Proceedings of the 2007 IEEE Symposium on
Computational Intelligence and Data Mining (CIDM 2007)

Scalable Clustering for Large High-Dimensional Data Based on Data Summarization

Ying Lai
Computer Science
Illinois Institute of Technology
Chicago, IL 60616, U.S.A
laiying@iit.edu

Ratko Orlandic
Computer Science
Univ. of Illinois at Springfield
Springfield, IL 62703, U.S.A
rorla2@uis.edu

Wai Gen Yee
Computer Science
Illinois Institute of Technology
Chicago, IL 60616, U.S.A
yee@iit.edu

Sachin Kulkarni
Computer Science
Illinois Institute of Technology
Chicago, IL 60616, U.S.A
kulksac@iit.edu

Abstract

Clustering large data sets with high dimensionality is a challenging data-mining task. This paper presents a framework to perform such a task efficiently. It is based on the notion of data space reduction, which finds high density areas, or dense cells, in the given feature space. The dense cells store summarized information of the data. A designated partitioning or hierarchical clustering algorithm can be used as the second step to find clusters based on the data summaries. Using Kmeans as an example, this paper presents GARDEN-Kmeans, which performs data space reduction using Gamma Region DENsity partition, and utilizes Kmeans to cluster the summarized information. The experimental study shows that GARDEN-Kmeans executes several orders of magnitude faster than basic Kmeans and the recursive bisection Kmeans algorithm of CLUTO, while producing comparable clustering quality.

1. Introduction

Clustering is the process of partitioning a given data set into groups of similar objects. Data objects are generally interpreted as points in a multi-dimensional feature space. Objects within the same cluster should be similar according to a similarity metric. Objects in different clusters should be dissimilar under the same metric. High quality clustering should result in high intra-cluster similarity/relatedness and low inter-cluster similarity/relatedness.

Clustering is one of the most frequently used data mining techniques. It can be used in many applications, such as multimedia content-based retrieval, geographic and molecular data analysis, bioinformatics, etc.

The explosive size and dimensionality of data in contemporary data-mining applications call for efficient and scalable clustering algorithms. Clustering sampled data and clustering summarized data are two widely used approaches for scaling up existing clustering algorithms without inventing new clustering methods. There are several methods with the sampling-based approach [9, 13]. They differ in the sampling process. With the second approach, the clustering algorithm is applied only to a summary of data rather than the original data set. BIRCH [21], Scalable Kmeans [1], Data Bubbles [2, 22] and EMADS [7] are some well-known examples. However, these algorithms are not fully effective when clustering high dimensional data [6, 11]. This paper presents a new clustering framework with data summarization, which can scale to both high dimensionality and large size of data with comparable clustering quality.

In the rest of the paper, Section 2 reviews related work. In Section 3, we introduce a cell-based compression technique. Section 4 describes the proposed clustering algorithm GARDEN-Kmeans. Section 5 shows the experimental results. Section 6 concludes the paper.

2. Related Work

There are four major types of clustering algorithms: partitioning, hierarchical, density-based, and grid- (cell-) based. Partitioning algorithms, such as Kmeans [10], decompose the data set into K clusters. They iteratively assign membership of each data point and re-estimate cluster parameters. For example, a convergence can be reached when data points no longer change their memberships. Hierarchical clustering algorithms create a hierarchical decomposition of a given set of data, which is usually represented by a tree that splits the data iteratively into smaller subsets. Density-based algorithms continuously grow the cluster as long as the density in the "neighborhood" exceeds some threshold. DBSCAN [3] represents this type. Its key idea is that, for each point in a cluster, the neighborhood of a given radius must contain a minimum number of points. Cell-base (grid-based) algorithms divide the input space into hyper-rectangular cells and then merge the adjacent high-density cells to form clusters. STING [19], OptiGrid [6], O-Cluster [11] and $GARDEN_{HD}$ [18] are the examples of cell-based clustering algorithms.

In order to accommodate the increasing size and dimensionality of data, many scalable algorithms have been proposed in the past several years. BIRCH [21], ScalableKM [1], FREM [15], EMADS [7] are all based on a similar approach. It first compresses the original data set to certain forms of summary, and then performs partitioning clustering on the summarized information. BIRCH uses a balanced tree structure (*CF* tree) to store summarized sub-cluster information. *CF* refers to Clustering Feature, which is defined as the triple *CF* = (*N*, *LS, SS*), where *N* is number of the data points, *LS* is the linear sum of the data points, and *SS* is the square sum of the data points. BIRCH performs a sequential scan over all data points and builds a *CF*-tree similar to the construction of B^+-tree.

ScalableKM is similar to BIRCH. However, it separates data points into three types of sets. The first is *DS*, containing data points that unlikely change membership in the iterations of the Kmeans algorithm. The second is *CS*, referring to the "tight" sub-clusters of data points. The third is *RS*, containing any remaining data points. DataBubbles [2] uses a structure, called *Data Bubble*, to store the data summarization. It is specifically designed to speed up hierarchical clustering algorithms.

More recently, Jin et al. [7] proposed EMADS, which applies model-based clustering algorithms, like EM, on data summaries to speed up the traditional model-based clustering algorithms. The authors apply two types of data compression: BIRCH and Grid-based. Their experimental results demonstrate a superior speed-up over the traditional EM algorithm with little loss of accuracy.

In this paper, we propose a new framework, based on $GARDEN_{HD}$ clustering [18], using the similar approach as described in the previous paragraph. Taking GARDEN-Kmeans as an example, we demonstrate the merit of the proposed framework. Our method differs from other scalable clustering algorithms in that it employs a cell-based compression scheme which has the advantage in dealing with large high-dimensional data.

3. Data Space Reduction (Data Compression)

We define *data space reduction* as the process of reducing the empty regions in the data space so that only tight, high-density regions are retained. The goal of this process is to reduce the size of the empty region as much as possible; in other words, to make the dense regions as tight as possible so that they will not falsely merge distinct clusters. The data space reduction technique used in this paper has been originally developed in [18] as the first phase of the cell-based $GARDEN_{HD}$ clustering technique. In this paper, we modify the space reduction process to make it suitable for summarization-based clustering.

Our data space reduction technique is built on γ partitioning [16, 18], an efficient method to isolate high density areas with points, and several heuristics designed to deal with certain "false dense cells" that can bridge distinct clusters [18]. The γ partitioning technique recursively decomposes the space into equal size hyper-rectangles, called *γ regions*, in a way that can support the differentiation of data along all dimensions of a high-dimensional space. However, to avoid false dense cells, typically not all sides of a given region are split [18].

The data space reduction technique performs a recursive γ partitioning of sparse "live regions" until the densities of the resulting cells are above the user-defined density threshold. We define *live region* as the minimum bounding hyper-rectangle that holds all points in the given γ region. The processes of splitting a γ region and computing its live regions are very efficient [18]. The live regions whose density is above the user-defined threshold represent *dense cells*. The result of the data space reduction is a vastly reduced space represented by a set of dense cells, which can be used as the summary of original data. The detailed algorithms of γ partitioning, live-region identification and data space reduction appear in [18].

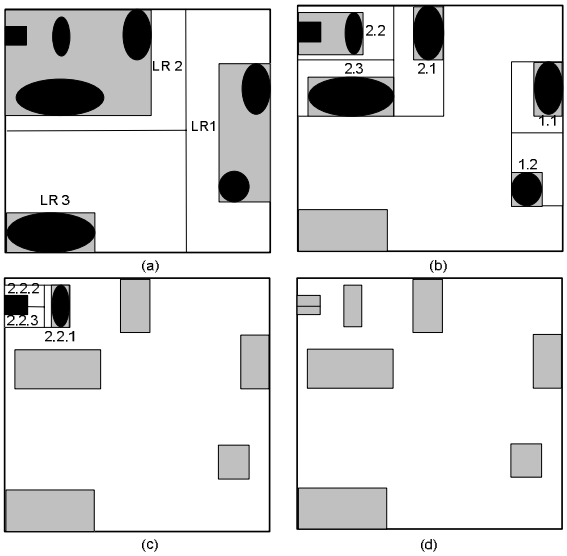

Figure 1: Illustration of the recursive data space reduction.

Figure 1 gives an example of this data space reduction process. The dark ovals represent areas populated by data points. This figure shows the γ partition of the initial base region and the live regions with density lower than the user-defined threshold. In Figures 1b and 1c, sparse live regions *LR1* and *LR2* are partitioned further until all their enclosed dense cells are identified. Whenever a high-density live region is detected (e.g., *LR3* in Figure 1a), it is included into the set of dense cells. The resulting dense cells of this example are shown in Figure 1d.

Due to the unique properties of γ partitioning [18], this data space reduction technique can handle large data sets of high dimensionality both efficiently and effectively. Different from traditional grid-based partitioning, which

in a d-dimensional space produces at least 2^d grid cells within each region that is split, γ partitioning generates at most $d+1$ cells within each such region. Thus, the number of resulting cells increases linearly with dimensionality. Because of this, we propose to use the above described data space reduction technique to compress large high-dimensional data sets.

The summary information we store in each dense cell includes: the number of points *N*, linear sum *LS*, squared sum *SS*, density of the dense cell *DEN*, low end point *LP* and high end point *HP* of the dense cell. Each dense cell can thus be denoted as (*N, LS, SS, DEN, LP, HP*). On this summarized information, we can perform either a partitioning clustering algorithm, e.g. Kmeans, or a hierarchical clustering algorithm, e.g. CURE, or a model-based clustering method, e.g. EM.

The previously proposed clustering technique, GARDEN$_{HD}$, uses the same technique to perform data space reduction. However, in the second phase, it merges the adjacent dense cells based on their distance in order to find the final clusters [18]. The data summarization based clustering framework we proposed in this paper provides a simple and straightforward way to scale up existing clustering methods while preserving the intrinsic property of data distributions. It is worthwhile to notice that the purpose of this paper is not to demonstrate the superiority of GARDEN-Kmeans over GARDEN$_{HD}$. Instead, the paper provides a framework which can improve the scalability of existing clustering algorithms in terms of dimensionality and the size of the data set.

4. GARDEN-Kmeans

The GARDEN space reduction technique can be used to scale up many kinds of existing clustering methods with slight modification of the original algorithms. Here, we take Kmeans as an example, and describe the way Kmeans can be applied on the summary data, i.e. the dense cells produced by the GARDEN space reduction technique.

Unlike BIRCH, which constrains the maximum neighbors in the leaf node, GARDEN data space reduction produces different sizes of dense cells based on the density of the decomposed areas. Therefore, the cardinalities and densities of different dense cells can vary greatly from one dense cell to another. When we perform Kmeans on the condensed information (the information stored in dense cells), we treat each dense cell as a pseudo-point. In order to reflect the varying size and density of each dense cell, we weight each dense cell by its density. We use the centroid as the representative point of the dense cells, and assign density as the weight for each representative point. The weight reflects the impact of density on shifting the centroid of each intermediate cluster during an iteration of Kmeans clustering.

Let us denote the point vector as P and weight vector as W. Then the centroid of the weighted dense-cell representatives can be calculated as: $\frac{\sum_{i=1}^{n} w_i p_i}{\sum_{i=1}^{n} w_i}$ where n is number of dense-cell representatives.

GARDEN-Kmeans first performs GARDEN space reduction, which results in a list of dense cells. The algorithm then uses the centroids to represent dense cells and assigns densities to the representatives as weights. The second step of GARDEN-Kmeans randomly selects k representatives as initial k centers. It iteratively assigns the representatives to the closest group based on the distance between the representatives and the center of the group. Afterwards, it re-calculates the center for each group and starts another round of membership assignment until the percentage of the points that change the membership is below the threshold. We set 0.5% as the convergent threshold. We also set the maximum number of iterations to 10. In other words, when there are less than 0.5% data points that change membership or the number of iterations reaches 10, the program will terminate. In order to perform fair comparison, we set the same thresholds in the implementation of the original Kmeans algorithm.

Since the data space reduction procedure adopted here does not restrict the depth of recursion, the depth of the recursion is O(log *N*). For a uniform distribution, γ partitioning of a base region produces a balanced distribution of points among the resulting γ regions, ensuring O(log *N*) levels of recursion regardless of the value of density threshold. For skewed data, the fact that γ partitioning is performed on live regions, and that it is multi-way partitioning, ensures that the volumes of live regions at some level O(log *N*) are so small that they are dense even when the selected density threshold is extremely high.

To derive live regions without restricting the level of recursion, at most *N* points will be examined and no point will be examined more than once. O(1) comparisons will be performed to allocate each point to a γ sub-region [18]. Thus, with the guaranteed O(log *N*) depth of the recursion, the running time of the data space reduction is O(*N* log *N*).

Kmeans performs in O($n' \cdot k \cdot l$) time, where n' is the number of dense cells, k is the number of clusters, and l is the number of iterations. Thus, the total running time of the two phases is O(*N* log *N*). By restricting the depth of recursion, GARDEN-Kmeans can run in guaranteed O(*N*) time.

5. Experimental Results

We conducted a series of experiments to test the efficiency and effectiveness of the proposed clustering algorithm. The experiments were performed on the following data sets:

DS1: A group of 10 synthetic data sets with 100,000 points and varying dimensionality from 10 to 100. We denote this data set as DS1-1. The second group of 5 synthetic 10-dimensional data has varying number of points from 100,000 to 500,000. We denote it as DS1-2. Data in these groups have "center-corners" distribution, in which a generated hyper-rectangle is placed in the center and others in 10 different corners of the space (origin, far corner, and 8 randomly selected corners). All generated hyper-rectangles have the same density. Moreover, each of the hyper-rectangles has uniform internal distribution and represents a different class of data. Thus, each point is assigned to one of 11 classes.

DS2: This synthetic data set with 500,000 points, called "animals", is produced by the "animals.c" program obtained from the UCI ML Repository (www.ics.uci.edu/~mlearn/MLRepository.html). It has 72 dimensions (the 73rd dimension records the class information). There are 4 classes in this set, each of which represents one type of animal. We denote this set as DS2.

DS3: The real data set, called "covtype", is also obtained from the UCI Machine Learning Repository. It has 581,012 points with 54 dimensions (the 55th dimension records the class information of objects). There are 7 classes in this data set each of which represents a type of tree. This data set will be referred to as DS3.

All experiments were performed on a PC with 3.6 GHz CPU, 3.25 GB RAM and 1 MB CPU Cache. We compare GARDEN-Kmeans with Kmeans, $GARDEN_{HD}$, and $CLUTO_{RB}$ [20], which applies recursive bisection Kmeans. In the implementation of the original Kmeans, we also select initial k centers randomly. For all metrics, we compare the average results of 10 trials.

We adopt purity and entropy to measure the effectiveness of clustering results. Both metrics measure the degree to which each cluster contains objects of a single class. Entropy metric is derived from information theory. It is defined as

$$Entropy(C) = -\sum_{i=1}^{k} p_i \log_2 p_i ,$$

where C is a cluster, k is the number of classes in cluster C, p_i is the fraction of points in class i of all points in the cluster C. The entropy of the whole data set is the total of weighted entropies of all clusters. Intuitively, entropy measures the impurity of the cluster. Therefore, the lower the entropy value, the better the cluster quality. Purity measures the proportion of the majority class in each cluster. Using the above notion p_i, purity is the maximum among p_i where $i=1,2...k$. Total purity is the total of the weighted purities of all clusters. Intuitively, purity measures the number of correctly classified points. Therefore, higher purity value indicates better clustering quality. For conciseness, we only show the purity results. The entropy results corroborate all our finding discussed below. The average running times shown here are in milliseconds.

Figure 2 shows the purity values produced by GARDEN-Kmeans, Kmeans, $GARDEN_{HD}$, and $CLUTO_{RB}$ on DS1-1. In Figure 2, we observe that, with the increasing dimensionality, GARDEN-Kmeans produces similar quality clusters as Kmeans. GARDEN-Kmeans performs slightly better than $CLUTO_{RB}$, especially in high-dimensional spaces. Among these, $GARDEN_{HD}$ produces clusters of the best quality.

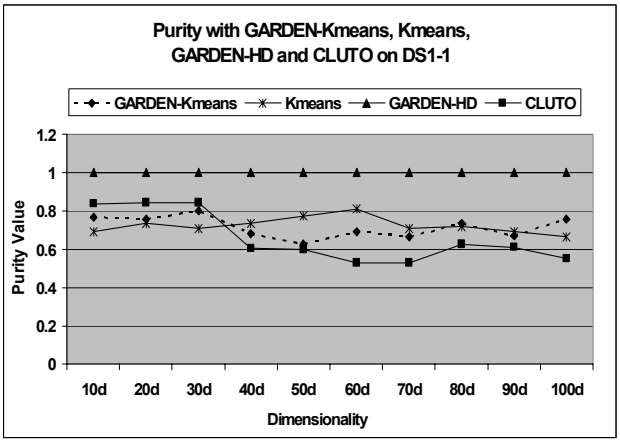

Figure 2: Purity results on DS1-1 as dimensionality grows.

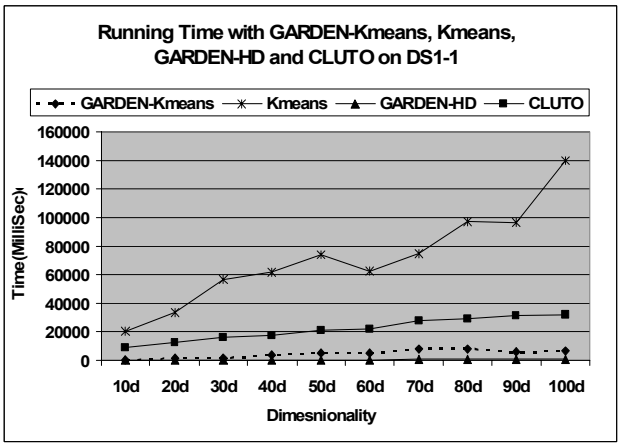

Figure 3: Running times on DS1-1 as dimensionality grows.

Figure 3 shows the running times of GARDEN-Kmeans, Kmeans, $GARDEN_{HD}$, and $CLUTO_{RB}$ on DS1-1. This figure clearly shows that GARDEN-Kmeans performs several magnitudes faster than Kmeans and $CLUTO_{RB}$ as dimensionality increases. It also shows that GARDEN-Kmeans performs slightly slower than $GARDEN_{HD}$.

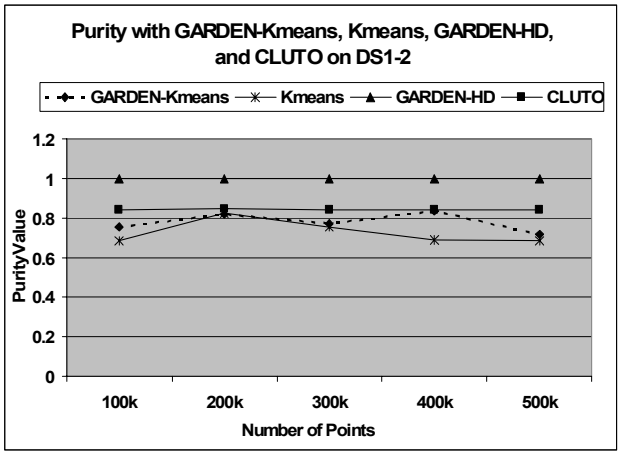

Figure 4: Purity results on DS1-2 as the volume of data grows.

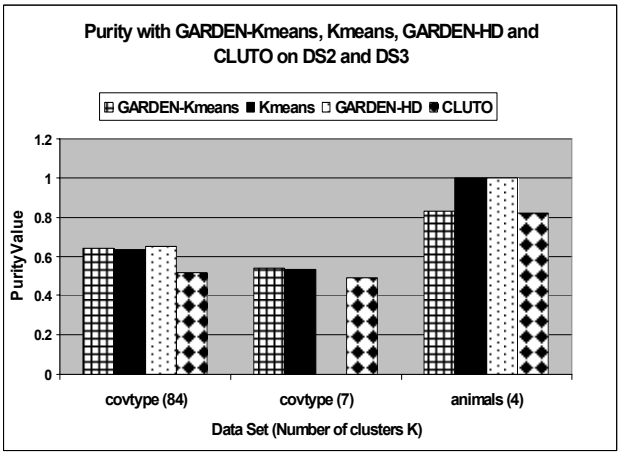

Figure 6: Purity results on DS2 and DS3.

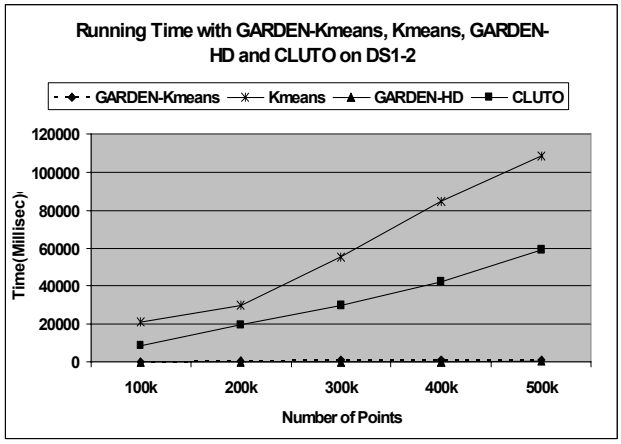

Figure 5: Running times on DS1-2 as volume of data grows.

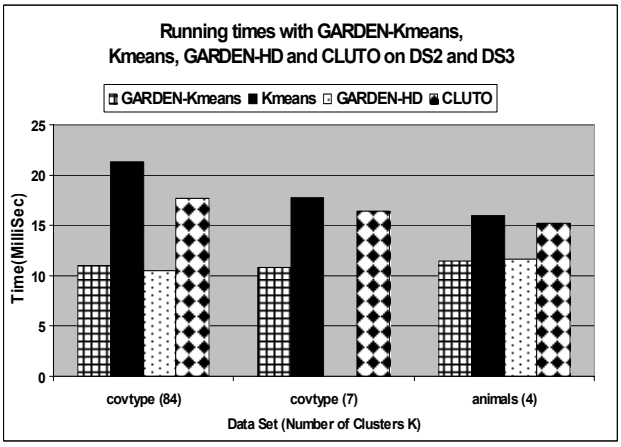

Figure 7: Running times on DS2 and DS3 (log scale).

Figure 4 compares the four methods on DS1-2 in terms of their purity values. This figure shows that GARDEN-Kmeans produces clusters of similar quality as Kmeans and $CLUTO_{RB}$ as the number of data points increases. $GARDEN_{HD}$ still outperforms the other methods.

Figure 5 compares the speed of GARDEN-Kmeans, Kmeans, $GARDEN_{HD}$, and $CLUTO_{RB}$ on DS1-2. This figure also shows superior speed of GARDEN-Kmeans compared to Kmeans and $CLUTO_{RB}$. The execution times of GARDEN-Kmeans and $GARDEN_{HD}$ are very close.

Figure 6 shows the purity values of the four methods on DS2 and the real data set DS3. The horizontal axis in this figure represents different data sets and the number of clusters, K, set by the user. The vertical axis shows the purity values. The first value on the horizontal axis, "covtype(84)", represents the real life data set "covtype", and the user input for number of clusters K = 84. Here, we set K = 84 because, for the selected input density, $GARDEN_{HD}$ generates 84 clusters [17, 18]. Since $GARDEN_{HD}$ does not take the number of clusters K as its parameter, the second value, covtype(7), is not applicable to $GARDEN_{HD}$. Thus, the corresponding bar is left blank.

In terms of purity, GARDEN-Kmeans performs as well as $GARDEN_{HD}$ on the real data set DS3, and slightly worse on the synthetic set DS2. However, unlike $GARDEN_{HD}$, GARDEN-Kmeans provides the user with the flexibility to specify the desired number of clusters. This feature can be an advantage in some applications. However, this does require the user's pre-knowledge of the data set. Kmeans also shows good clustering quality, similar to that of $GARDEN_{HD}$.

Figure 7 compares the running times of GARDEN-Kmeans, Kmeans, $GARDEN_{HD}$, and $CLUTO_{RB}$ on DS2 and DS3. The horizontal axis in this figure is the same as in Figure 6. The vertical axis represents the running time. In order to scale down the difference so that it can be shown in this figure, we take log of the running times. The results shown in Figure 7 are the log values of the recorded running times. Obviously, the performance improvements of GARDEN-Kmeans over the original Kmeans and $CLUTO_{RB}$ can be significant. Applying the GARDEN data space reduction to other clustering methods, not just Kmeans, can yield similar improvements.

6. Conclusions

In this paper, we proposed a framework for clustering large volumes of high-dimensional data. This framework is built on the data space reduction of $GARDEN_{HD}$ [18]. With a slight modification, the existing clustering techniques can perform clustering on the compressed summary information produced by the GARDEN space reduction. Taking Kmeans as an example, we developed GARDEN-Kmeans based on the proposed framework. The experimental results show that GARDEN-Kmeans executes several orders of magnitude faster than either the original Kmeans or a recursive bisection Kmeans algorithm, $CLUTO_{RB}$, while preserving the clustering accuracy. The results on real data show that GARDEN-Kmeans, which performs clustering on a compressed summary of data, achieves high clustering accuracy, comparable even to that of $GARDEN_{HD}$, though at a slightly lower speed. On synthetic data sets, $GARDEN_{HD}$ maintains a slight advantage over GARDEN-Kmeans both in terms of the clustering speed and clustering quality. These results further prove the superior speed and effectiveness of the stand-alone $GARDEN_{HD}$ clustering technique proposed in [18]. However, the general framework provided in this paper enables us to better scale existing clustering algorithms.

Several papers [12,14] have been published on the use of data summarization in clustering in order to handle dynamic or streaming data. In the future, we will adapt the proposed clustering framework to the problem of clustering dynamic data sets and streaming data. The superior speed of GARDEN-Kmeans, as demonstrated in this study, makes this research direction very promising.

7. Acknowledgement

This material is based upon work supported by the National Science Foundation under grants no. IIS-0312266 and IIS-0635365.

8. References

[1] P.S. Bradley, U. Fayyad, C. Reina: Scaling Clustering Algorithms to Large Databases. In *Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining*, pages, 9-15, 1998.

[2] M. Breunig, H.-P. Kriegel, P. Kröger, J. Sander. Data bubbles: quality preserving performance boosting for hierarchical clustering. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 79-90, 2001.

[3] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[4] F. Farnstrom, J. Lewis and C. Elkan. Scalability for clustering algorithms revisited. In *SIGKDD Explorations*, Volume 2, Issue 1, pages 51-57, June 2000.

[5] A. Hinneburg and D. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proc. 4th Int. Conf. on Knowl. Disc. and Data Mining*, pages 58–65, 1998.

[6] A. Hinneburg and D. Keim. Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. In *Proc. Conf. on Very Large Data Bases*, 1999.

[7] H. Jin, M-L, Wong, K-S Leung. Scalable Model-based clustering by working on data summaries. In *Proc. 13th IEEE Int. Con. On Data Mining, ICDM'03*, pages 91-98, 2003.

[8] G. Karypis. Cluto: A clustering toolkit. *Technical Report 02-017*, Computer Science, University of Minnesota, 2003.

[9] L. Kaufman and P.J. Rousseeuw. Finding Groups in Data: an Introduction to Cluster Analysis, *John Wiley & Sons*, 1990.

[10] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. 5th Symp. Math. Statist. and Probability,* Vol. 1, pages 281–297, 1967.

[11] B. L. Milenova and M. M. Campos. O-cluster: scalable clustering of large high dimensional data sets. In *Pro. IEEE Int. Conf. on Data Mining,* pages:290 – 297,2002.

[12] S. Nassar, J, Sander, C, Cheng. Incremental and effective data summarization for dynamic hierarchical clustering. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 2004.

[13] R. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proc. 20th Int. Conf. on Very Large Data Bases*, pages 144–155, 1994.

[14] C. Ordonez. Clustering binary data streams with K-means. In *Proc. 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 12-18, 2003

[15] C. Ordonez and E. Omiecinski. FREM: Fast and robust EM clustering for large data sets. In *Proc. 11th Int. Conf. on Information and Knowledge Management, CIKM 02,* pages 590 – 599, 2002.

[16] R. Orlandic. Effective management of hierarchical storage using two levels of data clustering. In *Proc. 20th IEEE/11th NASA Goddard Conf. on Mass Storage Systems and Technology*, pages 270–279, 2003.

[17] R. Orlandic, Y. Lai, and W. Yee. Clustering high-dimensional data using an efficient and effective data space reduction. *Technical Report*, Computer Science, Illinois institute of Technology, 2005. www.cs.iit.edu/~egalite

[18] R. Orlandic, Y. Lai and W.G. Yee. Clustering high-dimensional data using an efficient and effective data space reduction, In *Proc. ACM Conf. on Information and Knowledge Management CIKM'05*, pages 201-208, 2005.

[19] W. Wang, J. Yang, and R. Muntz. Sting: A statistical information grid approach to spatial data mining. In *Proc. 23rd Int. Conf. on Very Large Data Bases*, pages 186–195, 1997.

[20] Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proc. ACM Conf. on Inf. and Knowl. Mgt. CIKM'02*, pages 515–524, 2002.

[21] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 103–114, 1996.

[22] J. Zhou and J. Sander. Bata bubbles for non-vector data: speeding-up hierarchical clustering in arbitrary metric spaces. In *Proc. 29th Int. Conf. on Very Large Data Bases*, pages 452–463, 2003.