorithm divides the *N* data points in the *M*

processors. Here master processor is MATLAB client and MATLAB processors are participating processors. The proposed parallel fuzzy c-means algorithm is as follows:

```
Input: number of clusters K, number of data matrix N.
Output: K centroids. Initialize random cluster centroids;
        Distribute the N data matrix among all processors in N/P.

1.  MPI_Init();  %start the procedure
2.  M = MPI_Comm_size();
3.  Id = MPI_Comm_rank();
4.  Initialize the random cluster centroid for each data point in fuzzy cluster
    K;
5.  Max-itr = 0;
6.  while ((Max_itr == 100) || (Old_ Mem_fcn[j i] == Mem_fcn[i j]))
7.  For i = 1 : K
8.  Mat_of_centr[i j] = 0;  % stores matrix of centroids value Eq2.
9.  Val_centr[i j] = 0;  %  stores the value of cluster centroid
10. Mem_fcn[i j] = 0;  %  membership function for next iteration Eq3
11. end;   %  end for
12. for i = id * (N/M) + 1 : (Id + 1) * (N/M)
13. for j = 1 : K
14. Update Mat_of_centr[K];
15. Update Val_centr[K];
16. end;   % End for j
17. end;    %  End for i
18. for j = 1 : K
19. MPI_reduce(Mat_of_centr[j], Mat_centrP [j], MPI_SUM); %
    Mat_centrP- local matrix of centroids.
20. MPI_reduce(Val_centr[j], Val_centrP[j], MPI_SUM); %
    Val_centrP- local values of centroids.
21. Update centroid vectors:
    Val_centrP[j] = Val_centrP[j] / Mat_centrP [j];
22. end; %  End for j
23. for i = id * (N/M) + 1 : (Id + 1) * (N/M)
24. for j = 1 : K
25. Update Mem_fcn[j i];
26. Old_ Mem_fcn[j i] = Mem_fcn[i j];
27. Max_itr = Max_itr + 1;
28. end; %  End for i
29. end;  % End for j
30. MPI_reduce(max_itr, itr, MPI_MAX);
31. end;  % End While.
32. MPI_Finalize();
```

FIGURE 2: Algorithm1- Fuzzy C-means clustering procedure with MATLABMPI.

The algorithm is modified by implementing Fuzzy C-means at Step1 and 2 of Tang et al. [5] and has further implemented on MATLAB using Message Passing Interface. Gene reduction is done using gene entropy instead of correlation coefficient.

Main algorithm steps for parallel interrelated two-way clustering procedure are as follows:

1. Genes clustering: Parallel Fuzzy C-means algorithm is given in the previous section is used to cluster the genes using algorithm1 of figure 2.

2. Sample clustering: Parallel Fuzzy C-means algorithm is given in the previous section is used to cluster the samples algorithm1 of Figure 2.
   In Step 1 and 2 data points are divided into *K* groups which take the number of clusters as an input parameter. Here we take *K* = 4 and 6.

3. Find gene centers: Maximize the sum of distances between the genes having maximum membership value from membership matrix.

4. Gene centers reduction by gene entropy filter: Apply low gene entropy filter function on gene cluster centers found in Step2 and remove two gene cluster centers and reduce the genes of same cluster centers.

   This process is also parallelized to give one cluster to each processor to calculate gene entropy and give result back to master processor. Clients find the lower gene entropy and remove.

   Entropy is a function of correlation which provides the amount of information that may be gained by an observation of a system and it measures variation or changes in a series of events. Entropy denotes the diversity of information in given data set which makes it suitable for clustering genes. We calculate the low gene entropy of given gene expression datasets. For the measurement of interdependency of two random genes X and Y we have used a direct MATLAB function:

   [Mask, FData, FNames] = geneentropyfilter(GED-MATRIX, Names, 'Percentile', PercentileValue)

5. Termination condition: Repeat the process and compute the gene and sample groups with Step1. Stop the iteration if the iteration reached 50; otherwise continue. If the number of genes dropped is less than 15% of the total number of genes, the iterations are stopped.

Figure 2 shows the block diagram of parallel fuzzy c-means approach. The algorithm follows the master/slave modelling approach for implementation. Here master processor have MATLAB and others are the slaves in which MATLAB workers runs. All the processors can share the messages passing between them.

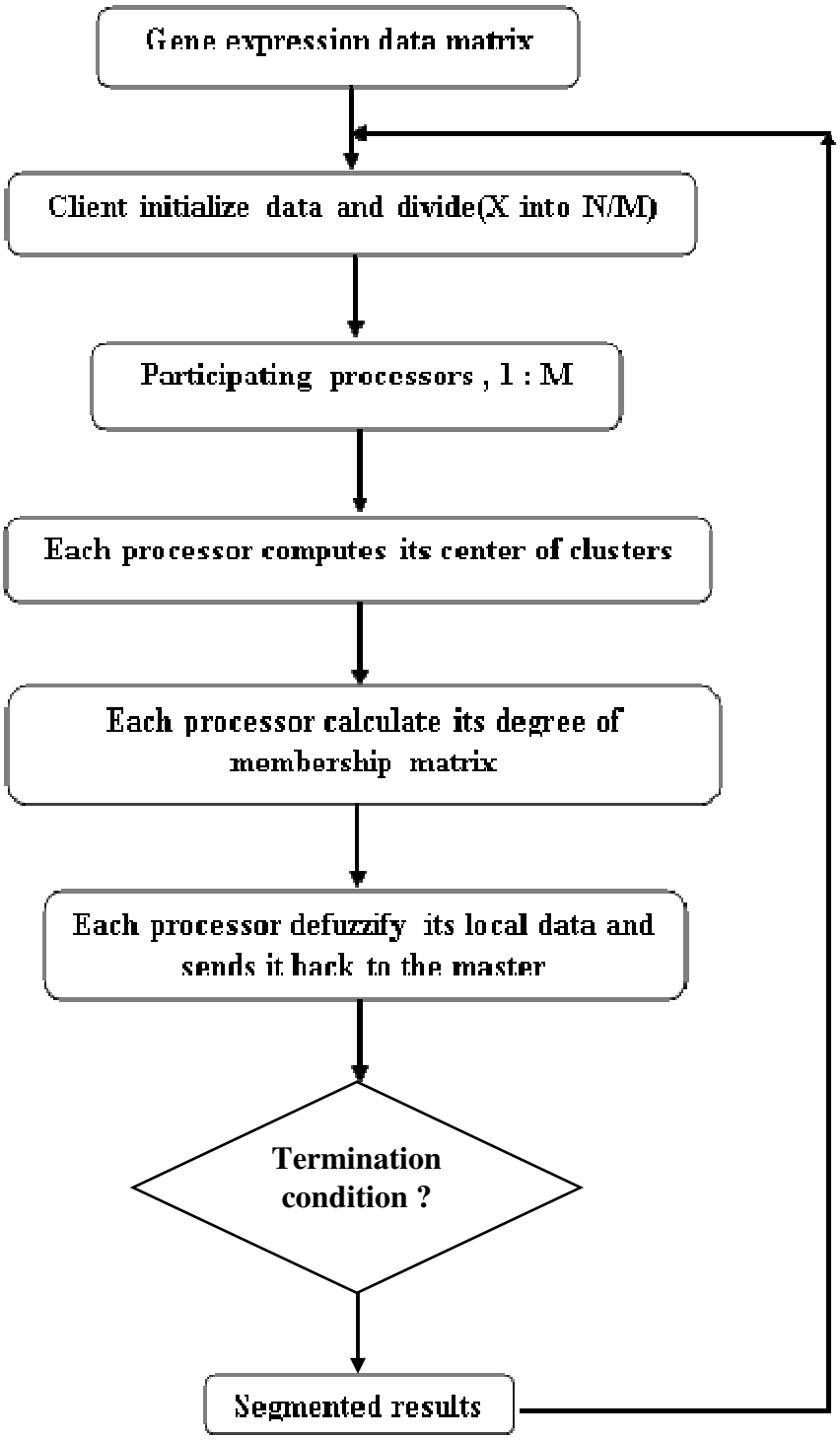

FIGURE 3: Block Diagram of Parallel Fuzzy c-means Approach.

3. RESULT & DISCUSSION

Microarray Yeast cell cycle data: In this paper we have used cell cycle of the budding yeast S. cerevisiae made by Cho et al. The data set contains 6149 genes were measured every 10 minutes under 17 time points [6]. Other yeast cell cycle data which contains the expression profiles of 6200 yeast genes were measured every 10 minutes during two cell cycles in 17

hybridization experiments [20]. In the selection of the data set for the time points 90, 100 and 120 minutes are excluded. Simple description of data sets is shown in Table1.

We have applied this algorithm on other two subtypes of breast cancer data sets named as subtype Basal A and Basal B [21].

TABLE 2: Description of Dataset Used in Study

Data Set	Dataset code	Data Size (kb)	No. of genes	No. of samples
yeast	Data1	2309	6149	17
yeast	Data2	939	2945	15
Basal A	Data3	776	1213	98
Basal B	Data4	521	1213	49
Leukemia	Data5	7222	12560	72

3.1 Performance Evaluation

We examined parallel programs for six to eight processors. The results are shown here is based on six processor computation to find the biclusters from the data sets. We have calculated 30 biclusters for each dataset and the execution time are shown in Table3, 4, 5 and 6 and graphs 4 and 5. The main objective of this study is to reduce the processing time to find the biclusters of the data. The CPU time measured of CPU time for clustering of genes and samples and is shown in Table3. A general rule of thumb is that a clustering result with lower CPU time is preferable. For a comparable assessment, we coded these methods by using the fuzzy tools available in MATLAB with MatlabMPI for parallelization [18, 19, 22]. Here we use the SPMD (Single Program Multiple Data) parallel computing model to parallelize the algorithm [20]. In SPMD the dataset is divided into number or processors and a single program will work on all the processor's local datasets.

Table4 reports the results of computation time taken by the algorithms, including clustering time of genes and samples, bicluster computation time and the total running time. Table4 shows compare that running time cost of the serial biclustering approach is higher than the parallel biclustering including parallel fuzzy c-means with MPI.

Table5 illustrates the total time overheads of serial fuzzy c-means of genes and samples, and applying MFCM on genes and samples respectively. Figure 4 and 5 show total computation time of Step 1 and 2 of the proposed biclustering algorithm when number of processors are two. We see that the execution time to cluster genes and sample in case of serial fuzzy clustering is higher than the MPI based fuzzy clustering.

TABLE 3: The Performance Result of Serial and Parallel Fuzzy c-means For Gene Based Clustering with Four Processors and K = 4.

Dataset code	FCM (sec)	MFCM (sec)
Data1	29.3829	12.9292
Data2	12.9276	10.0113
Data3	17.1909	9.0931
Data4	17.1206	10.926
Data5	48.7918	31.7711

TABLE 4: Performance Evaluation of the Complete Execution time of Serial and Parallel Biclustering with Two Processors and K=6.

Data set	Biclustering using FCM (sec)			Biclustering using MFCM (sec)		
	Clustering	Biclustering	Total	Clustering	Biclustering	Total
Data1	31.289	39.12	70.409	15.0192	19.014	34.0332
Data2	18.901	28.5	47.401	10.9018	11.0112	21.913
Data3	19.109	21.082	40.191	10.0991	11.7919	21.891
Data4	19.208	20.774	39.982	11.828	8.94	20.768
Data5	52.998	47.203	100.201	32.6431	18.1779	50.821

The parallel execution time is the elapsed time from when parallel computation starts to the moment when the last processor finishes its computation [1]. Here the sequential execution time is denoted by *TSeq* and the parallel execution time is denoted by *TPar*. Speedup calculation is defined as the ratio of the sequential execution time and the parallel execution time to solve the same problem in sequential and parallel computers respectively.

The speedup is calculated according to the following equation:

$$Speedup = \frac{TSeq}{TPar} \quad (6)$$

Table5 shows the execution time difference between serial and parallel algorithm, and speedup performance of the parallel biclustering algorithm calculated for k = 6 clusters of genes/samples for Step1 and 2. The parallel calculated time is of biclustering algorithm is based on four processors.

TABLE 5: Speedup Performance for the Biclustering of the Dataset with Time Difference Between Serial and Parallel Approach on Six Processors and Six Clusters.

Dataset Code	Number of Clusters	TSeq	TPar	TDiff	Speedup
Data1	6	70.409	22.121	48.288	3.183
Data2	6	47.401	12.01	35.391	3.947
Data3	6	40.191	12.989	27.202	3.094
Data4	6	39.982	11.5249	28.4571	3.4692
Data5	6	100.201	43.9981	56.2029	2.277

Efficiency is a measure of the fraction of time for which a processor is usefully employed. It is defined as the ratio of speedup to the number of processors (P). We examined parallel programs for six to eight processors. Efficiency by the symbol *Eff*:

$$Eff = \frac{Speedup}{P} \quad (7)$$

TABLE 6: Efficiency Calculations.

Dataset	Number of processors	TPar	Speedup	Efficiency
Data1	2	34.0332	2.068832787	1.034416393
	4	30.119	2.337693815	0.584423454
	6	22.121	3.182903124	0.530483854
	8	19.225	3.66236671	0.457795839
	2	21.913	2.163145165	1.081572582

Data2	4	19.2213	2.466066291	0.616516573
	6	12.01	3.946794338	0.657799056
	8	10.09	4.697819623	0.587227453
Data3	2	21.891	1.835959984	0.917979992
	4	20.212	1.988472195	0.497118049
	6	12.989	3.094233582	0.515705597
	8	11.2201	3.582053636	0.447756704
Data4	2	20.768	1.925173344	0.962586672
	4	20.001	1.99900005	0.499750012
	6	11.5249	3.469184114	0.578197352
	8	10.771	3.712004456	0.464000557
Data5	2	50.821	1.97164558	0.98582279
	4	45.662	2.194406728	0.548601682
	6	43.9981	2.277393797	0.379565633
	8	42.2332	2.372564712	0.296570589

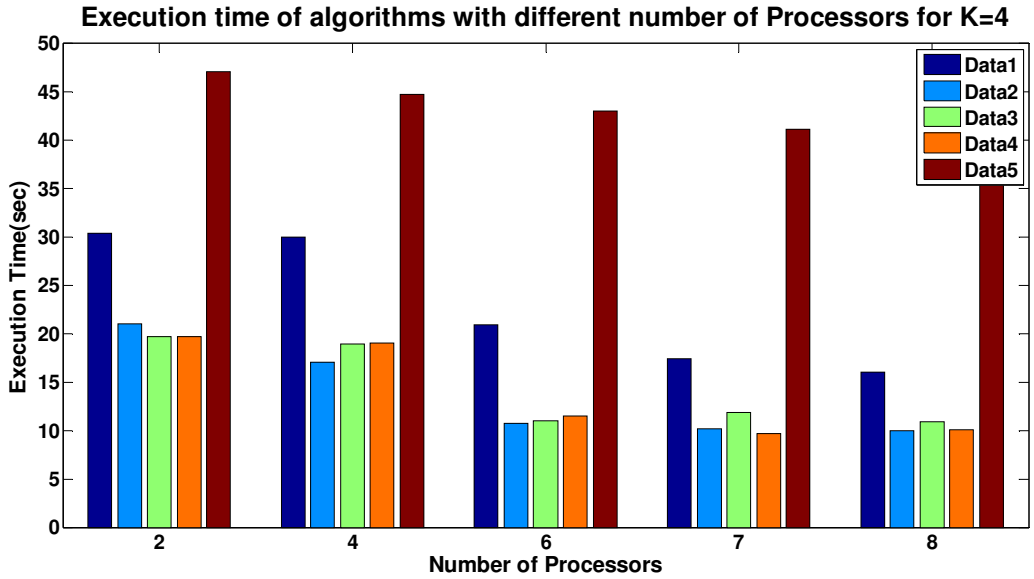

FIGURE 4: Execution time with proposed method of Datasets 1,2,3,4 and 5 with number of processors for K=6.

Figure 4 shows that the running time of biclustering with serial fuzzy c-means and biclustering with parallel fuzzy c-means with MATLABMPI when number of clusters of genes and samples is 4. It is clear from figure 4 that execution time is reducing when we increase the number of processors.

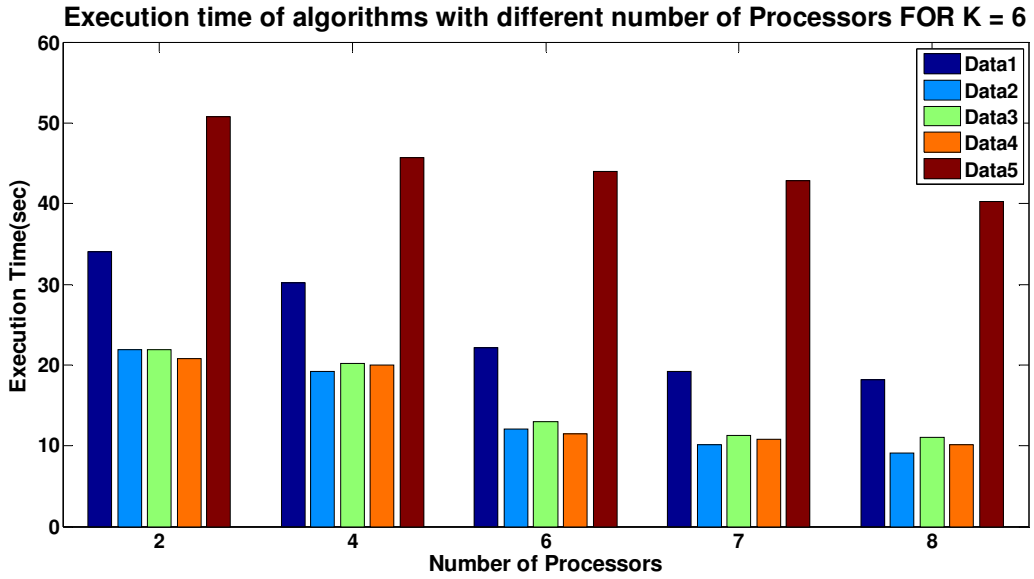

FIGURE 5: Execution time with proposed method of Datasets 1,2,3,4 and 5 with number of processors for K=6.

Figure 5 shows that the running time of biclustering with serial fuzzy c-means and biclustering with parallel fuzzy c-means with MatlabMPI when the number of clusters of genes and samples is 6. From the figure 5 it is clear that execution time is going lower when we increase the number of processors. After sixth processor the graph is showing little bit difference between execution time of the algorithm for all the data sets. The formation of biclusters also indicates that a gene or a sample may belong to more than one cluster, which in turn may be helpful in understanding the nature of biological process.

4. CONCLUSION

In this paper, we have presented parallel biclustering algorithm for yeast, cancer subtypes (BasalA, BasalB), and Leukemia microarray gene expression data used in previous study by Cho et al. and Tavazoie et al. We proposed a parallel biclustering algorithm based on MATLABMPI (MFCM) to find the genes and samples clusters. It is clear that serial algorithm needs more time to compute the clusters from datasets, and the biclustering also takes higher execution time. Parallel technology provides efficient tools to let these data clusters execute on a computer cluster. The algorithm is based on data parallel computing model, and is implemented with MatlabMPI. Experimental results show the feasibility and effectiveness of parallel algorithms of clustering and biclustering of the gene expression data. Message passing is the communication between the processes, then some time it is also possible that it takes longer time to communicate with other processes for large datasets, but it gives better performance than previously implemented algorithm [9].

This paper presented algorithms for finding biclusters from gene expression data in parallel, further exploration on the resulting biclusters will be another interesting research approach to reveal gene functions, gene regulations, and cellular process of gene regulatory networks.

5. ACKNOWLEDGEMENT

One of the authors Dwitiya Tyagi-Tiwari would like to thank Åbo Akademi ComBio Lab to provide me space and resources for working and staff for helpful discussion.

6. REFERENCES

[1] Ananth Grama, Anshul Gupta, George Karypis, and Vipin Kumar, “Introduction to parallel computing”, Addison-Wesley, 2003.

[2] A.H. Tewfik, A.B. Techagang and I. Vertatsehitsch, “Parallel Identification of Gene Biclusters with Coherent Evolutions”, IEEE Transaction on Signal Processing, Vol. 54, No. 6, June-2006.

[3] Bezdek,J.C., “Pattern Recognition With Fuzzy Objective Function Algorithms”, Plenum Press, New York, 1981.

[4] B. Chandra, S. Shankera, Saroj Mishra, "A new approach: Interrelated two-way clustering of gene expression data", Statistical Methodology 3, 2006, pp. 93–102.

[5] Chun Tang and Aidong Zhang, “Interrelated Two-Way Clustering and Its Application on Gene Expression Data ", International Journal on Artificial Intelligence Tools, 2005; Vol. 14, No. 4; pp. 577-598.

[6] Cho RJ, Campbell MJ, Winzeler EA, Steinmetz L, Conway A, Wodicka L, Wolfsberg TG, Gabrielian AE, Landsman D, Lockhart DJ, Davis RW, ‘A genome-wide transcriptional analysis of the mitotic cell cycle’, Molecular Cell, Vol. 2, 65–73, July, 1998.

[7] Dembele D, Kastner P. Fuzzy C-means method for clustering microarray data. Bioinformatics 2003; 19(8):973–80.

[8] Dwitiya Tyagi-Tiwari, Sujoy Das, and Namita Srivastava, Parallel Two-way Clustering for Microarray Gene expression data’, International Journal of Computer Science Trends and Technology, Vol. 3 Issue 3, May-June 2015.

[9] Dwitiya Tyagi-Tiwari, Sujoy Das and Namita Srivastava. Article: Two-Way Clustering Analysis using Parallel Fuzzy Approach for Microarray Gene Expression Data. International Journal of Computer Applications 124(9), pp. 39-45, August 2015.

[10] G. Kerr, H.J. Ruskin, M. Crane and P. Doolan, “Techniques for clustering gene expression data”, Computers in Biology and Medicine 38, pp. 283-293, 2008.

[11] Hartigan J.: “Direct Clustering of a Data Matrix”, J Am Stat Assoc 1972, 67(337), pp. 123-129.

[12] Huimin Geng, Dhundy Bastola, and Hesham Ali, “A New Approach to Clustering Biological Data Using Message Passing”, Proceedings of the IEEE Computational Systems Bioinformatics Conference, 2004.

[13] Jeremy Kepner, “Parallel programming with MATLABMPI”, 2002, High Performance Embedded Computing (HPEC) workshop, MIT Lincoln Laboratory, Lexington, MA, http://arXiv.org/abs/astro-ph/0107406

[14] Liu Weihj And Chen Ling, “*A Parallel Algorithm for Gene Expressing Data Biclustering*”, Journal Of Computers, Vol. 3, No. 10, October 2008.

[15] Li Li, Yang Guo, Wenwu Wu, Youyi Shi, Jian Cheng and Shiheng Tao, 'A comparison and evaluation of five biclustering algorithms by quantifying goodness of biclusters for gene expression data', BioData Mining 2012, Vol-8 pp. 1756-0381.

[16] Matthias E. Futschik and Nikola K. Kasabov, “Fuzzy Clustering of Gene Expression Data”, 2002 IEEE International Conference on Fuzzy Systems, 2002, Vol 1, pp. 414-419.

[17] MATLAB the MathWorksTM Accelerating the pace of engineering and science Parallel Computing Toolbox™ 4 User's Guide, 2009.

[18] Sara C. Madeira and Arlindo L. Oliveira, "Biclustering Algorithms for Biological Data Analysis: A Survey", IEEE/Acm Transactions on Computational Biology and Bioinformatics Vol 1, No. 1, January-March 2004, pp. 24-45.

[19] T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gassenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing,M.A. Caligiuri, D.D. Bloomfield, E.S. Lander, Molecular classification of cancer: class discovery and class prediction by gene expression monitoring, Science 286 (15) (1999) pp 531–537.

[20] Terence Kwok, Kate Smith, Sebastian Lozano and David Taniar, "Parallel Fuzzy c-Means Clustering for Large Data Sets", Springer-Verlag Berlin Heidelberg 2002, LNCS 2400, pp. 365-374.

[21] Yizong Cheng and George M. Church, "*Biclustering of Expression Data*", Proc. ISMB'00, pp. 93-103, 2000.

[22] Tavazoie,S., Hughes,J.D., Campbell,M.J., Cho,R.J. and Church,G.M. (1999) Systematic determination of genetic network architecture. Nat. Genet., 22, 281–285.

[23] Hoshida Y, Brunet J-P, Tamayo P, Golub TR, Mesirov JP Subclass Mapping: Identifying Common Subtypes in Independent Disease Data Sets. PLoS ONE 2(11), 2007.

[24] https://www.ll.mit.edu/mission/cybersec/softwaretools/MATLABmpi/MATLABmpi.html.

[25] https://www.mpich.org/documentation/guides/.