International Journal of Innovative Research in Computer Science & Technology (IJIRCST)
ISSN: 2347-5552, Volume-2, Issue-3, May-2014

Fast and Highly Scalable Multiresolution Linear Word based Clustering in Multidimensional data

P.Rubi , M.Govindaraj

***Abstract*—Clustering problems are well known in database literature for their use in numerous applications. Multidimensional data always is a challenge for clustering algorithms. The Halite, fast and scalable clustering method that looks for clusters in subspaces of multidimensional data. The tree root corresponds to a hypercube embodying the full data set. The next level divides the space in a set of 2D hypercube. The resulting hypercube are divided again, generating the tree structure. Bump Hunting task refers to apply for each level of the Counting-tree one d-dimensional Laplacian mask over the respective grid to spot bumps in the respective resolution. Specifically the main contributions of Halite are: Scalability: it is linear in time and space regarding the data size and dimensionality of the clusters' subspaces. Usability: it is deterministic, robust to noise, doesn't take the number of clusters as an input parameter, and detects clusters in subspaces generated by original axes or by their linear combinations, including space rotation. Effectiveness: it is accurate, providing results with equal or better quality. It is achieved through word based approach Generality: it includes a soft clustering approach.**

***Index Terms*— Bump Hunting, Correlation Connected Objects, Harp , Spotting clusters .**

I. INTRODUCTION

The new Halite method for linear clustering, a fast and scalable algorithm that spots clusters in subspaces of multidimensional data using a top down strategy. It analyzes the point distribution in the "full dimensional"[3] space by performing a multiresolution, recursive partition of that space, which helps finding clusters covering regions with varying sizes, shapes, density, correlated axes, and number of points. Existing methods are typically superliner[2] in space or time. It improves the basic Halite0 by providing an optimized implementation strategy for the Counting-tree, even for the case when it does not fit in main memory. The Halite0 algorithm has linear space complexity the number of points, axes and clusters. Thus, for large data sets, the use of Operational System's disk cache may become a considerable bottleneck. In order to overcome this problem, Halite has a table-based implementation that never uses disk cache, regardless of the input data set. Therefore, it allows us to efficiently analyze large amounts of data. The idea is to represent the Counting-tree by tables stored in main memory. Each table represents one tree level, by storing in key/value entries the data related to all nonempty cells of that level. Remember that Halite0 uses cells with the structure where loc is the cell spatial position inside its parent cell, n is the number of points in the cell, p[ ] is an array of half-space counts, used Cell is a Boolean flag, and ptr is a pointer to the next tree level.

Manuscript received May 12, 2014.

P.**Rubi**, Computer Science and Engineering Department, Bharathidasan University, Tiruchirappalli/Tamilnadu, India, 9790534573., (e-mail: rubi.joyce04@gmail.com).

M.Govindaraj, Computer Science and Engineering Department, Bharathidasan University, Tiruchirappalli/Tamilnadu, India, 9443597246., (e-mail: mgr@bdu.ac.in).

II. LITERATURE SURVEY

A. Finding Non-Redundant Statistically Significant Regions in High Dimensional Data: a Novel Approach to Projected and Subspace Clustering

Projected and subspace clustering algorithms search for clusters of points in subsets of attributes. Projected clustering computes several disjoint clusters, plus outliers, so that each cluster exists in its own subset of attributes. Subspace clustering enumerates clusters of points in all subsets of attributes, typically producing many overlapping clusters [5][6]. One problem of existing approaches is that their objectives are stated in a way that is not independent of the particular algorithm proposed to detect such clusters. A second problem is the definition of cluster density based on user-defined parameters, which makes it hard to or whether they actually stand out in the data in a statistical sense. We propose a novel problem formulation that aims at extracting axis-parallel regions that stand out in the data in a statistical sense. The set of axis-parallel, statistically significant regions that exist in a given data set is typically highly redundant. Therefore, formulate the problem of representing this set through a reduced, non-redundant set of axis parallel, statistically significant regions as an optimization problem. Exhaustive search is not a viable solution due to computational infeasibility, and we propose the approximation algorithm STATPC. Our comprehensive experimental evaluation shows that STATPC significantly outperforms existing projected and subspace clustering algorithms in terms of accuracy.

B. Computing Clusters of Correlation Connected Objects

The detection of correlations between different features in a set of feature vectors is a very important data mining task because correlation indicates a dependency between the features or some association of cause and between them. This association can be arbitrarily complex, i.e. one or more features might be dependent from a combination of several other features. Well-known methods like the principal components analysis (PCA) can perfectly and correlations which are global, linear, not hidden in a set of noise vectors, and uniform. In many applications such as medical diagnosis, molecular biology, credit card fraud, monitoring of criminal activities, time sequences, or electronic commerce, however, correlations are not global since the dependency between features can be different indifferent subgroups of the set. In this paper, we propose a method called 4C (Computing Correlation Connected Clusters) to identify local subgroups of the data objects sharing a uniform but arbitrarily complex correlation. Our algorithm is based on a combination of PCA and density-based clustering. Our method has a determinate result and is robust against noise.

C. Iterative Projected Clustering by Subspace Mining

Irrelevant attributes add noise to high-dimensional clusters and render traditional clustering techniques inappropriate. In this paper, we realize the analogy between mining frequent item sets and discovering dense projected clusters around random points. Based on this, we propose a technique that improves the efficiency of a projected clustering algorithm. Our method is an optimized adaptation of the frequent pattern tree growth method used for mining frequent item sets [8]. We propose several techniques that employ the branch and bound paradigm to efficiently discover the projected clusters. An experimental study with synthetic and real data demonstrates that our technique significantly improves on the accuracy and speed of previous techniques CLUSTERING partitions a collection of objects S into a set of groups, such that the similarity between objects of the same group is high and objects from different groups are dissimilar. Clustering finds many applications in marketing image analysis, bioinformatics, document classification, indexing, etc. In many such applications, the objects to be clustered are represented by points in a high-dimensional space, where each dimension corresponds to an attribute/feature and the feature value of each object determines its coefficient in the corresponding dimension. A distance measure between two points is used to measure the dissimilarity between the corresponding objects.

D. Harp: A Practical Projected Clustering Algorithm

The main theme of this thesis is to study the feasibility of extracting useful information from gene expression profiles by a relatively new data mining approach known as projected clustering. This is a multi-disciplinary topic, involving research efforts from areas such as data mining, applied mathematics, genetics and genomics. This chapter consists of two main parts. The first part provides a short overview of the objectives and methods of data mining, and quickly moves to the topic of clustering and finally projected clustering. The second part starts with an introduction to bioinformatics in general, and then narrows down to microarray technology and gene expression profiles, and finally focuses on some clustering methods proposed for gene expression profile analysis. Clustering is a process to group similar objects together. Before directly jumping into the detailed discussion of clustering, it is instrumental to spend some time on the format of data that can be clustered. In the following definitions, the preferred terms of some concepts appear first and some alternative terms that have the same or similar meanings are listed in brackets. A cluster is defined as a non-empty subset of the objects, which are called the members of the cluster. The centroid of a cluster is a virtual object with its projected value on each dimension equal to the arithmetic mean of the projected values of all the cluster members on the dimension1 Two clusters are disjoint (non-overlapping) if they contain no common objects, and a set of clusters is disjoint if all clusters in it are pair wise disjoint. A clustering algorithm is said to produce disjoint clusters if the clusters that it produces are always disjoint. Otherwise, the algorithm is said to produce non-disjoint clusters, even the clusters are not always non-disjoint. Some objects that do not fit into any cluster can be left UN clustered. They are called the outliers of the dataset, which are reported on a separate outlier list.

E. Locally adaptive metrics for clustering high dimensional data

Clustering suffers from the curse of dimensionality, and similarity functions that use all input features with equal relevance may not be effective. We introduce an algorithm that discovers clusters in subspaces spanned by different combinations of dimensions via local weightings of features. This approach avoids the risk of loss of information encountered in global dimensionality reduction techniques, and does not assume any data distribution model. Our method associates to each cluster a weight vector, whose values capture the relevance of features within the corresponding cluster. We experimentally demonstrate the gain in performance our method achieves with respect to competitive methods, using both synthetic and real datasets. In particular, our results show the feasibility of the proposed technique to perform simultaneous clustering of genes and conditions in gene expression data, and clustering of very high-dimensional data such as text data Clustering suffers from the curse of dimensionality problem in high-dimensional spaces. In high dimensional spaces, it is highly likely that, for any given pair of points within the same cluster, there exist at least a few dimensions on which the points are far apart from each other. Furthermore, several clusters may exist in different subspaces, comprised of different combinations of features. In many real world problems, in fact, some points are correlated with respect to a given set of dimensions, and others are correlated with respect to different dimensions. Each dimension could be relevant to at least one of the clusters. The problem of high dimensionality could be addressed by requiring the user to specify a subspace for cluster analysis. More importantly,

correlations that identify clusters in the data are likely not to be known by the user.

F. Locally adaptive metrics for clustering high dimensional data

Clustering suffers from the curse of dimensionality, and similarity functions that use all input features with equal relevance may not be effective. We introduce an algorithm that discovers clusters in subspaces spanned by different combinations of dimensions via local weightings of features. This approach avoids the risk of loss of information encountered in global dimensionality reduction techniques, and does not assume any data distribution model. Our method associates to each cluster a weight vector, whose values capture the relevance of features within the corresponding cluster. We experimentally demonstrate the gain in performance our method achieves with respect to competitive methods, using both synthetic and real datasets. In particular, our results show the feasibility of the proposed technique to perform simultaneous clustering of genes and conditions in gene expression data, and clustering of very high-dimensional data such as text data Clustering suffers from the curse of dimensionality problem in high-dimensional spaces. In high dimensional spaces, it is highly likely that, for any given pair of points within the same cluster, there exist at least a few dimensions on which the points are far apart from each other. Furthermore, several clusters may exist in different subspaces, comprised of different combinations of features. In many real world problems, in fact, some points are correlated with respect to a given set of dimensions, and others are correlated with respect to different dimensions. Each dimension could be relevant to at least one of the clusters. The problem of high dimensionality could be addressed by requiring the user to specify a subspace for cluster analysis. More importantly, correlations that identify clusters in the data are likely not to be known by the user.

III. System Architecture Design

The structure of the system which comprises system components, the externally visible properties of those components, the relationships between them, and provides a plan from which products can be procured, and systems developed, that will work together to implement the overall system.

As far as the data allocation strategies are concerned, our work is mostly relevant to watermarking that is used as a means of establishing original ownership of distributed objects. Our approach and watermarking are similar in the sense of providing agents with some kind of receiver identifying information. By its very nature, a watermark modifies the item being watermarked. If the object to be watermarked cannot be modified, then a watermark cannot be inserted. In such cases, methods that attach watermarks to the distributed data are not applicable.

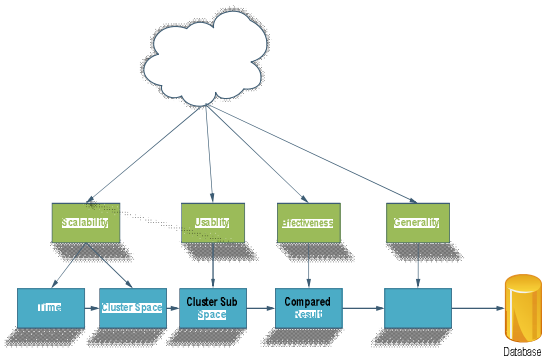

Fig 1: System Design

IV. Classification & Retrieval

A. Dataset Partition

The data set partition by ensuring that each point belongs to at most one cluster. The data contain a pair of clusters that overlap, making any data set partitions not a good choice, provided that the points in light-gray should belong to both clusters. In cases like that, the so-called soft clustering methods are more appropriate, since they allow points in the overlapping spaces to belong to more than one cluster**.**

B. Spotting clusters

It generalizes the structure of these systems to the d-dimensional case in order to describe clusters of any shape and size, hence its name Halite. Halite uses spatial convolution masks in a novel way to efficiently detect density variations in a multi scale grid structure that represents the input data, thus spotting clusters. These masks are extensively used in digital image processing to detect patterns in images Spot Point: To encode an input data set, selecting a minimal code length.

C. Spot points

The Minimum Description Length (MDL) principle is also used in a novel way. Its idea is to encode an input data set, selecting a minimal code length. Halite uses MDL to automatically tune a density threshold. The data distribution, which helps spotting the clusters' subspaces. Finally, Halite includes a compression-based analysis to spot points that most likely belong to two or more clusters that overlap in the space. It allows soft clustering results.

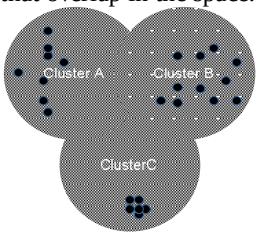

Fig 2: Spot points

D. Bump Hunting

The "Bump Hunting" spots cells on the "positive" side of the largest, local density changes, but it does not guarantee

that these changes are statistically significant – some of them could have been created by chance. Even when analyzing only points randomly distributed through a d-dimensional space, the mask will return bumps. This strategy allows Halite0 to identify the best resolution to spot each bump, as this resolution is the one in which the respective local density change is more intense, thus, it avoids overestimating the clusters bounds and spotting only clusters' borders. It also spots the right moment to stop the "Bump hunting" it stops once, in all resolutions, the clearest bump was potentially created by chance.

E. Counting Tree

The Counting-tree by tables stored in main memory and/or disk. Each table represents one tree level, by storing in key/value entries the data related to all nonempty cells of that level. Remember that Halite0 uses cells with the structure where loc is the cell spatial position inside its parent cell, n is the number of points in the cell, P is an array of half-space counts, used Cell is a Boolean flag, and ptr is a pointer to the next tree level. For Halite, this cell structure was slightly modified. Here, the pointer ptr does not exist and loc has the absolute position for the cell. In a key/value pair, loc is the key, and the other attributes form the value.

The data storage for Halite: The tables shown consist in a different way of storing the Counting tree of Fig. 3c. Both approaches represent the same data, the 2D data set from the one.

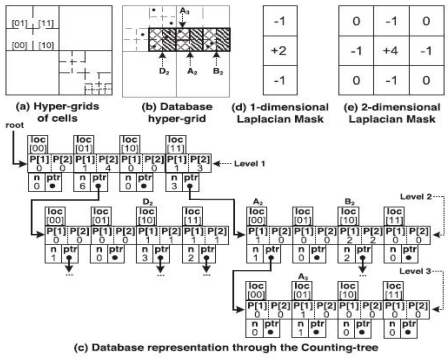

Fig 3: Counting Tree Construction

V. METHODOLOGY AND IMPLEMENTATION

A. Finding β -Clusters

The second phase of Halite0 uses the counts in the tree to spot bumps in the space with all axes that indicate β-clusters. Halite0 looks for β-clusters by applying convolution masks over each level of the Counting-tree. We name this task "Bump Hunting." The masks are integer approximations of the Laplacian filter, a second-derivative operator that reacts to transitions in density. Figs. 3d and 3e show examples of 1D and 2D Laplacian masks, respectively. In a nutshell, the "Bump Hunting" task refers to: to apply for each level of the Counting-tree one d-dimensional Laplacian mask over the respective grid to spot bumps in the respective resolution. Fig. 4a illustrates the process on a toy 1D data set with grids in five resolutions. To spot bumps, for each cell of each resolution, the 1D mask is applied as follows: multiply the cell's count of points by the mask's center value. Multiply the point count of each neighbor of the cell by the respective mask value, and get the convoluted value for the cell by summing the results of the multiplications. After visiting all cells in one resolution, the cell with the largest convoluted value represents the clearest bump in that resolution, i.e., the largest positive magnitude of the density gradient. In Fig. 4a, for each resolution, one dark-gray arrow points to this cell.

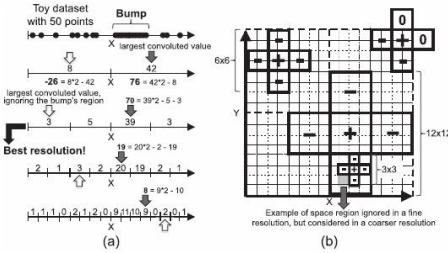

Fig 4: Coarser Resolutions

Algorithm : Building the local-correlation clusters.
Input: matrices of β-clusters L, U and V,
number of β-clusters ${}_{\beta}k$
Output: set of local-correlation clusters C,
number of local-correlation clusters ${}_{\gamma}k$

1: identify all pairs of β-clusters that overlap;
2: merge each pair of overlapping β-clusters into a single cluster, including merged clusters in further mergers;
3: define the points that belong to each merged cluster, as the ones that belong to at least one of its β-clusters;
4: define the axes relevant to each merged cluster, as those relevant to at least one of its β-clusters;
5: C = the merged clusters;
6: ${}_{\gamma}k$ = the number of merged clusters;

B. Halite Algorithm

The Halite method for local correlation clustering. It improves the basic Halite0 by providing an optimized implementation strategy for the Counting-tree, even for the case when it does not fit in main memory. The Halite0 algorithm has linear space complexity the number of points, axes and clusters. However, using the recommended configuration, the amount of memory required by it in our previous experiments of varied between 25 and 50 percent of the data size, depending on the point's distribution. Thus, for large data sets, the use of Operational System's disk cache may become a considerable bottleneck. In order to overcome this problem, Halite has a table-based implementation that never uses disk cache, regardless of the input data set. Therefore, it allows us to efficiently analyze large amounts of data. For Halite, cell structure was slightly modified. Here, the pointer ptr does not exist and loc has the absolute position for the cell. In a key/value pair, loc is the key, and the other attributes form the value. Fig. 5 exemplifies the data storage for Halite. The tables shown consist in a different way of storing the Counting tree both approaches represent the same data, the 2D data set.

Illustration of our soft clustering method Halites: Beta-clusters may stay apart if they are incompatible,

resulting in soft clustering are merged together. The compression-based formulas of Halite's automatically make the right choice.

C. BETA Cluster Algorithm

The "Bump Hunting" task allows one to efficiently spot the clearest bumps in a data set. However, two open questions still prevent its use for clustering: 1) what is the best resolution to spot each bump? And 2) when should the "Bump Hunting" stop? To give an intuition on both questions, we use again the 1D data with grids in five resolutions from Fig. 4a. A dark-gray arrow points to the cell with the largest convoluted value, which describes the clearest bump in the data for each resolution.

Notice: a procedure to automatically identify the best resolution to spot a bump is still missing, which refers to question one. In the example, the bump is best described in the third coarsest resolution, as its bounds are overestimated in the two coarser resolutions and only its borders are spotted in the two finer resolutions. In the example, ignoring the bump's region, a white arrow points to the cell with the largest convoluted value for each resolution, which, clearly, doesn't lead to a cluster. Thus, the "Bump Hunting" should stop.

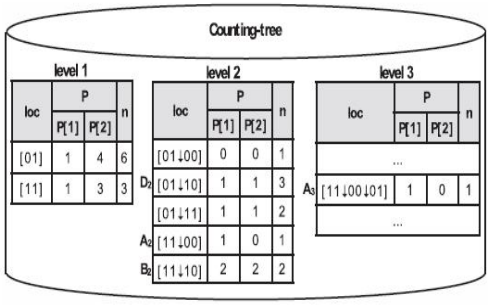

Fig 5: Storing the counting tree

Notice that the "Bump Hunting" spots cells on the "positive" side of the largest, local density changes, but it does not guarantee that these changes are statistically significant some of them could have been created by chance. Even when analyzing only points randomly distributed through a d-dimensional space, the mask will return bumps. Therefore, we propose to automatically answer both questions previously posed by ignoring bumps that, potentially, were created by chance, assuming that only statistically significant bumps lead to clusters. This strategy allows Halite0 to identify the best resolution to spot each bump, as this resolution is the one in which the respective local density change is more intense, thus, it avoids overestimating the clusters bounds and spotting only clusters' borders. Notice that the "Bump Hunting" spots cells on the "positive" side of the largest, local density changes, but it does not guarantee that these changes are statistically significant some of them could have been created by chance. Even when analyzing only points randomly distributed through a d-dimensional space, the mask will return bumps. Therefore, we propose to automatically answer both questions previously posed by ignoring bumps that, potentially, were created by chance, assuming that only statistically significant bumps lead to clusters. This strategy allows Halite0 to identify the best resolution to spot each bump, as this resolution is the one in which the respective local density change is more intense, thus, it avoids overestimating the clusters bounds and spotting only clusters' borders.

The evaluation of a clustering result the quality of the uncovered relevant axes is similar. We also computed the harmonic mean of the averaged precision for all found clusters and the averaged recall for all real clusters, but we exchanged the sets of points in the two last equations, precision and recall, by sets of axes. We name this harmonic mean as Subspaces Quality. In the cases where a technique does not find clusters in a data set, the value zero is assumed for both qualities. Halite uses fixed input parameter values, as defined in Halite0 was tuned in the same way.

The other algorithms were tuned as follows: ORCLUS, LAC, EPCH, CFPC, and HARP received as input the number of clusters present in each data set. Also, the known percent of noise for each data set was informed to HARP. The extra parameters of the previous works were tuned as in their original authors' instructions. LAC was tested with integer values from 1 to 11, for the parameter 1=h. However, its runtime differed considerably with distinct values of 1=h. Thus, a time out of 3 hours was specified for LAC executions. All configurations that exceeded this time limit were interrupted. EPCH was tuned with Fig.7 integer values from 1 to 5 for the dimensionalities of its histograms and several real values varying from 0 to 1 were tried for the outliers' threshold.

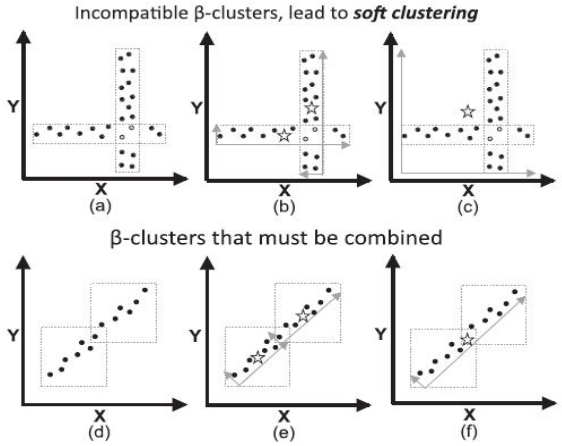

Fig 6: Illustration of Soft Clustering method

Its parameter was defined as suggested in COPAC's original publication and received the smallest value between k and the known size of the smallest cluster present in each data set .In a nutshell: 1) we initially created axes

aligned, elliptical clusters of random sizes that follow normal distributions with random means and random variances in at least 50 percent of the axes, spreading through at most 15 percent of these axes domains. In other axes, the irrelevant ones, all clusters follow the uniform distribution, spreading through the whole axes domains; and 2) an optional data rotation allowed creating clusters not aligned to the original axes. In this step, each data set was rotated four times in random planes and random degrees. The behavior of our techniques varies based on two parameters: alpha and H. This section analyses how they affect our methods. We varied both parameters for Halite, Halite0, and Halite's to maximize the techniques. Each data set and technique, we modified the best configuration, changing one parameter at a time, and analyzed the technique's behavior.

For example, when varying H for a data set and a technique, the value of α was fixed at the value in the respective best configuration. The tested values of α and H vary from 1:0E _ 3 to 1:0E _ 160 and from four to 80, respectively. We report the results of Halite0. **Figs. 7** show results .Notice: the values of _ that led to the best Quality vary from 1:0E _ 5 to 1:0E _ 20 and the runtime was barely affected by changes in α. Concerning H, Figs. 7n and 7o show that the Quality does not increase for H higher than four. But, the runtime increased as expected w.r.t. H. Thus, the methods were compared in a scenario with high probability of cluster overlap. The images, available at "geoeye.com", amount to 17 MB. Each image was divided into equalsized rectangular tiles, from which Haar wavelets features were extracted. The process led to a 10D data set of 14,336 points.

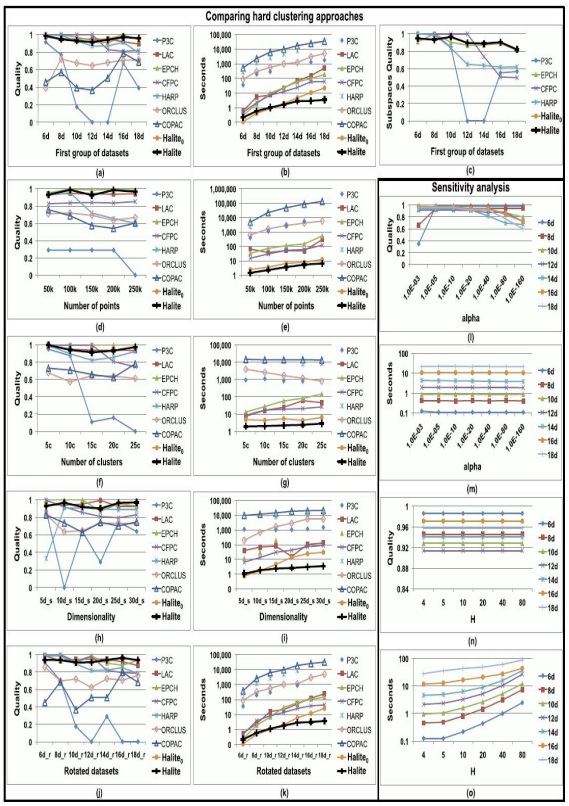

Fig 7: Halite is shown in black vertical crossing lines

D. *Tests on large datasets*

The purpose of this experiment was to test the scalability of the *Halite* algorithm in clustering very large real world data sets. We selected a large data set from a health insurance database. The data set consists of 500000 records, each being described by 34 categorical attributes in which 4 have more than 1000 categories each. We tested two scalabilities of the algorithm using this large data set. The first one is the scalability of the algorithm against the number of clusters for a given number of objects and the second is the scalability against the number of objects for a given number of clusters. If we plot in the figures, it would represent the average time performance of 5 independent runs.

These results are very encouraging because they show clearly a linear increase in time as both the number of clusters and number of records increase. Clustering half a million objects into 100 clusters took about an hour, which is quite acceptable. Compared with the results of clustering data with mixed values this algorithm is much faster than its previous version because it needs many less iterations to converge. The above soybean disease data tests indicate that a good clustering result should be selected from multiple runs of the algorithm over the same data set with different record orders and/or different initial modes. This can be done in practice by running the algorithm in parallel on a parallel computing system. Other parts of the algorithm such as the operation to allocate an object to a cluster can also be parallelized to improve the performance.

E. *Accuracy*

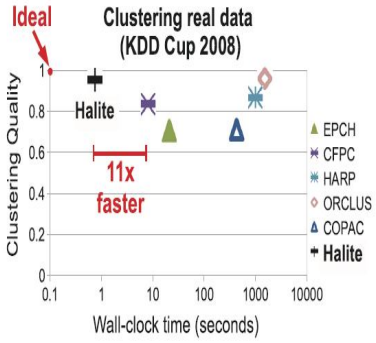

Fig 8: Halite's accuracy measured graph

The accuracy in up to 35 percent. Finally, we report experiments in a real scenario where soft clustering is desirable. Halite analyzed 25D data for breast cancer diagnosis (KDD Cup 2008) at least 11 times faster than five previous works, increasing their accuracy input 35 percent.

Halite0 's clustering accuracy improves a little when we use having nonzero values at all elements, but the time required increases too much – in the order of as compared to when using masks of order 3 having nonzero values only at the center and the facing elements. To explain the success of the latter masks, we point to a fact: the multiresolution allows Halite0 to use masks of a low order to efficiently "simulate" masks of higher orders. Fig. gives

an intuition on this fact by illustrating masks applied to grids in distinct resolutions over a 2D space.

Word-based approach to build clusters. It first forms initial clusters of the documents, with each cluster representing a single word. For instance, Word based forms a cluster for the word 'tiger' made up of all the documents that contain the word 'tiger'. After that, WBSC merges similar clusters are similar if they contain the similar set of documents using a hierarchical based approach until some stopping criterion is reached. At the end, the clusters are displayed based on the words associated with them.

F. Execution Time

We also measured the execution time of various algorithms. Fig. 9 gives the comparison of execution time.

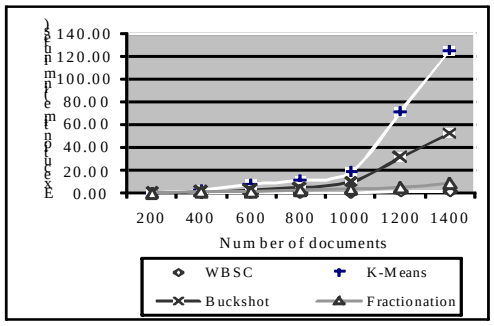

Fig 9: Execution time of various algorithms

As the graph shows, WBSC outperforms almost all other algorithms in execution time, especially as the number of documents increases.

We also tested WBSC [17] on the results from Web search engines. We downloaded documents returned from the Google search engine (www.google.com) and we apply WBSC on them. Limitation on space prohibits us from showing all the results. Here we show some of the clusters found by WBSC by clustering the top 100 URLs returned from searching the term "cardinal"[18][19].The categories correspond to the common usage of the word "cardinal" in documents over the Web Figure 10 shows some of the clusters formed by WBSC.

Cluster results: Keywords and sample documents	Related topic
Bishops, Catholic, world, Roman, Church, cardinals, College, Holy 1. Cardinals in the Catholic Church Catholicpages.com 2. The Cardinals of the Holy Roman Church	Roman Catholic Church Cardinals library
Dark, Bird, female, color 1. Cardinal Page 2. First Internet Bird Club	Cardinal Bird
Benes, rookie, players, hit, innings, runs, Garrett, teams, league, Saturday 1. St.Louis Cardinals Notes 2. National League Baseball - Brewers vs. Cardinals	St.Louis Cardinals (MLB team)

Fig 10: Clusters formed for search term 'cardinals' by WBSC

This shows that our algorithm is effective even with the web search results.

ACKNOWLEDGMENT

First of all I wish to express my profound thanks to God almighty for his abundant blessing. I would like to thank Mr.M.Govindaraj, Professor, Department of Computer Science and Engineering, Bharathidasan University, Tiruchirappalli for his cheerful encouragement helped me through seemingly endless details. Finally thanks to my parents, brother, sister and my friends for their wholehearted support throughout this project.

REFERENCES

[1] R.L.F.Cordeiro,A.J.M. Traina,C.Faloutsos and C. Traina Jr., ., "Finding Clusters in Subspaces of Very Large, Multi-Dimensional Data Sets," Proc. IEEE 26th Int'1 Conf.Data Eng.(ICDE),pp.625-636,2010.

[2] R.C. Gonzalez and R.E. Woods, Digital Image Processing, third ed. Prentice-Hall, Inc., 2006.

[3] P.D. Grunwald, I.J. Myung, and M.A. Pitt, Advances in Minimum Description Length: Theory and Applications (Neural Information Processing). The MIT Press, 2005.

[4] C. Traina Jr., A.J.M. Traina, C. Faloutsos, and B. Seeger,"Fast Indexing and Visualization of Metric Data Sets Using Slim-Trees," IEEE Trans. Knowledge Data Eng., vol. 14, no. 2, pp. 244-260, Mar./ Apr. 2002.

[5] C. Traina Jr., A.J.M. Traina, L. Wu, and C. Faloutsos, "Fast Feature Selection Using Fractal Dimension," Proc. 15th Brazilian Symp. Databases (SBBD), pp. 158-171, 2000.

[6] H.-P. Kriegel, P. Kro¨ger, and A. Zimek, "Clustering High-Dimensional Data: A Survey on Subspace Clustering, Pattern-Based Clustering, and Correlation Clustering," ACM Trans. Knowledge Discovery from Data, vol. 3, no. 1, pp. 1-58, 2009.

[7] C. Domeniconi, D. Gunopulos, S. Ma, B. Yan, M. Al-Razgan, and D. Papadopoulos, "Locally Adaptive Metrics for Clustering High Dimensional Data," Data Mining and Knowledge Discovery, vol. 14, no. 1, pp. 63-97, 2007.

[8] A.K.H. Tung, X. Xu, and B.C. Ooi, "Curler: Finding and Visualizing Nonlinear Correlation Clusters," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 467-478, 2005.

[9] C. Aggarwal and P. Yu, "Redefining Clustering for High-Dimensional Applications," IEEE Trans. Knowledge and Data Eng., vol. 14, no. 2, pp. 210-225, Mar./Apr. 2002 .

[10] E.K.K. Ng, A.W. chee Fu, and R.C.-W. Wong, "Projective Clustering by Histograms," IEEE Trans. Knowledge and Data Eng., vol. 17, no. 3, pp. 369-383, Mar. 2005.

[11] G. Moise, J. Sander, and M. Ester, "Robust Projected Clustering," Knowledge Information Systems, vol. 14, no. 3, pp. 273-298, 2008.

[12] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications," SIGMOD Record, vol. 27, no. 2, pp. 94- 105, 1998.

[13] C.C. Aggarwal, J.L. Wolf, P.S. Yu, C. Procopiuc, and J.S. Park, "Fast Algorithms for Projected Clustering," SIGMOD Record, vol. 28, no. 2, pp. 61-72, 1999.

[14] M.L. Yiu and N. Mamoulis, "Iterative Projected Clustering by Subspace Mining," IEEE Trans. Knowledge and Data Eng., vol. 17, no. 2, pp. 176-189, Feb. 2005.

[15] K. Yip, D. Cheung, and M. Ng, "Harp: A Practical Projected Clustering Algorithm," IEEE Trans. Knowledge and Data Eng., vol.

16, no. 11, pp. 1387-1397, Nov. 2004.

[16] G. Moise and J. Sander, "Finding Non-Redundant, Statistically Significant Regions in High Dimensional Data: A Novel Approach to Projected and Subspace Clustering," Proc. 14th ACM SIGKDD Int'l Conf. Knowledge Discovery Data Mining (KDD), pp. 533-541, 2008

[17] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, John W. Tukey, Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections, In *Proceedings of the Fifteenth Annual International ACM SIGIR Conference*, pp 318-329, June 1992.

[18] Dean, P. M. Ed., Molecular Similarity in Drug Design, Blackie Academic & Professional, 1995, pp 111 –137.

[19] D. R. Hill, A vector clustering technique, in: Samuelson (Ed.), *Mechanized Information Storage, Retrieval and Dissemination*, North-Holland, Amsterdam, 1968.