Incremental Cluster Detection using a Soft Computing Approach

Alpa Reshamwala
MPSTME, SVKM's NMIMS, Mumbai-400056

Vijay Katkar
MPSTME, SVKM's NMIMS, Mumbai-400056

Mamta Ubnare
MPSTME, SVKM's NMIMS, Mumbai-400056

ABSTRACT
Clustering is the process of locating patterns in large data sets. As databases continue to grow in size, efficient and effective clustering algorithms play a paramount role in data mining applications. Traditional clustering approaches usually analyze static datasets in which objects are kept unchanged after being processed, but many practical datasets are dynamically modified which means some previously learned patterns have to be updated accordingly. Re-clustering the whole dataset from scratch is not a good choice due to the frequent data modifications and the limited out-of-service time, so the development of incremental clustering approaches is highly desirable. In this paper, we propose an incremental algorithm, IPYRAMID: *Incremental Parallel hYbrid clusteRing using genetic progrAmming and Multiobjective fItness with Density* employs a combination of data parallelism, genetic programming (GP), special operators, and multi-objective density-based incremental fitness function. Although many incremental clustering algorithms have been proposed which can handle insertion of new record properly using incremental approach but cannot handle deletion of record properly. This issue is resolved in the proposed algorithm and density based incremental fitness function that helps to handle outliers. Use of parallelism increases the speed of execution as well as identifies clusters of arbitrary shapes. The incremental merge engine can dynamically determine the number of clusters. Preliminary experimental results show that it can increase the efficiency of clustering process.

Keywords
Data Mining, Clustering, Genetic Programming, Parallelism, Density, Incremental mining.

1. INTRODUCTION
Clustering is a division of data into groups of similar objects. Representing the data by fewer clusters results in loss of certain fine details, but achieves simplification. It models data by its clusters. Data modeling puts clustering in a historical perspective rooted in mathematics, statistics, and numerical analysis. From a machine learning perspective clusters correspond to hidden patterns, the search for clusters is unsupervised learning, and the resulting system represents a data concept. From a practical perspective clustering plays an outstanding role in data mining applications such as scientific data exploration, information retrieval and text mining, spatial database applications, Web analysis, CRM, marketing, medical diagnostics, computational biology, and many others. Traditional clustering approaches usually analyze static datasets in which objects are kept unchanged after being processed, but many practical datasets are dynamically modified which means some previously learned patterns have to be updated accordingly. Re-clustering the whole dataset from scratch is not a good choice due to the frequent data modifications and the limited out-of-service time, so the development of incremental clustering approaches is highly desirable. [1], [2] propose efficient methods for incrementally modifying a set of association rules mined from a database. [3] Introduces generalization algorithms for incremental summarization in a data warehouse environment. [4] Proposes the first incremental clustering algorithm in a data warehouse environment. The paper implements an algorithm called ***I**ncremental **P**arallel h**Y**brid cluste**R**ing using genetic progr**A**mming and Multi-objective f**I**tness with **D**ensity* (IPYRAMID). While still leaving significant challenges unresolved, such as handling higher dimensions and dependence on user supplied parameters, IPYRAMID employs a combination of data parallelism, genetic programming (GP), special operators, and multi-objective density-based fitness function in the context of clustering to resolve most of the above challenges. The data space is divided into cells that become the target of clustering thus eliminating dependence on the order of data input. A divide-and-conquer data parallelism is used to increase execution speed. The algorithm divides the data set onto multiple processors each of which executes a genetic program that uses a flexible individual representation that can represent arbitrary shaped clusters. The genetic program also utilizes a density-based incremental fitness function that helps to handle outliers. It also introduces an incremental merge method that determines the number of clusters dynamically. Preliminary experiments have shown positive results.

2. RELATED WORK
Zhou et al.[5] has proposed GDCP Genetic Clustering algorithm and to make it applicable in incremental environment it has proposed an Incremental Clustering algorithm—ICGD. [6] Has proposed GDCA which is a Grid Density-based Clustering Algorithm and to make it useful in incremental environment it has proposed an Incremental Grid Density based Clustering Algorithm-IGDCA. Both [5] and [6] deals with a bulk of updates rather than single update. The problem with [5], [6] is that, they can handle insertion of new record in database properly using incremental approach but cannot handle deletion of record form database properly. [7] Has proposed an incremental clustering algorithm which takes only one scan of the data to find clusters. The use of objects for representation of points of cluster along with the Balanced Search tree increases the speed of the algorithm. But the major limitation of this algorithm is that, it works with relational database only. [5], [6], [7] cannot handle outliers properly. [8] Has proposed IOFCA which is an Incremental algorithm for clusters generation and it can handle Outlier Factors. It handles insertion as well as deletion of records from database properly. The major drawback of this approach is, it requires huge amount of memory while processing. [9] Has proposed Genetic Algorithm for generating K no of Clusters-*GAKC* and it can efficiently cluster the data set which has

ambiguous boundaries. The major drawback of this approach is, it cannot handle outliers properly

3. ARCHITECTURE OF IPYRAMID

The architecture of IPYRAMID is as shown in Figure 1:

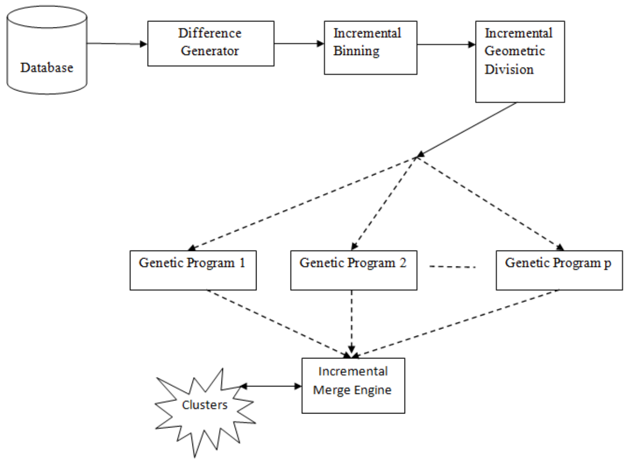

Figure 1 Architecture of IPYRAMID

A. Difference Generator It identifies the change in database D. When new data is inserted or original data is deleted, clustering process must modify the existed clusters to reflect the changes. Let ∇d be the deleted data from *D*, and Δd be the inserted data to *D*. The new database $D' = D \cup \Delta d - \nabla d$. Since updates can be seen as a series of insertions and deletions, we only consider insertions and deletions. So we define the difference database ΔD as the set of $\Delta d \cup \nabla d$.

B. Incremental Binning

This is a quantization process, where a Minimum Bounding Rectangle (MBR) is defined on the difference database ΔD. MBR is the smallest hyper-rectangular area in the data space that contains all the *2D* data points in a given data set. Master Processors conduct binning on MBR. On these MBR, master processors conduct binning. Each dimension is divided into finite number of intervals called bins. All bins within a dimension *m* have the same bin width, denoted w*m*. This is calculated using equation (1), where *óm* is the standard deviation of the coordinates of the data points on dimension *m*.

$$Wm = 3.5 \text{ X } \delta_m \text{ X } n^{-1/3} \quad (1)$$

There is no overlap between any two bins within the same dimension. The lower and upper bounds of the *i*th bin with respect to dimension *m*, called bin*i*,*m*, are respectively denoted lb_bin*i*,*m* and ub_bin*i*,*m*. Experimental results have been taken with *m*=2.

C. Incremental Geometric Division

It divides the data space of the ΔD into quadrants. A quadrant encompasses a data subset, which is formed by the data points that belong to its constituent cells as defined in [10]. Fig 2 and Fig 3 shows the geometric division of the initial increment on dataset1. The details of the algorithm are outlined below.

Algorithm:

1. Select *v*, number of divisions per dimension of ΔD. Calculate required division subset size as $s = n/v$, where *n* is the number of points.
2. For every dimension $m = 1,...,d$, traverse grid columns from lower to upper bin numbers, adding the cardinalities of their constituent cells. Stop when > *s*.
3. Compare which is closest to *s*: before or after adding the last column. If before, do not count the last column, else count it.
4. Continue until *v* divisions have been identified.
5. The intersections of the resulting divisions on every dimension form the $p = vd$ subsets, called *quadrants*, which will be transmitted to the *p* slave processors for parallelism.
6. Examine number of data points in every quadrant. If one quadrant contains less than half of an adjacent quadrant, adjust border between the two quadrants, with respect to all dimensions if necessary, until that is not the case anymore. Repeat for all quadrants.

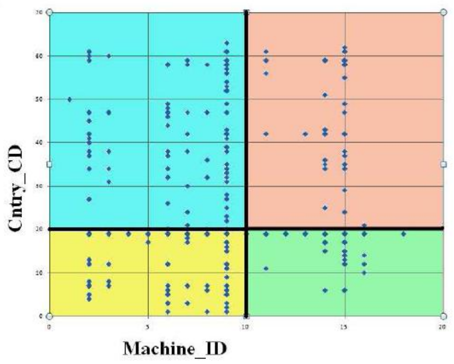

Figure 2 Incremental Geometric division

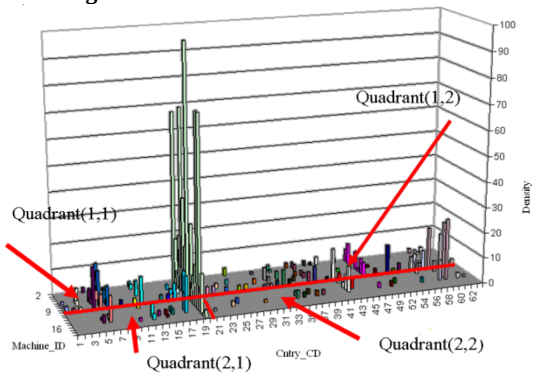

Figure 3 3D view of the incremental geometric division

D. Genetic Program

A genetic program typically represents a solution as a tree Δbased individual [7]. In this paper, each individual is encoded as a combination of blocks (rules) to form a genetic programming tree with leaf nodes symbolizing these constituent rules [11]. *Individual:* Individual I is the region in the MBHR that is a union of rules, called I's constituent rules. In the context of clustering, an individual constitutes one possible solution to the clustering. The tree representation and its 2D representation is shown in Figure 4 and its rules in Figure 5.

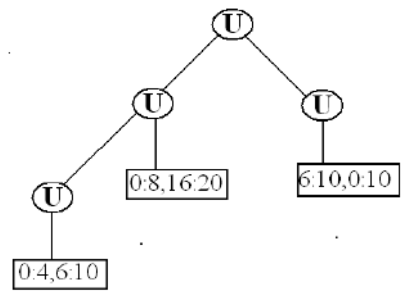

Figure 4 Tree representation of an Individual

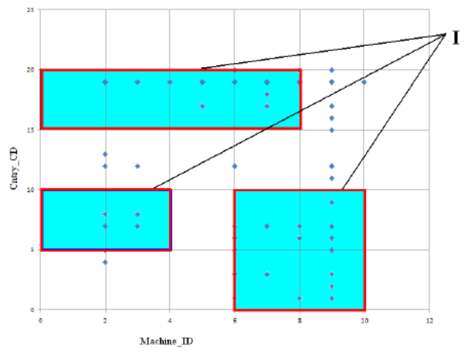

Figure 5 Rules for individual in Figure 4

Crossover: Experimental results are taken on consideration of rule-level crossover by swapping rules between individuals thus producing two new individuals.
Smart Mutation: Experimental results are taken by examining the densities of the cells that surround the target rule and performs enlarge mutation towards the denser cells. Another variant of mutation is shrinking whereby a rule is diminished by one bin with respect to a certain dimension *m*.
Repair Operator: Overlaps occur when one of the above operators produces a change in an existing rule, or an addition of a new rule, that shares at least one cell with an existing rule within the same individual. As mentioned earlier, overlaps are prohibited by [10]. Therefore, this paper introduces a novel repair operator that results in smoother detection by reforming the overlapping rules into new ones that align better with the distribution of the data points as shown in [10].
IFitness Function: This paper focuses on three main factors to achieve good solutions. It attempts to find a solution that captures as much of the data set as possible, thus high coverage. It also tries to identify gatherings of dense areas in the MBHR by looking for solutions in the form of dense individuals composed of dense rules that contain dense cells. Finally, as shown in equation (2), it attempts to avoid complex individuals by having a bias in favor of those having a smaller number of member rules.
(a). Maximize *IFcoverage*(*I*) which is the number of data points in an individual over the total number of data points.[4]
(b). Maximize *IFdensity*(*I*), which is product of the individual, rule, and cell densities.
(c). Minimize *IFsize*(*I*), which is the size of the individual by exerting parsimony pressure on large individuals[12].

$$IFitness(I) = \frac{IFCoverage(I)\ x\ IFdensity(I)}{IFSize(I)} \qquad (2)$$

Selection Operator and Elitism: The selection operator is based on tournament selection with a tour size of three. Furthermore, this paper adopts one-individual elitism, whereby in every iteration, the best performer is preserved for the next generation.
Main Algorithm: The GP that is run on each slave processor is summarized in [10]. After each operator is applied, the fitness of resulting individuals is evaluated.
E. Incremental Merge Engine
After the discovered points are reported back to the master or the server, which traverses those discovered cells; assigning them cluster labels based on their neighborhoods in the database D'. This is driven by the algorithm shown below, resulting in the final solution.
Algorithm:
1. Determine the changed return cells that contain the returned data points, denoted the Δ*returned cells*.
2. Label all cells of the Clusters of Database D as "No Label".
3. Label all changed returned cells with "Δ Label".
4. Traverse returned cells. For each cell check neighbor cells, as follows:
If neighbor cell does not belong to returned cells and is not labeled, skip to the next neighbor, unless that cell is denser than the average density of returned cells and "No Label" cell, in which case both that and the current cell are assigned same label.
Else if neighbor cell is labeled with *L*1, assign current cell with the same label as neighbor, unless the current cell is labeled with a different label *L*2, then change both labels to *L*3.
Else if all neighbors are labeled with "No Label" or "Δ Label", then assign the current label.
5. Return to Step 3 until all returned cells are processed.
6. Assign different clusters to different labels. The algorithm IPYRAMID implemented is summarized below:
Master/ Server Processor
1. Conduct incremental binning on MBR.
2. Perform incremental geometric division.
3. Send each subset to a different slave/ client processor.
4. Receive *p* resulting subsets of discovered data points from *p* slaves. Determine cells that contain returned points. Mark those cells as Δ*returned cells*.
5. Merge returned cells along with the cells of the clusters of database D into global solution that labels every cell with a cluster label.
Slave/ Client Processor
1. Receive a data subset P of ΔD from master processor. Perform quantization on local data.
2. Run genetic program on the local data points in P (on current slave processor).
3. After algorithm finishes, send points in discovered cells to master/ server processor.

4. RESULTS

In this section, experiments made to show the performance of the proposed approach are described. They were implemented in Java on a personal computer with Intel Pentium IV 3.20GHz and 1GB RAM. IPYRAMID requires 1 Server and 4 Clients, in all 5 Processors to implement Parallel Genetic Programming to detect Clusters.

A dynamic database of having two incremental datasets dataset1 (1500 data points) and dataset2 (8500 data points) with increments of 100 and 1500 data points respectively, is used to test the performance of IPYRAMID algorithm. Experimental results are as shown in Figure 6.

From Figure 6 top part is for dataset1 and bottom is dataset2. After bulk updates (insertion and deletions) of 100 or 1500 data points the clusters discovered are as shown in Figures below. We can see that by dividing the data set onto multiple processors each of which executes a genetic program that uses a flexible individual representation leading to arbitrary shapes.

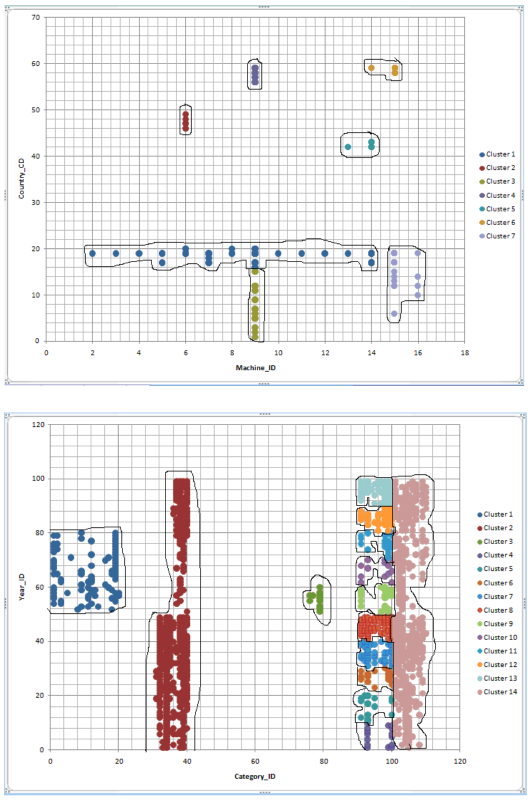

Figure 6 Arbitrary shaped Cluster Detection

The genetic program also utilizes a density based fitness function that helps avoid outliers by either considering a new cluster or merging it into an existing one as shown in Figure 7.

From Figure 8 and Figure 9 it can be analyzed that the number of clusters are reduced due to the updates (deletion). This proves that IPYRAMID incremental merge engine can handle deletion of records efficiently.

The results have shown that detection remained fairly similar even when crucial parameters such as the genetic algorithm population size, number of rules per individual, and the genetic operator percentages were changed as shown in Figure 7 for dataset2.

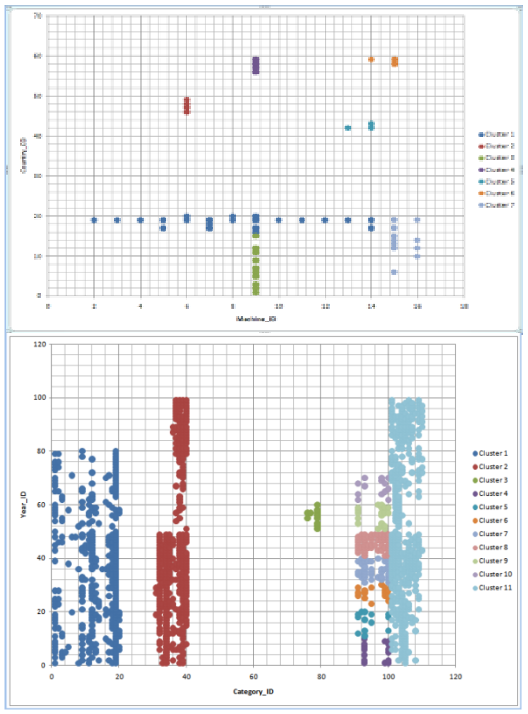

Figure 7 Outlier Handling

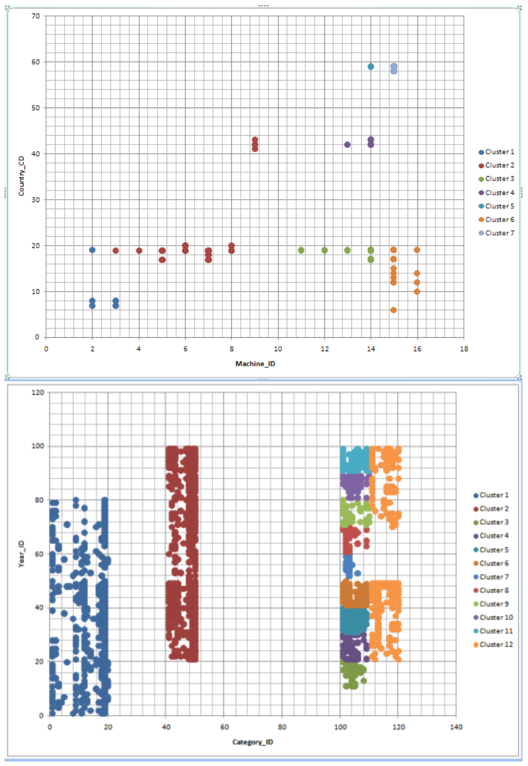

Figure 8 Cluster Detection with Updates (deletions)

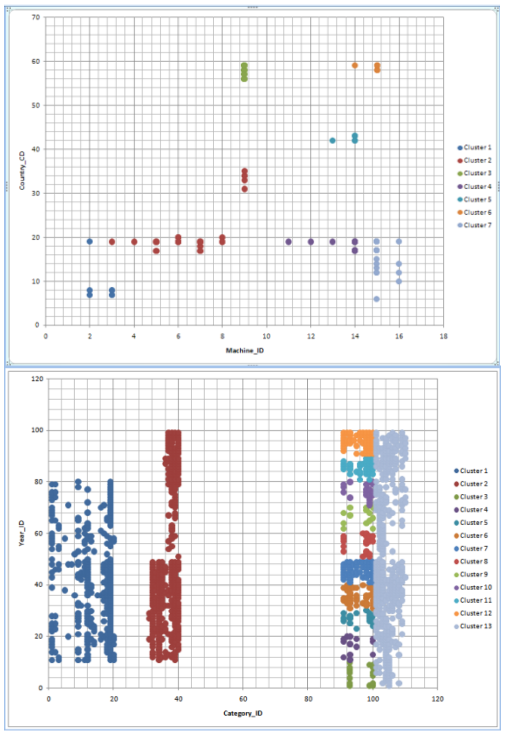

Figure 9 Cluster Detection with Updates (Additions)

5. CONCLUSION AND FUTURE WORK

This paper proposed an approach to clustering large data sets called IPYRAMID, which improves the approach by employing a hybrid combination of GP's global search and strong representational capabilities along with a powerful density-aware multi-objective fitness function. Use of data parallelism is increases the speed of execution. The proposed approach can handle outliers properly as well as it can handle deletion of records from database with high efficiency which is not taken care in most of the incremental clustering algorithms. Experimental results demonstrated that IPYRAMID detects clusters of arbitrary shapes and is immune to outliers, and independent of the order of input. In addition, it does not require prior knowledge of the number of clusters, and its inherent data parallelism allows it to have better performance than its sequential counterpart.

Possible Avenue for future research is to performing additional experiments to assess various aspects of cluster detection such as exploring the use of rules with variable shapes, not strictly rectangular, and using other data sets such as spatial or categorical, as well as other forms of parallelism.

6. REFERENCES

[1] Cheung D. W., Han J., Ng V. T., Wong Y.: "Maintenance of Discovered Association Rules in Large Databases: An Incremental Technique", Proc. 12th Int. Conf. on Data Engineering, New Orleans, USA, 1996, pp. 106-1 14.

[2] Feldman R., Aumann Y., Amir A., Mannila H.: "Efficient Algorithms for Discovering Frequent Sets in Incremental Databases", Proc. ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, Tucson, *AZ,* 1997, pp. 59-66.

[3] Ester M., Wittmann R.: "Incremental Generalization for Mining in a Data Warehousing Environment", Proc. 6th Int. Cod on Extending Database Technology, Valencia, Spain, 1998, in: Lecture Notes in Computer Science, Vol. 1377, Springer, 1998, pp. 135-152.

[4] Ester. M., Kriegel. H.-P. Sander. J. et. al. Incremental Clustering for Mining in a Data Warehousing Environment. In: Gupta. A., Shmueli. 0.. Widom. J., eds. Proceedings of the 24& International Conference on Very .Lyge Data Bases. New York: Morgan Kaufmann Publishers Inc., 1998.323-333.

[5] Chen Zhuo, Liu Xiang-shuang, Zhuang Xiao-dong (2007), "A Fast Incremental Clustering Algorithm Based on Grid and Density", *Third International Conference on Natural Computation (ICNC 2007)* ISBN: 0-7695-2875-9/07 IEEE 2007, 207 – 211.

[6] Chen, An, Chen Ning, "An Incremental Grid Density-Based Clustering Algorithm in Large Spatial Databases", http://www.jos.org.cn/ch/reader/download_pdf.aspx?file_no = 20020101&year_id=2002&quarter_id=1&falg=1.

[7] Tao Li, Saabjot S. Anand (2008) HIREL: An Incremental Clustering Algorithm for Relational Datasets" *2008 Eighth IEEE International Conference on Data Mining,* 887 – 892.

[8] Yong-Feng Zhou, Qing-Bao Ln, Shu Deng, Qing Yang (2002) "An Incremental Outlier Factor Based Clustering Algorithm" *Proceedings of the First International Conference on Machine Learning and Cybernetics,* ISBN: 0-7803-7508-4, 1358 - 1361 vol.3.

[9] Li Xiaohong, Luo Min (2009) , "*GAKC*: A new GA-based *k* clustering algorithm", Second International Symposium on Information Science and Engineering, ISBN: 978-0-7695-3991-1, IEEE 2009, 334 – 338.

[10] Tout, S., Sverdlik, W., & Sun, J. (2006). "Parallel Hybrid Clustering using Genetic Programming and Multi-objective Fitness with Density (PYRAMID)", *Proceedings of the 2006 International Conference on Data Mining* (*DMIN'06*), *Las Vegas, NV, USA,* 197-203.

[11] Koza, J.R. (1991). "Evolving a Computer Program to Generate Random Numbers using the Genetic Programming Paradigm", *Proceedings of the Fourth International Conference on Genetic Algorithms,* La Jolla, CA, 37-44.

[12] Karypis, G., Han, S., & Kumar, V. (1999). "Chameleon: A Hierarchical Clustering using Dynamic Modeling.*" IEEE Computer: Special Issue on Data Analysis and Mining, 32*(8), 68-75.