ISSN 2047-3338

Unsupervised Learning Algorithms to Identify the Dense Cluster in Large Datasets

Dr. Ch. G.V.N. Prasad, K. Hanumantha Rao, Depa Pratima and B.N. Alekhya
Sri Indu College of Engineering and Technology, Hyderabad, India

***Abstract*– High dimensional database is having large datasets while solving the cluster identification problem and for identifying dense clusters in a noisy data. Our analysis works to identifies clusters through the identification of densely intra connected sub graphs, we have employed a pattern recognition algorithms representation of the graph and solve the cluster identification problem using K-means, K-modes, Single Linkage Clustering. The computational analysis indicate that when running on 150 CPUs, one of our algorithm can solve a cluster identification problem on a data set with 1,000,000 data points almost 100 times faster than on single CPU, indicating that this program is capable of handling very large data clustering problems in an efficient manner The implementation of the clustering algorithm as the software CLUMP.**

***Index Terms*– Data Mining, Clustering, K-means and K-modes**

I. INTRODUCTION

CLUSTERING means the act of partitioning an unlabeled dataset into groups of similar objects. The goal of clustering is to group sets of objects into classes such that similar objects are placed in the same cluster while dissimilar objects are in separate clusters. Clustering is used as a data processing technique in many different areas, including artificial intelligence, bioinformatics, biology, computer vision, city planning, data mining, data compression, earth quake studies, image analysis, image segmentation, information retrieval, machine learning, marketing, medicine, object recognition, pattern recognition, spatial database analysis, statistics and web mining [1]. Any cluster should exhibit two main properties; low inter-class similarity and high intra-class similarity. Clustering is an unsupervised learning i.e. it learns by observation rather than examples. There are no predefined class label exists for the data points. Clustering helps in gaining, overall distribution of patterns and correlation among data objects.

In pattern recognition, data analysis is concerned with predictive modeling: given some training data, we want to predict the behavior of the unseen test data. This task is also referred to as learning. Often, a clear distinction is made between learning problems that are: (i) supervised (classification) or (ii) unsupervised (clustering), the first involving only labeled data (training patterns with known category labels) while the latter involving only unlabeled data [Duda et al., 2001]. Clustering is a more difficult and challenging problem than classification. There is a growing interest in a hybrid setting, called semi-supervised learning [Chapelle et al., 2006] in semi-supervised classification; the labels of only a small portion of the training data set are available. The unlabeled data, instead of being discarded, are also used in the learning process. In semi-supervised clustering, instead of specifying the class labels, pair-wise constraints are specified, which is a *weake*r way of encoding the prior knowledge A pair-wise must-link constraint corresponds to the requirement that two objects should be assigned the same cluster label, whereas the cluster labels of two objects participating in a cannot-link constraint should be different. Constraints can be particularly beneficial in data clustering [Lange et al., 2005, Basu et al., 2008], where precise definitions of underlying clusters are absent. In the search for good models, one would like to include all the available information, no matter whether it is unlabeled data, data with constraints, or labeled data. Fig. 1 illustrates this spectrum of different types of learning problems of interest in pattern recognition and machine learning.

II. DATA CLUSTERING

Data clustering, also known as cluster analysis, is to discover the natural grouping(s) of a set of patterns, points, or objects. Webster [Merriam-Webster Online Dictionary, 2008] defines cluster analysis as "*a statistical classification technique for discovering whether the individuals of a population fall into different groups by making quantitative comparisons of multiple characteristics.*" The objective is to develop an automatic algorithm that will discover the natural groupings (Fig. 2 (b)) in the unlabeled data (Fig. 2 (a)).

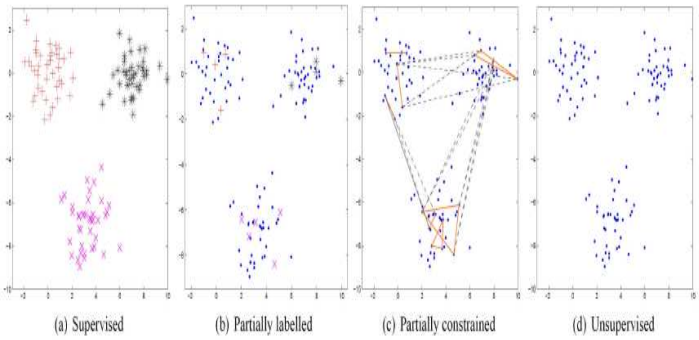
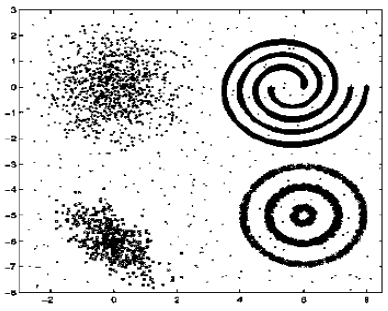


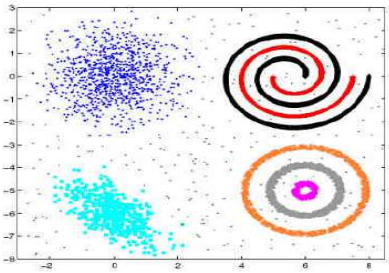
Fig. 1. Learning problems: dots correspond to points without any labels. Points with labels are denoted by plus signs, asterisks, and crosses. In (c), the must-link and cannot-link constraints are denoted by solid and dashed lines, respectively (figure taken from [Lange et al., 2005]).

Definition of Cluster: Given a *representation* of *n* objects, find *K* groups based on a measure of *similarity* such that the similarities between objects in the same group are high while the similarities between objects in different groups are low. Figure 2 shows that clusters can differ in terms of their *shape*, *size*, and *density*. The presence of noise in the data makes the detection of the clusters even more difficult. An ideal cluster can be defined as a set of points that is *compact* and *isolated*.

In reality, a cluster is a subjective entity that is in the eye of the beholder and whose significance and interpretation requires domain knowledge. But, while humans are excellent cluster seekers in two and possibly three dimensions, we need automatic algorithms for high dimensional data. It is this challenge along with the unknown number of clusters for the given data that has resulted in thousands of clustering algorithms that have been published and that continue to appear.



(a) Input data



(b) Desired clustering

Fig. 2. Diversity of clusters: The seven clusters in (a) denoted by seven different colors in 1, (b) differ in shape, size, and density. Although these clusters are apparent to a data analyst, none of the available clustering algorithms can detect all these clusters

A. Why to Cluster

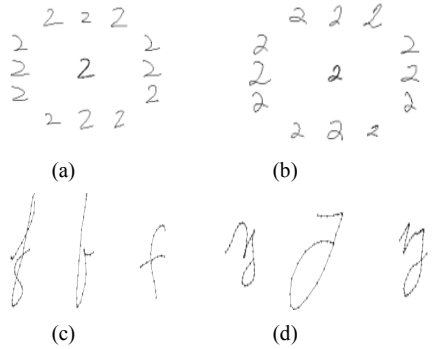
Cluster analysis is prevalent in any discipline that involves analysis of multivariate data. A search via Google Scholar [gsc, 2009] found 1,660 entries with the words data clustering that appeared in 2007 alone. This vast literature speaks to the importance of clustering in data analysis. It is difficult to exhaustively list the numerous scientific fields and applications that have utilized clustering techniques as well as the thousands of published algorithms. Clustering is also used to group customers into different types for efficient marketing [Arabie & Hubert, 1994], to group services delivery engagements for workforce management and planning [Hu et al., 2007] as well as to study genome data [Baldi & Hatfield, 2002] in biology.

Data clustering has been used for the following three main purposes:

• *Underlying structure*: to gain insight into data, generate hypotheses, detect anomalies, and identify salient features.

• *Natural classification*: to identify the degree of similarity among forms or organisms (phylogenetic relationship).

• *Compression*: as a method for organizing the data and summarizing it through cluster prototypes. An example of class discovery is shown in Fig. 3. Here, clustering was used to discover subclasses in an online handwritten character recognition application [Connell & Jain, 2002]. Different users write the same digits in different ways, thereby increasing the within-class variance. Clustering the training patterns from a class can discover new subclasses, called the lexemes in handwritten characters. Instead of using a single model for each character, multiple models based on the number of subclasses are used to improve the recognition accuracy.



(a) (b) (c) (d)

Fig. 3. Finding subclasses using data clustering: (a) and (b) show two different ways of writing the digit 2; (c) three different subclasses for the character ‘f’; (d) three different subclasses for the letter ‘y’

B. Types of Clusters

1). Hierarchical Clustering Algorithm

Hierarchical clustering algorithm group’s data objects to form a tree shaped structure. It can be broadly classified into agglomerative hierarchical clustering and divisive hierarchical clustering. In agglomerative approach which is also called as bottom up approach, each data points are considered to be a separate cluster and on each iteration clusters are merged based on a criteria. The merging can be done by using single link, complete link, and centroid or wards method. In divisive approach all data points are considered as a single cluster and they are splited into number of clusters based on certain criteria, and this is called as top down approach. Examples for this algorithm are LEGCLUST [4] and BRICH.

2). Spectral Clustering Algorithm

Spectral clustering refers to a class of techniques which relies on the Eigen structure of a similarity matrix. Clusters are formed by partition data points using the similarity matrix. Any spectral clustering algorithm will have three main stages [5]. They are as follows:

i). Preprocessing: Deals with the construction of similarity matrix.

ii). Spectral Mapping: Deals with the construction of Eigen Vectors for the similarity matrix

iii). Post Processing: Deals with the grouping data points

The following are advantages of Spectral clustering algorithm:

i. Strong assumptions on cluster shape are not made
ii. Simple to implement
iii. Objective does not consider local optima
iv. Statistically consistent
v. Works faster

The major drawback of this approach is that it exhibits high computational complexity. For the larger dataset it requires O (n3) where n is the number of data points [2]. Examples for these algorithms are SM (Shi and Malik) algorithm, KVV (Kannan, Vempala andVetta) algorithm, NJW (Ng, Jordan and Weiss) algorithm [4].

3). *Grid based Clustering Algorithm*

Grid based algorithm quantize the object space into a finite number of cells that forms a grid structure [1].Operations are done on these grids. The advantage of this method is lower processing time. Clustering complexity is based on the number of populated grid cells and does not depend on the number of objects in the dataset. The major features of this algorithm are:

i. No distance computations.
ii. Clustering is performed on summarized data points.
iii. Shapes are limited to union of grid-cells.
iv. The complexity of the algorithm is usually O (Number of populated grid-cells)

STING [1] is an example for this algorithm.

4). Density based Clustering Algorithm

Density based algorithm continue to grow the given cluster as long as the density in the neighborhood exceeds certain threshold [1]. This algorithm is suitable for handling noise in the dataset. The following points are enumerated as the features of this algorithm:

i. Handles clusters of arbitrary shape
ii. Handle noise
iii. Needs only one scan of the input dataset?
iv. Needs density parameters to be initialized

DBSCAN, DENCLUE and OPTICS [1] are examples for this algorithm.

III. Pattern Recognition

Pattern recognition is generally categorized according to the type of learning procedure used to generate the output value. Many common pattern recognition algorithms are *probabilistic* in nature, in that they use statistical inference to find the best label for a given instance. Unlike other algorithms, which simply output a "best" label, often times probabilistic algorithms also output a probability of the instance being described by the given label. In addition, many probabilistic algorithms output a list of the *N*-best labels with associated probabilities, for some value of *N*, instead of simply a single best label. When the number of possible labels is fairly small (e.g. in the case of classification), *N* may be set so that the probability of all possible labels is output. Probabilistic algorithms have many advantages over non-probabilistic algorithms:

They output a confidence value associated with their choice. (Note that some other algorithms may also output confidence values, but in general, only for probabilistic algorithms are this value mathematically grounded in probability theory. Non-probabilistic confidence values can in general not be given any specific meaning, and only used to compare against other confidence values output by the same algorithm).

Correspondingly, they can *abstain* when the confidence of choosing any particular output is too low.

Because of the probabilities output, probabilistic pattern-recognition algorithms can be more effectively incorporated into larger machine-learning tasks, in a way that partially or completely avoids the problem of *error propagation*.

Techniques to transform the raw feature vectors are sometimes used prior to application of the pattern-matching algorithm. For example, feature extraction algorithms attempt to reduce a large-dimensionality feature vector into a smaller-dimensionality vector that is easier to work with and encodes less redundancy, using mathematical techniques such as principal components analysis (PCA). Feature selection algorithms, attempt to directly prune out redundant or irrelevant features. The distinction between the two is that the resulting features after feature extraction has taken place are of a different sort than the original features and may not easily be interpretable, while the features left after feature selection are simply a subset of the original features.

A. Pattern Recognition Algorithms

1). K-Means Clustering

Unsupervised K-means learning algorithms that solve the well known clustering problem. The procedure follows to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centroids, one for each cluster. These centroids should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early group age is done. At this point we need to re-calculate k new centroids as barycenters of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid.

A loop has been generated. As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done. In other words centroids do not move anymore. Finally, this algorithm aims at minimizing an *objective function*, in this case a squared error function. The objective function:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^j - c_j \right\|^2$$

Where $\left\| x_i^j - c_j \right\|^2$ is a chosen distance measure between a data point x_i^j and the cluster centre c_j is an indicator of the distance of the n data points from their respective cluster centers.

i). Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.

ii). Assign each object to the group that has the closest centroid
iii). When all objects have been assigned, recalculate the positions of the K centroids
iv). Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated

2). Agglomerative & Divisive

Agglomerative is a bottom up approach which builds up clusters from single objects
Divisive is a top-down approach which breaks up cluster containing all objects into smaller clusters. Both are hierarchy can be trivial:

1 (. .) . . . 2 ((. .) .) . .
3 (((. .) .) .) . 4 ((((. .) .) .) .)

General Agglomerative

• Uses any computable cluster similarity measure sim(Ci, Cj)
• For n objects v1, …, vn, assign each to a singleton cluster Ci = {vi}.
• repeat {
identify two most similar clusters Cj and Ck (could be ties chose one pair)
delete Cj and Ck and add (Cj U Ck) to the set of clusters
} until only one cluster
• Dendrograms diagram the sequence of cluster merges.

Divisive
1. Put all objects in one cluster
2. Repeat until all clusters are singletons
a) choose a cluster to split
• what criterion?
b) replace the chosen cluster with the sub-clusters
Split into how many?
How split?
• "reversing" agglomerative => split in two
• cutting operation: cut-based measures seem to be a natural choice.
– focus on similarity across cut - lost similarity
• not necessary to use a cut-based measure

General Divisive :
Iterative Improvement; no hierarchy
1. Choose initial partition C1, … , Ck
2. repeat {
unlock all vertices
repeat {
choose some Ci at random
choose an unlocked vertex vj in Ci
move vj to that cluster, if any, such that move gives maximum decrease in cost
lock vertex vj
} until all vertices locked
}until converge

IV. Problem Definition

Data (Objects) clustering represents one of the most often encountered problems in the data analyses. The basic problem is to partition a data set into "clusters" of data points that are "close" to each other but relatively "far from" other data points.

A more general problem is the so-called cluster identification problem which is to identify "dense" clusters in a possibly noisy background. Need to develop an efficient clustering approach to perform clustering on large data sets where high dense clusters involved

There are so many clustering algorithm means "K-Means, K-Mode, Single Linkage Algorithm (SLA)" that are working under distance measurement mechanism. A new parallel clustering algorithm has been proposed to perform clustering in large data sets under parallel mechanism.

A. The k-modes Algorithm

In principle the formulation of K-mode is also valid for categorical and mixed type objects. The cause that the *k*-means algorithm cannot cluster categorical objects is its dissimilarity measure and the method used to solve problem *P*2.

These barriers can be removed by making the following modifications to the *k*-means algorithm:

i). using a simple matching dissimilarity measure for categorical objects,
ii). replacing means of clusters by modes,
iii). using a frequency-based method to find the modes to solve problem *P*2

• *K-modes Algorithm*

The dissimilarity measure for categorical objects becomes

$$P(W,Q) = \sum_{l=1}^{k}\sum_{i=1}^{n}\sum_{j=1}^{m} w_{i,l}\,\delta(x_{i,j}, q_{l,j})$$

Where $w_{i,l} \in W$ and $Q_l = [q_{l,1}, q_{l,2,\ldots,} q_{l,m}] \in Q$

In the basic algorithm we need to calculate the total cost *P* against the whole data set each time when a new ***Q*** or *W* is obtained. To make the computation more efficient we use the following algorithm instead in practice:

i). Select *k* initial modes, one for each cluster.
ii). Allocate an object to the cluster whose mode is the nearest to the mode of the cluster after each allocation.
iii). After all objects have been allocated to clusters, retest the dissimilarity of objects against the current modes. If an object is found such that its nearest mode belongs to another cluster rather than its current one, reallocate the object to that cluster and update the modes of both clusters.
iv). Repeat 3 until no object has changed clusters after a full cycle test of the whole data set.

B. Single Linkage Algorithm

Single link algorithm is an example of agglomerative hierarchical clustering method. We recall that is a bottom-up strategy: compare each point with each point. Each object is placed in a separate cluster, and at each step we merge the closest pair of clusters, until certain termination conditions are

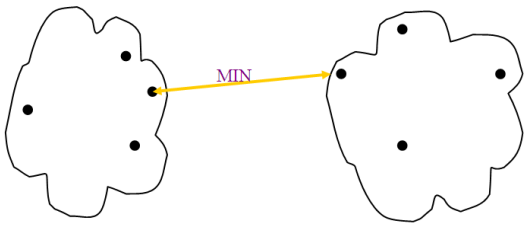


Fig. 4. Two clusters

satisfied. This requires defining a notion of cluster proximity. For the single link, the proximity of two clusters is defined as the minimum of the distance between any two points in the two clusters. Using graph terminology, if we start with all points, each one a separate cluster on its own (called a singleton cluster), and then add links between all points one at a time – shortest links first, then these single links combine the points into clusters. (i.e. the points with the shortest distance between each other are combined into a cluster first, then the next shortest distance are combined, and so on)

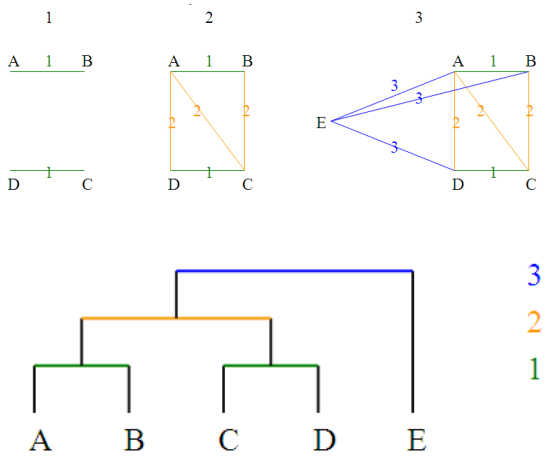


Fig. 5. Two clusters example

Dendogram- shows the same information as in the graph above, however distance threshold is vertical, and points are at the bottom (horizontal). The height at which two clusters are merged in the dendogram reflects the distance of the two clusters.

C. Applications of Parallel Clustering Algorithms

Parallel clustering algorithms can be applied to any of applications using clustering algorithms for efficient computing. One of the common applications of clustering algorithms is image segmentation. Khotanzad and Bouarfa presented a non-parametric parallel clustering algorithm in [11] for the task of unsupervised image segmentation. It builds a multi-dimensional histogram of the feature vectors to see the distribution of feature vectors and approximate the density of the data. Each peak is considered as a cluster. The process of identifying clusters is automatic as it proposes systematic methods for choosing an appropriate histogram cell size. After the clustering, the segmentation task is followed and they are implemented in a SEQUENT parallel computer. The parallel computation is done using message-passing protocol as well.

A parallel bottom-up clustering algorithms were applied to partition circuits in VLSI in the [10]. Unlike many top-down partitioning algorithms for layouts, the bottom-up layout clustering algorithms are useful as circuit sizes increase. The parallel clustering algorithm was implemented by dividing the graph into multiple processors. Each processor forms clusters of the graph, occasionally swaps parts so that cliques to be found. The bottom-up clustering tasks also are used as preprocessing for partitioning as each cluster is then assigned to each node.

Bandyopadhyay and Coyle presented a distributed and randomized clustering algorithm to layout the sensors in a wireless sensor network [9]. On top of the clustering network sensors, the authors organized the cluster-heads hierarchically, so that it saves the energy more. It assumes the sensors in a network are simple and all the sensors communicate at a fixed power level. Using the experimental fact that a network with one level of clustering has an optimal number of cluster heads minimizing the energy, analysis to obtain the optimal number of cluster-heads at each level of clustering. Anantharaman *et al.* [7] developed software called PaCE which is parallel and fast clustering method of large-scale EST (Expressed Sequence tags) in gene identification.

ESTs are sequences at most a few hundred base pairs determined by single-pass sequencing of the 5' or 3' ends of cDNA. EST representations provide a dynamic view of genome content and expression. With the huge EST data library, more than 5 million human ESTs, the primary goal in ESTs is to derive sets of unique genes from the data which represent the transcript me of each species. Therefore, the first step would be clustering ESTs to remove redundancy in EST sets. A parallel software program, named PaCE, which clusters ESTs into the same gene or paralogous genes. Then each cluster is further processed with an assembly tool for generating one consensus sequence per putative transcript. The memory with PaCE in O (n) and the input size of the assembly tools is now reduced to the size of each cluster.

The PaCE implements one of the bottom-up hierarchical clustering in parallel. That is, each pair of clusters is grouped recursively. The PaCE software has two phases: GST (generalized suffix tree) construction as a preprocessing phase, and pair generation, pair wise alignment and EST cluster management as the clustering phase. That is, a distributed representation of the GST is first built in parallel. A master processor maintains and updates the EST clusters and the rest of processors generate batches of promising pairs and perform pair wise alignment on selected promising pairs. Master process is also responsible for selecting promising pairs at each step. Kraj *et al.* illustrated the parallel k-means algorithm, called para KMeans, for general laboratory use in [8].

The parallel computation is performed to the cluster assignment by dividing the data set into sub-processors to assign the memberships. The Para KMeans is a software program that can be used for practical applications.

V. Conclusion

Our article shows the analysis of unsupervised learning, pattern recognition, and parallel algorithms to identify the dense clusters in the noisy data. To identify the dense cluster, we need to construct the one of the clustering algorithm in our analysis using the software CLUMP has proved to be a highly useful tool for clustering large quantities of dataset. According to analysis of the article for a typical data clustering problem with 1,000,000 data points in 30 dimensions, CLUMP calculates in 40 minutes on 105 Intel x 86 processors. CLUMP is an open source software developing tool by adding new distance functions with software features.

References

[1]. A.K. Jain, M.N. Murty and P.J. Flynn, "Data Clustering: a review", (Balance Iterative Reducing and Clustering using Hierarchies), CURE (Cluster Using Representatives) Chemeleon.

[2]. Cai X. Y. et al,. "Survey on Spectral Clustering Algorithms", Computer Science, (14-18), 2008.

[3]. Santos, J.M, de Sa, J.M, Alexandre, A,. "LEGClust- A Clustering Algorithm based on Layered Entropic subgraph. Pattern Analysis and Machine Intelligence", IEEE Transactions, 62- 75, 2008.

[4]. M Meila, D Verma,. "Comparison of spectral clustering algorithm", University of Washington,Technical Report, 2001.

[5]. Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values ZHEXUE HUANG huang@mip.com.au ACSys CRC, CSIRO Mathematical and Information Sciences, GPO Box 664, Canberra, ACT 2601, Australia.

[6]. Anantharaman Kalyanaraman, Srinivas Aluru, Suresh Kothari, and Volker Brendel. Efficient clustering of large EST data sets on parallel computers. Nucl. Acids Res., 31(11):2963–2974, 2003.

[7]. Piotr Kraj, Ashok Sharma, Nikhil Garge, Robert Podolsky, and Richard A. Mcindoe. ParaKMeans: Implementation of a parallelized K-means algorithm suitable for general laboratory use. BMC Bioinformatics, 9:200+, April 2008.

[8]. Seema Bandyopadhyay and Edward J. Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks, Volume 3, pages: 1713–1723, Vol.3, March-April 2003.

[9]. Jason Cong and M'Lissa Smith. A parallel bottom-up clustering algorithm with applications to circuit partitioning in VLSI design. In DAC '93: Proceedings of the 30th international conference on Design automation, Pages: 755–760, New York, NY, USA, 1993. ACM.

[10]. Alireza Khotanzad and Abdelmajid Bouarfa. Image segmentation by a parallel, non-parametric histogram based clustering algorithm. Pattern Recognition, 23(9):961 – 973, 1990.

Dr. Ch. G.V.N. Prasad has done his M.Tech in Computer Science from JNTU, Hyderabad. Ph.D (CS) in Data Mining from Allahabad University. He has 20 years of Teaching and Industry experience, Also has worked as Scientist in National Informatics Centre, Hyderabad. Currently working as Professor and Head of the Department CSE at Sri Indu College of Engineering & Technology, Hyderabad, He has guided many PG level and engineering students. Life member of CSI.

K. Hanumantha Rao, working as Asst. Prof. at Sri Indu College of Engineering & Technology, Hyderabad, He has guided many PG level and Engineering students, areas of interest are Operating System, Mobile Computing, Information Security.

Depa Pratima pursuing M.Tech CSE at Sri Indu College of Engineering & Technology from JNTU Hyderabad, areas of interest are Networks, Data Warehousing & Mining (Email_ID: pratima.depa@gmail.com).

B.N. Alekhya M.Tech CSE at Sri Indu College of Engineering & Technology from JNTU Hyderabad, areas of interest are Mobile Computing, Data Warehousing & Mining (Email_ID: alekhya_blissful@yahoo.com)