Dynamic Incremental K-means Clustering

Bryant Aaron, Dan E. Tamir
Department of Computer Science,
Texas State University,
San Marcos, Texas, USA
{ba1127,dt19}@txstate.edu

Naphtali D. Rishe, and Abraham Kandel
School of Computing and Information Sciences
Florida International University
Miami, Florida, USA
rishen@fiu.edu,abekandel@yahoo.com

***Abstract*— K-means clustering is one of the most commonly used methods for classification and data-mining. When the amount of data to be clustered is "huge," and/or when data becomes available in increments, one has to devise incremental K-means procedures. Current research on incremental clustering does not address several of the specific problems of incremental K-means including the seeding problem, sensitivity of the algorithm to the order of the data, and the number of clusters. In this paper we present static and dynamic single-pass incremental K-means procedures that overcome these limitations.**

Keywords—Clustering; K-means Clustering; Incremental Clustering; Data-mining

I. INTRODUCTION

K-means clustering is an iterative optimization algorithm, which is one of the most commonly used highly effective methods for data-mining and non-supervised classification. Traditionally, in each of the iterations, the algorithm is applied to the entire data set represented as a vector residing in the processor memory and representing a multi-dimensional set of measurements. Recently, however, the K-means algorithm, as well as numerous other clustering and data-mining algorithms such as fuzzy C-means (FCM), ISODATA, Kohonen neural networks (KNN), expectation maximization, and simulated annealing [1-6], have been exposed to a relatively new challenge referred to as "big-data." Often, the enormous amount of data available online cannot fit processors' physical memory. In fact, often the data does not even fit secondary memory. Given that input/output operations are generally the most taxing computer operations, working on the entire data in every K-means iteration requires numerous consecutive reads of massive amounts of data. This might dim the K-means algorithm as a non-practical approach for mining in big-data environments. Another scenario that might challenge the traditional approach to K-means clustering occurs when portions of the data are generated or become available dynamically and it is not practical to wait for the entire dataset to be available.

This brings the need for incremental clustering into the forefront. Incremental clustering is also referred to as a single-pass clustering whereas the traditional clustering is referred to as multi-pass clustering [7-11]. The idea is to cluster a manageable portion of the data (a data-block) and maintain results for the next manageable block until exhausting the data. Under this approach each block is processed by the algorithm a limited number of times, potentially only once. Ideally, a block, along with the preliminary number of initial centers selected should be as large as possible, occupying as much of the available internal processor memory. One might question the validity of "visiting" every data element for a limited number of iterations in a specific order as opposed to the traditional approach which considers every piece of data in each iteration. This is further discussed in section III-E.

A related approach is the multi-resolution or multistage clustering. Researchers observed that a multistage based training-procedure can accelerate the convergence and improve the quality of the training as well as the quality of the classification/decision phases of many of the clustering algorithms [5,12,13]. For example, our previous research reports show that the pyramid K-means clustering algorithm, the pyramid Fuzzy C-means algorithm, and multi-resolution KNN yield two-to-four times convergence speedup [13]. Both the multistage clustering and the incremental clustering apply an approach of sampling the data. In the multistage clustering, however, data is sampled with replacement, whereas in incremental clustering, due to the cost of replacement, the data is sampled without replacement. In both cases the validity of the sampling has to be addressed.

This paper describes a new and novel approach for incremental K-means clustering. The method stems from the pyramid K-means algorithm presented in [5,12,13]. In difference from the pyramid approach, the sampling is done without replacement. Furthermore, the sampling size is fixed. On the other hand, two measures are applied to the data in order to overcome the fact that each data block is processed only one time. First, the algorithm starts with a relatively large number of clusters and scales the number down in the last stage. This is referred to a two phase procedure. Hence, in the intermediate stages (first phase) each block might affect different cluster centers. A second and innovative version of the algorithm enables a dynamic number of clusters. Again, the algorithm starts with a relatively large number of clusters, however, each transaction on a block might change (increase or decrease) the number of clusters. Notably, the dynamic approach can be used to mitigate another inherent problem of K-means where the number of clusters has to be predetermined. In both cases the algorithms works on

This material is based in part upon work supported by the National Science Foundation under Grant Nos. CNS-0821345, CNS-1126619, HRD-0833093, IIP-0829576, CNS-1057661, IIS-1052625, CNS-0959985, OISE-1157372, IIP-1237818, IIP-1330943, IIP-1230661, IIP-1026265, IIP-1058606, IIS-1213026.

"chunks" of data referred to as blocks. The paper presents several experiments with multi-pass and static/dynamic single-pass versions of the K-means and empirically evaluates the validity of the static and dynamic incremental clustering approach.

The main contributions of the paper are: 1) a new approach for incremental clustering where the number of clusters is relatively high, followed by clustering the resultant centers is presented. This approach increases the validity of incremental clustering, 2) a second approach where initially the number of clusters is relatively high, and the number of clusters dynamically changes throughout execution provides better incremental clustering quality/validity and can be used to resolve the issue of identifying the right number of cluster centers.

A literature review performed shows numerous papers on incremental clustering. Nevertheless, the number of papers that apply incremental clustering to K-means is relatively small and to the best of our knowledge there are no reports on research that applies the operations listed in this paper to the K-means algorithm.

The rest of the paper is organized in the following way. Section II reviews related research. Section III provides the details of several single-pass and multi-pass variants of K-means clustering and lists metrics used to assess the quality of clustering. Section IV describes a set of experiments conducted to assess the performance and validity of the incremental clustering algorithms described in section III and section V includes conclusions and proposals for further research.

II. Review of Related Research

Clustering is a widely-used data classification and data-mining method applied in numerous research fields including image segmentation, vector quantization, data-mining, and data compression [7,14-20]. K-means is one of the most commonly used clustering algorithms, and the Linde, Buzo, and Gray (LBG) vector quantization (VQ) algorithm with unknown probability distribution of the sources, which is a variant of K-means, is utilized in many applications [15].

Garey has shown that the LBG VQ converges in a finite number of iterations, yet it is NP complete [16]. Thus, finding the global minimum solution or proving that a given solution is optimal is an intractable problem. Another problem with K-means is that the number of clusters (K) is fixed and has to be set in advance of executing the algorithm. ISODATA is a generalization of K-means which allows splitting, merging, and eliminating clusters dynamically [2]. This might lead to better clustering (better local optimum) and eliminate the need to set K in advance. ISODATA, however, is computationally expensive and is not guaranteed to converge [2].

Numerous applications require clustering of very large data-sets which are too big to fit the available memory and/or clustering data that become available in "increments". Both scenarios induce the need for incremental clustering [7-11]. Significant amount of work was done on the subject of incremental clustering [7-10]. Nevertheless, there are relatively few papers that deal with incremental K-means and/or incremental dynamic clustering. Several of these algorithms load a slice of the data, where the size of a slice is constraint by available memory, and cluster this slice [21]. Results of clustering current slices (e.g., centers, partition matrices, dispersion, etc.) are used in the process of clustering upcoming slices.

Charikar et al. propose a framework for incremental clustering [8]. Their framework, however, is not specific enough for incremental K-means clustering. Young et al. propose a fast and stable incremental clustering algorithm that is based on competitive learning. Hore has proposed a slice based single-pass FCM algorithm for large data-sets [21]. The proposed method lumps data that has been clustered in previous slices into a set of weighted points and uses the weighted points along with fresh slices to commence with the clustering of the entire set in one path [21]. Another approach for clustering large data-sets is to sample, rather than slice, the data [22]. Berkhin provides a thorough survey of clustering and relates to the problem of incremental single-pass clustering. He notes that one problem with incremental clustering is the order in which the data is visited [3]. This is also observed by Young [23]. Our two phase approach implementing a very large number of clusters in the first phase mitigates this problem. Several authors describe the application of K-means in problems that relate to incremental clustering. Nevertheless, they use K-means without checking all alternatives and validity. Cheng proposes a divide and merge algorithm yet this algorithm does not utilize K-means [24]. Lughofer proposes a dynamic Evolving Cluster Models using On-line Split-and-Merge Operations [22] this is somewhat similar to dynamic ISODATA. Our method for changing the number of clusters is simpler.

III. K-means Clustering Variants

In this section we present several variants of K-means clustering.

A. The Classical K-means Clustering Algorithm

We refer to the K-means variant introduced by MacQueen as the classical K-means algorithm [1]. The algorithm is an iterative criterion optimization attempting to optimize the sum of the squared distances from all the data points to their assigned cluster center [1-3]. In this case the clustering problem can be stated in the following way: Given a set of N inputs or measurements $\vec{x}_1, \vec{x}_2, ..., \vec{x}_N$ where $\vec{x}_i \in R^n$ (that is, each element of the set of measurements is a vector in an n-dimensional space), find K ($K << N$) clusters $c_1, c_2, ..., c_k$ with cluster-centers $\vec{\omega}_1, \vec{\omega}_2, ..., \vec{\omega}_k$ respectively, such that 'D,' the sum of squared distances of data points to their respective centers given by the following equation (equation 1), is equivalent to the mean-square-error (MSE). Often, D is referred to as the distortion. This is especially relevant when K-means is used for vector quantization [15,16]:

$$D = \frac{1}{N} \sum_{j=1}^{K} \sum_{\vec{x} \in c_j} (\vec{x} - \vec{\omega}_j)^2 \quad (1)$$

The algorithm consists of the following steps: First, initial centers are selected in a process that is referred to as seeding and all objects are classified into the appropriate clusters based on the nearest neighbor rule [1-3]. Next, cluster-centers are updated to be at the centroid of their respective clusters. The objects are then reclassified, and new centers are calculated. The algorithm continues until convergence; where the convergence criterion dictates that the centers' location at iteration i are identical to the centers' location at iteration $i+1$.

B. The LBG Termination Criterion

The LBG vector quantization algorithm generalizes the K-means algorithm [15]. The main difference between the LBG algorithm and the classical algorithm is the termination condition. The LBG algorithm stops when an approximation for the derivative of the MSE given by $\frac{D^{(m)} - D^{(m+1)}}{D^{(m)}}$ is smaller than a threshold; ($\frac{D^{(m)} - D^{(m+1)}}{D^{(m)}} < \varepsilon$) . The advantage of this criterion is that generally convergence occurs faster with quality results (minimum MSE/minimum distortion) that are about the same as the results of the classical K-means.

C. The Sequential K-means Algorithm

K-means, as well as numerous other iterative optimization algorithms, can be executed in one of two basic modes; batch mode and on-line mode. The on-line mode is also referred to as the sequential mode. In an epoch of a sequential mode the parameters of prototype patterns are recalculated subsequent to the introduction of a new element of the training set. In batch mode each epoch uses the entire training set to recalculate parameters. In K-means, the sequential mode assigns a point to a cluster and immediately updates the cluster centers. Due to the heavy overhead of recalculating centers, the resource requirement of sequential K-means might be a prohibitive factor. Nevertheless, in certain scenarios the data appears in a mode of one element at a time. In these cases, it might be desirable to update results per data occurrence rather than waiting for the accumulation of the entire data set.

D. The Block Sequential K-means Algorithm

The block sequential mode is a compromise between the stringent computational requirements of the sequential K-means and the need to operate on data online. In this case, the clustering occurs on accumulated blocks of data. Each block is going through l epochs of K-means where the final centers of block i are used as the initial centers for block $i+1$. In many cases $l = 1$. In this sense, the algorithm resembles other multistage clustering such as the pyramid K-means [12-14]. The block sequential algorithm might be utilized in an iterative fashion where each of the iterations performs l epochs of K-means on a single block of data elements at a time.

Note that all the clustering algorithms described so far assume that (at some point) the entire data set is available. Moreover, generally, due to the iterative fashion of execution, these algorithms access the same elements more than once (in different iterations). When the data is very large and cannot fit the memory of the processor, a different approach, referred to as single-pass, has to be adapted. Under the single-pass (incremental) approach, each block/data-element is accessed only one time and then removed from internal memory to provide space for new elements. This is described next.

E. The Incremental K-means Algorithm

The incremental K-means algorithm presented in this paper is similar to the block sequential algorithm with the exception that each block is accessed only one time. Each block is going through a set of l epochs of K-means where the final centers of block i are used as the initial centers for block $i+1$.

The fact that each block is "touched" just one time might raise a question about the validity of the results. The results might be valid if the data elements of blocks share similar features. For example, the data elements are drawn from the same probability distribution function or the same fuzzy membership function. Alternatively, validity might be attained if the features of data elements vary "slowly" between blocks. We use two methods to improve the validity of results. First, we use a two phase incremental algorithm. In the first phase a very large set of clusters is used. Practically, we are trying to fit as many elements in a block and as many clusters per block as possible in the memory. In the next phase, after processing all the blocks, a process of clustering the centers obtained from the last block is applied. The second measure for increasing validity is using a relatively large number of clusters and at the same time allowing the number of clusters to change dynamically. This is described in the next section.

F. The Dynamic Incremental K-means Algorithm

The dynamic incremental K-means algorithm presented in this paper is similar to the incremental K-means algorithm. The difference is that the number of clusters is allowed to change.

Several operations can change the number of clusters. First, following the ISODATA algorithm principles, clusters with too few elements might be eliminated, clusters that are too close to each other might be merged, and clusters with large dispersion might be split [22-24]. The criteria for merge and split might be related to the within and between dispersion of the clusters [1-3,22]. Other methods for changing the number of clusters might include incrementing/decrementing the number of clusters (without split/merge) based on a criterion such as a threshold on the distortion. We have implemented the threshold approach.

Each block is going through a set of l epochs of K-means where the final centers of block i are used as the initial centers for block $i+1$. Following the application of K-means on a block, a decision concerning the effective number of clusters

is made and the number might be incremented or decremented based on predetermined quality criteria threshold. We place an upper bound and a lower bound on the number of clusters where the lower bound ensures that we still have enough clusters to maintain validity and enable the two phase approach described above. Again, we are trying to fit as many elements in a block and a large number of clusters per block in the memory and apply a two phase approach where the centers from the last iteration are clustered and provide the final set of clusters.

IV. Experiments and Results

A. Experimental Setup

In order to compare and contrast the performance and validity of the K-means variants presented we have implemented these algorithms numerous times on different data-sets, using different parameters. Two sets of data are used for the experiments performed; the first set consists of the Red, Green, and Blue (RGB) components of color images used for color quantization and the second set includes synthetic data with known centers and known distribution. Three types of output data/results are collected: 1) records of convergences (distortion per iteration), 2) execution time and 3) clustering quality (inverse of distortion at the final iteration).

The experiments are divided into two classes; multi-pass and single-pass. In the multi-pass experiments we compared the performance of classical K-means, LBG based K-means, sequential K-means, and block sequential K-means. Given the constraints of these algorithms they were applied to a manageable data set (i.e., a data-set with a medium number of elements). In the single-pass experiments we tested the incremental and the dynamic incremental approaches with the same data used for the multi-pass algorithms and with a "huge" set of synthetic data that is not suitable for multi-pass processing. Nevertheless, to verify the results with the large data-set we ran the LBG algorithm on that data using a "powerful" multicore computer. The computer worked on the data for several hours. For the dynamic incremental algorithm, we used an approach where the number of clusters is incremented/decremented by 1based on a threshold on distortion.

All the experiments are performed with the script version of MATLAB using the built-in MATLAB K-means function where the function is executed one epoch at a time.

1) Color Quantization

The problem of color quantization can be stated in the following way: given an image with N different colors, choose $K \ll N$ colors such that the resulting K-color image is the least distorted version of the original image [19]. Color quantization can be implemented by applying the K-means clustering procedure to the image-pixels where each pixel represents a vector in some color representation system. For example, the clustering can be performed on the three-dimensional vectors formed by the red, green, blue (RGB) color components of each pixel in the image [19]. After clustering, each three-dimensional vector (pixel) is represented by the cluster-number to which the vector belongs, and the cluster centers are stored in a color-map. The K-value image along with the color-map is a compressed representation of the N-colors, original image. The compressed image can be used to reconstruct the original three-dimensional data-set by replacing each cluster-number by the centroid associated with the cluster. In the case of 8 bit per color component and $K = 16$, the original 24-bit per pixel image is represented by a 4-bits per pixel image along with a small color map. Hence, about 6 times compression is achieved. In this set of experiments, a block processed by the single-pass algorithm consists of an image row.

The sequential algorithm is applied to every pixel of a scaled down version of the images while the rest of the multi-pass algorithms operate on the entire set of the pixels of the original images. The experimental results are scaled to represent the distortion for the entire image. The static incremental algorithm starts with 192 clusters per block. Following the processing of the last block, the 192 centers are clustered into 16 centers and the distortion for the entire image with these centers is measured. The dynamic incremental algorithm starts with 192 clusters per block and allows fluctuations of ± 64 in this number. Following the processing of the last block, the centers are clustered into 16 centers and the distortion for the entire image with these centers is measured.

2) Synthetic Data

A set of 20 random cluster centers with a total of 20,000,000 6-dimensional vectors is generated. The vectors within a cluster are distributed according to a 2-D normal distribution with standard deviation of 0.05 around the center. For the single-pass experiments the data is divided into 500 blocks of 40,000 elements per block. Other parameters are identical to the ones used for the color map quantization experiments.

B. Experimental Results

1) Experiments with Color Quantization

Figure 1 shows the distortion per iteration of the multi-pass algorithms executed on the image Lena which is a 512×512 RGB image. Figure 1a shows the results of the classical K-means. Following a long convergence curve with 114 iterations it settles down at a distortion rate of 216. The LBG variant, depicted in Figure 1b, converges much faster to about the same value (209) after 32 iterations. The sequential algorithm (Figure 1c) operates on a scaled version of the image and converges relatively fast, using the LBG criterion to a distortion value that is lower than the ones attained for the classical and LBG variants (204). Nevertheless, this variant would not be applicable for a large amount of data.

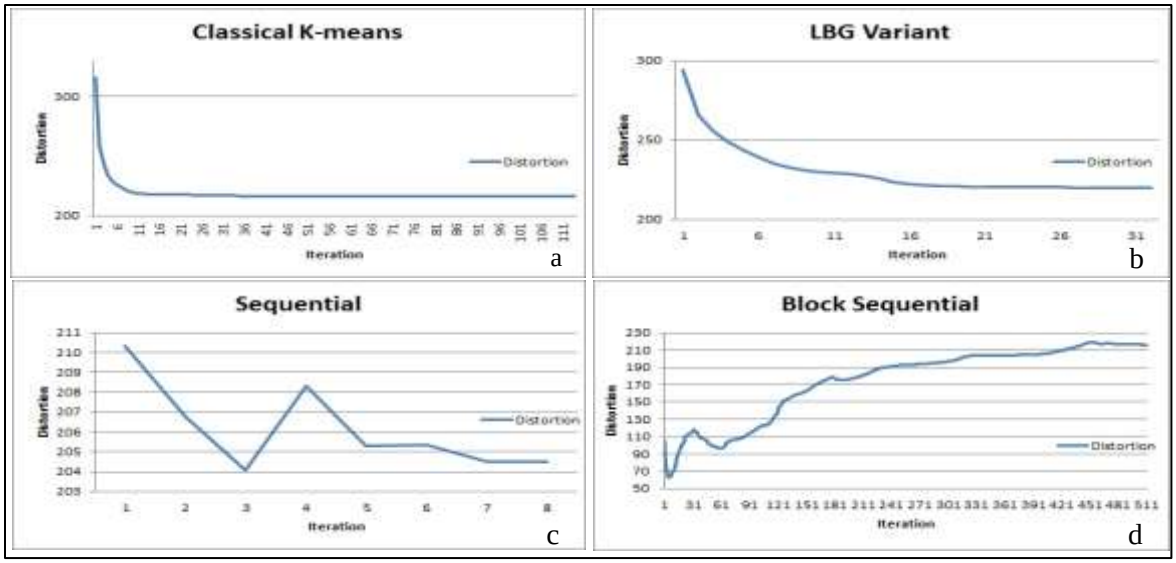

Figure 1 Distortion per iteration for the multi-pass algorithms with the Image Lena

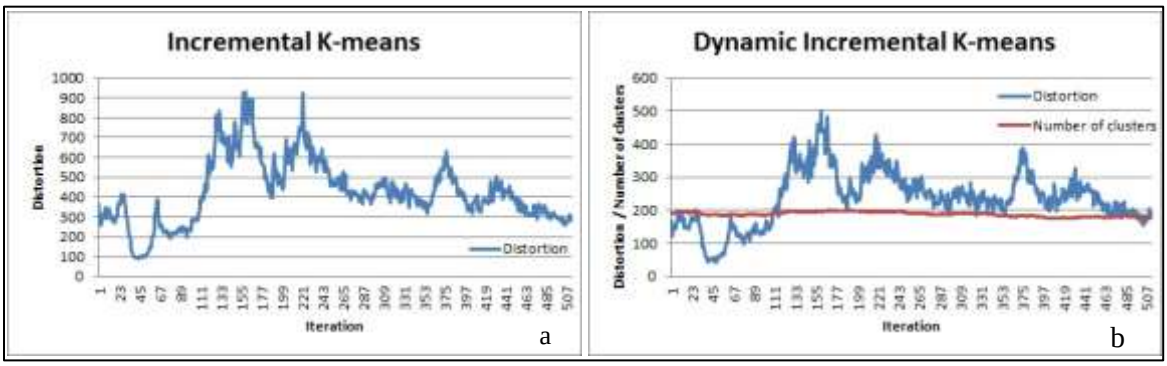

Figure 2 Incremental and Dynamic Incremental Clustering of the Image Lena

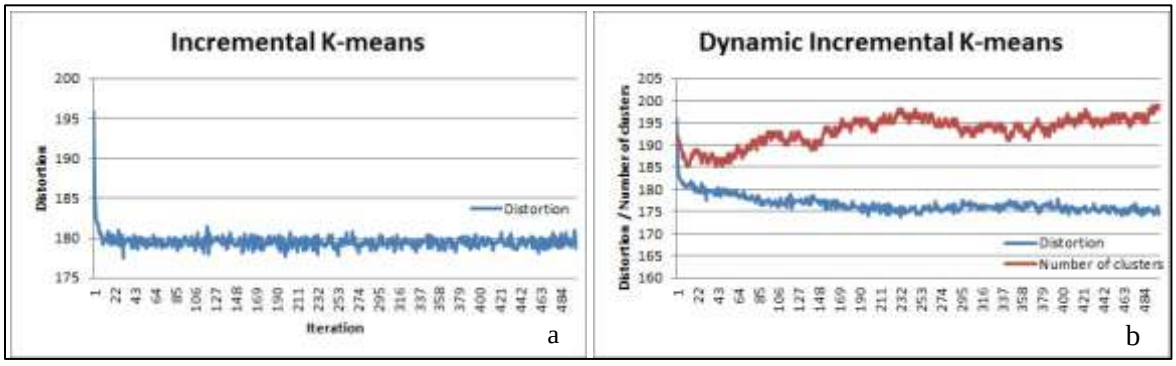

Figure 3 Incremental and Dynamic Incremental Clustering of Synthetic Data

Finally, the block sequential algorithm presents an interesting behavior. Figure 1d depicts the distortion in iteration 1 per block for the 512 blocks of the image. The results converge to the value of 216 which is very close to the results of running the other algorithms for several iterations. Furthermore, the next iteration produces the same results. Hence, practically, the algorithm converges in a single iteration. This is consistent with results of multistage and pyramid K-means clustering and can be used to implement fast versions of multi-pass K-means. Visually, all the versions of clustering executed in this research produced about the same reconstructed image.

Figure 2 shows the results of running the single-pass incremental versions on the image Lena with single row per block. Figure 2a shows the results of dedicating 1 epoch of the incremental K-means per block. The distortion is about 275. Figure 2b shows the number of clusters (in brown) and distortion per block for the dynamic incremental K-means algorithm with 5 epochs per block. The number of clusters is fluctuating around 200. In fact, a run with 1, 3, 5, and 100

epochs per block produces similar results. Hence, in this case, it is sufficient to allocate 1 epoch per block. Compared to the static incremental algorithm the distortion per iteration is a bit more stable and converges to 172 which is somewhat better than the value for the static cases. In terms of time, except for the sequential K-means, all the algorithms are comparable with a relatively short CPU time of about 300 seconds. This is a reasonable CPU time for a scripting MATLAB implementation.

2) Experiments with Synthetic Data

Figure 3 shows the results of incremental clustering and dynamic incremental clustering with a "huge" amount of synthetic data points (20,000,000 points) in a 6 dimensional space). In addition, the figure shows the results of LBG clustering on the entire data.

Figure 3a shows the results of running the static incremental K-means on the synthetic data. Due to the fact that the data is drawn from a fixed distribution, the results of distortion per block are quite stable and the execution ends up at a value of 180. In Figure 3b, the execution of the dynamic version of incremental K-means is depicted (in brown). The number of centers starts at 192 and ends up at around 200 while the distortion per block is better than results obtained for the static case and stabilizes at 174. To validate the results we ran the LBG algorithm on the entire set of value for the static cases. In terms of time, except for the sequential K-means, all the algorithms are comparable with a relatively short CPU time of about 300 seconds. This is a reasonable CPU time for a scripting MATLAB implementation.

C. Result Evaluation

The results of the experiments reported and additional experiments performed show the utility of using a two phase single-pass incremental K-means algorithm where the first phase uses a large number of centers and the second phase clusters the centers obtained in the first phase into a desired size of clusters. Moreover, the dynamic clustering approach allows the number of centers in the first phase to vary and outperforms the static incremental approach.

V. CONCLUSIONS AND FUTURE RESEARCH

This paper has reviewed static and dynamic single-pass and multi-pass variant of the K-Means. A novel two phase static single pass algorithms as well as a dynamic two phase single-pass algorithm have been presented and are showing high utility. Future research will concentrate on additional methods for dynamic change in the number of clusters in both steps of dynamic incremental K-means. In addition, we plan to initiate research on equivalent approaches in Fuzzy c-means and in the KNN. Finally, we plan to investigate parallel incremental algorithms.

REFERENCES

[1] MacQueen, J., *Some Methods for Classification and Analysis of Multivariate Data*. 5th Berkeley Symposium on Probability and Statistics, 1967.

[2] Tou, J. and Gonzalez, R.C., Pattern Recognition Principles. 1974, London: Addison-Wesley.

[3] Pavel Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002

[4] El-Gamal, A.A., Using Stimulated Annealing to Design Good Codes. IEEE Transactions on Information Theory, 1987. 33(1): p. 116-123.

[5] Kohonen, T., et al., Artificial Networks. 1991, Amsterdam: Elsevier.

[6] D. E. Tamir and A. Kandel "The Pyramid Fuzzy C-means Algorithm," *International Journal of Computational Intelligence in Control*, (2)2 2010.

[7] Bradley, P.S., Usama, F., and Corey, R.C. Scaling Clustering Algorithms to Large Database. in Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining. 1998.

[8] Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. 1997. Incremental clustering and dynamic information retrieval. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing* (STOC '97). ACM, New York, NY, USA, 626-635.

[9] Farnstrom, F., Lewis, J., and Elkan, C. Scalability for Clustering Algorithms Revisited. in ACM SIGKDD Explorations. 2000.

[10] Gupta, C. and Grossman, R. GenIc:A Single Generalized Incremental Algorithm for Clustering. in The Fourth (SIAM) International Conference on Data Engineering. 2004.

[11] Lin, J., et al., Iterative Incremental Clustering of Time Series. Advances in Database Technology - EDBT 2004, 2004: p. 106-122.

[12] Tamir, D.E., Park, C., and Yoo, B., The Validity of Pyramid K-means, in SPIE conference on Optics and Photonics / Optical Engineering and applications. 2007: San Diego.

[13] Tamir, D.E., Cluster Validity of Multi Resolution Competitive Neural Networks. International Conference on Artificial Intelligence, 2007.

[14] Coleman, G. and Andrews, H., *Image Segmentation by Clustering.* Proceedings of the IEEE, 1979. **67**(5): p. 773-785.

[15] Linde, Y., Buzo, A., and Gray, R., An Algorithm for Vector Quantizer Design. IEEE Transactions on Communications, 1980. 28(1): p. 84-95.

[16] Garey, R.M., Johnson, S., and Witsenhausen, H.S., The Complexity of the Generalized Lloyd-Max Problem. IEEE Transactions on Information Theory, 1972. 28: p. 255-256.

[17] Huang, J.F., Image Segmentation Using Discrete Cosine Transform and K-Means Clustering Algorithm, in Computer Science. 1991, Florida Institute of Technology: Melbourne, Florida

[18] Zhu, Y., Image Segmentation for Color Separation in Computer Science. 1991, Florida Institute of Technology: Melbourne, Florida.

[19] Heckbert, P., Color Image Quantization for Frame Buffer Display. ACM Transactions on Computer Graphics, 1982. 16(3): p. 297-307.

[20] Xu, X., Jager, J., and Kriegel, H.P., A Fast Parallel Clustering Algorithm for Large Spatial Databases. . Data Mining and Knowledge Discovery, 1999. 3(3): p. 263-290.

[21] Hore, P., Hall, L.W., and Goldgof, D.B., Speedup of Fuzzy Clustering Through Stream Processing on Graphics Processing Units, in IEEE International Conference on Fuzzy Systems. 2007 London.

[22] Edwin Lughofer. 2011. Dynamic Evolving Cluster Models Using On-line Split-and-Merge Operations. In *Proceedings of the 2011 10th International Conference on Machine Learning and Applications and Workshops - Volume 02* (ICMLA '11), Vol. 2. IEEE Computer Society, Washington, DC, USA, 20-26.

[23] Young, S.; Arel, I.; Karnowski, Thomas P.; Rose, D., "A Fast and Stable Incremental Clustering Algorithm," *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on* , vol., no., pp.204,209, 12-14 April 2010

[24] David Cheng, Ravi Kannan, Santosh Vempala, and Grant Wang. 2006. A divide-and-merge methodology for clustering. *ACM Trans. Database Syst.* 31, 4 (December 2006), 1499-1525.