K-Harmonic Means - A Data Clustering Algorithm

Bin Zhang, Meichun Hsu, Umeshwar Dayal
Software Technology Laboratory
HP Laboratories Palo Alto
HPL-1999-124
October, 1999

Clustering, K-Means, K-Harmonic Means, data mining

Data clustering is one of the common techniques used in data mining. A popular performance function for measuring goodness of data clustering is the total within-cluster variance, or the total mean-square quantization error (MSE). The *K*-Means (KM) algorithm is a popular algorithm which attempts to find a K-clustering which minimizes MSE. The K-Means algorithm is a center-based clustering algorithm. The dependency of the K-Means performance on the initialization of the centers is a major problem; a similar issue exists for an alternative algorithm, Expectation Maximization (EM), although to a lesser extent. In this paper, we propose a new clustering method called the K-Harmonic Means algorithm (KHM). KHM is a center-based clustering algorithm which uses the Harmonic Averages of the distances from each data point to the centers as components to its performance function. It is demonstrated that K-Harmonic Means is essentially insensitive to the initialization of the centers. In certain cases, *K*-Harmonic Means significantly improves the quality of clustering results comparing with both *K*-Means and *EM,* which are the two most popular clustering algorithms used in data exploration and data compression. A unified view of the three performance functions, *K*-Means', *K*-Harmonic Means' and *EM*'s, are given for comparison. Experimental results of *KHM* comparing with *KM* on high dimensional data and visualization of the animation of the convergence of all three algorithms using 2-dimensional data are given.

K-Harmonic Means
-A Data Clustering Algorithm

Bin Zhang, Meichun Hsu, Umeshwar Dayal
Hewlett-Packard Research Laboratory
June 28, 1999

Abstract
Data clustering is one of the common techniques used in data mining. A popular performance function for measuring goodness of data clustering is the total within-cluster variance, or the total mean-square quantization error (MSE). The *K*-Means (*KM*) algorithm is a popular algorithm which attempts to find a K-clustering which minimizes MSE. The K-Means algorithm is a center-based clustering algorithm. The dependency of the K-Means performance on the initialization of the centers is a major problem; a similar issue exists for an alternative algorithm, Expectation Maximization(*EM*), although to a lesser extent. In this paper, we propose a new clustering method called the *K*-Harmonic Means algorithm (*KHM*). *KHM* is a center-based clustering algorithm which uses the Harmonic Averages of the distances from each data point to the centers as components to its performance function. It is demonstrated that *K*-Harmonic Means is essentially insensitive to the initialization of the centers. In certain cases, *K*-Harmonic Means significantly improves the quality of clustering results comparing with both *K*-Means and *EM*, which are the two most popular clustering algorithms used in data exploration and data compression. A unified view of the three performance functions, *K*-Means', *K*-Harmonic Means' and *EM*'s, are given for comparison. Experimental results of *KHM* comparing with *KM* on high dimensional data and visualization of the animation of the convergence of all three algorithms using 2-dimensional data are given.

Keywords – Clustering, *K*-Means, *K*-Harmonic Means, Data Mining.

1. Introduction

Clustering has applications in many different areas like data mining [FPU96], statistical data analysis [KR90], data compression and vector quantization [GG92], and many others. *K*-Means (*KM*), first developed more than three decades ago [M67], and the Expectation Maximization (*EM*) with linear mixing of Gaussian distributions [DLR77] are the most popular clustering algorithms [BFR98a], [SI84], [MK97]. See [GG92] for more complete references for *K*-Means and [MK97][RW84] for *EM*.

K-Harmonic Means (*KHM*), like *KM* and *EM*, is a center-based, iterative algorithm that refines the clusters defined by *K* centers. *K*-Harmonic Means takes the sum over all data points of the harmonic average of the squared distance from a data point to all the centers as its performance function (see formula (1)), which is different from *the total with-in cluster variance* used by *KM*. Let $M = \{m_l \mid l=1,\ldots,K\}$ be K centers and $S = \{x_i \mid i=1,\ldots,N\}$ be N given data points, the *K*-Harmonic Means' performance function is

$$Perf_{KHM}(\{x_i\}_{i=1}^{N}, \{m_l\}_{l=1}^{K}) = \sum_{i=1}^{N} \frac{K}{\sum_{l=1}^{K} \frac{1}{\| x_i - m_l \|^2}}. \qquad (1)$$

The quantity inside the outer summation is the harmonic average of K squared distances, $\{||x - m_l||^2 \mid l = 1, \ldots, K\}$. To give a reason for choosing this function, we briefly review the concept of *harmonic average* (also called harmonic mean) in the next section.

For all three algorithms, the computation starts with an initialization of the center positions and followed by iterative refinement of these positions. Many experimental results show that *KHM* is essentially insensitive to the initialization of the centers than *KM* and *EM*. The dependency of the K-Means performance on the initialization of the centers is a major problem; a similar issue exists for an alternative algorithm, Expectation Maximization(*EM*), although to a lesser extent. Many papers have been published to find good initializations for *KM* [BF98]. This paper takes a totally different approach by changing *MIN()* used in *KM* to *HA()* (Harmonic Average), which is similar to *MIN()* but "softer", to make the performance function "easier to optimize" by an algorithm that is essentially insensitive to initialization. More explanation will be given later.

The rest of the paper is organized as follows:

- Harmonic Average *HA();*
- comparing with the *MIN();*
- comparing *KHM* performance function with *KM*;
- *KHM*' algorithm;
- implementation issues;
- computational complexity per iteration;
- *EM* algorithm based on linear mixing (limited version);
- a unified view of all three performance functions – *KM*', *KHM*' and *EM*'s;
- different ways of mixing bell shape functions;
- experimental results.

2. Harmonic Average

Harmonic Average (*HA*) has been known for a long long time. The harmonic average of K numbers is

$$HA(\{a_i \mid i = 1, \ldots, K\}) = \frac{K}{\sum_{i=1}^{K} \frac{1}{a_i}}, \qquad (2)$$

the reciprocal of the arithmetic average of the reciprocals of the numbers in the set.

Here is an example on when the harmonic average occurs. A person driving a car on a highway noticed that between the adjacent two exits he has been driving at average speed of *60* miles/hr, *70* miles/hr and *75* miles/hr (he passed four exits). The exits are equally distanced of *A* miles apart. What is the average speed between the first exit and the last exit he passed? The total number of miles between the first and the last exit is *(A+A+A)*. The number of hours used to pass two adjacent exits are *A/60*, *A/70* and *A/75*. The total amount of time used between the first exit and the last exit is *(A/60 + A/70 + A/75)*.

The average speed = (total number of miles)/(total number of hours)
= *(A + A + A)/(A/60 + A/70 + A/75) = 3/(1/60 + 1/70 + 1/75)*
= The harmonic average of *60*, *70* and *75*.

3. Comparing *HA()* with *MIN()*

We compare *HA()* with *MIN()* in the first quadrant of a *K*-dimensional space, $\boldsymbol{Q_1} = \{ (a_1,\ldots\ldots,a_K) \mid a_i \quad 0 \text{ for } i=1, \ldots,K\}$. The comparison in this section provides the basis for comparing the *KHM* performance function with *KM* which uses *MIN()*.

The plot of *HA()* is very similar to that of *MIN()*. Figure 1 has a plot of *MIN()* and *HA()/K* for *K=2*.

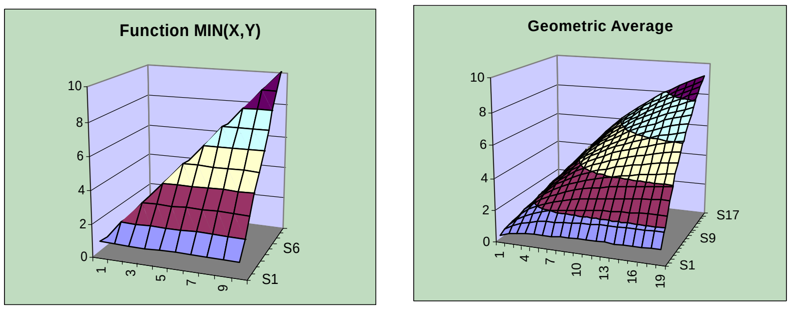


Figure 1. The plots of *MIN()* on the left and *HA()/K* on the right.

Analytically, we show that *HA()* and *MIN()* are the same on the boundary of $\boldsymbol{Q_1}$ and their derivatives are also "close". On any boundary, where at least one of the coordinates is zero, both *MIN()* and *HA()* equal to zero (the boundary value of *HA()* can be defined by the limit, which always exists, approaching from inside of the first quadrant). In the interior, we compare their gradient. Since the value of $MIN(a_1,\ldots\ldots,a_K)$ is determined by the smallest value of a_l*'s*, the derivative of $MIN(a_1,\ldots\ldots,a_K)$, with respect to the smallest variable is *1* and all others zero.

The derivative of $HA(a_1,\ldots\ldots,a_K)$, with respect to the smallest variable is

$$\frac{\partial HA(\{a_i \mid i=1,\ldots,K\})}{\partial a_{\min}} = \frac{K}{a_{\min}{}^2 (\sum_{i=1}^{K} \frac{1}{a_i})^2} = \frac{K}{(1+\sum_{i\neq\text{'min'}}^{K} \frac{a_{\min}}{a_i})^2} \xrightarrow[a_{\min}/a_i \to 0]{} K.$$

and w.r.t. a variable other than the smallest one, is

$$\frac{\partial HA(\{a_i \mid i=1,\ldots,K\})}{\partial a_k} = \frac{1}{a_k{}^2 (\sum_{i=1}^{K} \frac{1}{a_i})^2} = \frac{1}{(1+\sum_{i\neq k}^{K} \frac{a_k}{a_i})^2} = O(\frac{1}{(a_k / a_{\min})^2}) \xrightarrow[a_k/a_{\min} \to +\infty]{} 0,$$

where a_{min} = min{a_i | $i=1,\ldots,K$}. The derivative of *HA()* differs from the derivative of *MIN()* significantly only near "the main diagonal" – $a_1 = a_2 = \ldots\ldots = a_K$. When the value of a variable is relatively large comparing with a_{min}, the harmonic average is less sensitive to its change. The last formula shows the exact rate the derivative of *HA()* diminishes.

4. Comparing *KHM'* Performance Function with *KM'*

KM' performance function is

$$Perf_{KM}(\{x_i\}_{i=1}^{N},\{m_l\}_{l=1}^{K}) = \sum_{l=1}^{K}\sum_{x\in S_l} \| x - m_l \|^2, \tag{3}$$

where S_l is the subset of *x's* that are closest to m_l than any other centers in *M*. (Or $\{S_l|l=1,\ldots,K\}$ is the Voronoi partition given by the *K* centers). The double summation in (3) can be considered as a single summation over all *x* (data points) and the squared distance under the summations can be expressed by *MIN()*. Therefore, the *KM* performance function can be rewritten as

$$Perf_{KM}(\{x_i\}_{i=1}^{N},\{m_l\}_{l=1}^{K}) = \sum_{i=1}^{N} MIN\{\| x_i - m_l \|^2 \mid l = 1,\ldots,K\}, \tag{4}$$

(The name "*K*-Means" come from the algorithm that finds a local optimal of this performance function. From the last formula, the *KM*' performance function can be called *K-MIN*'s, which provides the comparison with the name – "*K*-Harmonic Means"). Replacing *MIN()* by *HA(),* we get the performance function of *KHM*:

$$Perf_{KHM}(\{x_i\}_{i=1}^{N},\{m_l\}_{l=1}^{K}) = \sum_{i=1}^{N} HA\{\| x_i - m_l \|^2 \mid l = 1,\ldots,K\} = \sum_{i=1}^{N} \frac{K}{\sum_{l=1}^{K} \frac{1}{\| x_i - m_l \|^2}}.$$

The distance function used in this paper is the squared Euclidean distance, $||x - m||^2$, but generalizing to other distance functions are possible.

A unified view of the *KM*', *KHM*' and *EM*'s performance functions is given later in Section 9, in which all are considered as ways of mixing bell-shape functions. Linear mixing of bell-shape functions is used in *EM* [MK97].

5. The *KHM* Algorithm

There can be many different optimization algorithms to find a local optimum of a non-linear function. In this section, we derive an algorithm for *KHM*' performance function which is very insensitive to initialization.

To derive the optimization algorithm, we take partial derivatives of the *KHM*'s performance function in (1) with respect to the center positions, m_k, $k=1,\ldots,K$, and setting them to zero, (to simplify the notations, let $d_{i,l} = ||x_i - m_l||$),

$$\frac{\P Perf_{KHM}(X,M)}{\P m_k} = -K * \sum_{i=1}^{N} \frac{4*(x_i - m_k)}{d_{i,k}^{\ 3} (\sum_{l=1}^{K} \frac{1}{d_{i,l}^{\ 2}})^2} = \vec{0}.$$

(An arrow is put on top of the zero on the right to show it is a zero vector. The center positions, m_k, are also vectors.)

"Solving" m_k's from the last set of equations, we get a recursive formula: (Reminder: $d_{i,l} = ||x_i - m_l||$ on the right of (5) are still functions of the center positions.)

$$m_k = \frac{\sum_{i=1}^{N} \frac{1}{d_{i,k}^{\ 3} (\sum_{l=1}^{K} \frac{1}{d_{i,l}^{\ 2}})^2} x_i}{\sum_{i=1}^{N} \frac{1}{d_{i,k}^{\ 3} (\sum_{l=1}^{K} \frac{1}{d_{i,l}^{\ 2}})^2}}. \tag{5}$$

This is the *KHM*' recursive formula. The *KHM* algorithm starts with a set of initial positions of the centers, $d_{i,l} = ||x_i\text{-}m_l||$ are calculated, and then the new positions of the centers are calculated using (5) or from the decomposed sequence below (implementation details are given later),

$$a_i = \frac{1}{(\sum_{l=1}^{K} \frac{1}{d_{i,l}^{\ 2}})^2}, \qquad q_{i,k} = \frac{a_i}{d_{i,k}^{\ 3}}, \qquad q_i = \sum_{k=1}^{K} q_{i,k}, \qquad p_{i,k} = \frac{q_{i,k}}{q_i}, \qquad m_k = \sum_{i=1}^{N} p_{i,k} x_i. \tag{6.1 - 6.5}$$

The recursion is continued until the performance value stabilizes.

Numerical results show a super-linear convergence rate. The proof of convergence is still in progress.

6. Implementation of *KHM*

A naïve implementation of the *KHM* algorithm tends to encounter numerical difficulties due to the reciprocals, $1/||x\text{-}m||^2$, in the recursion formula. Proper calculation of the coefficients is important for a successful implementation. The calculation of $q_{i,k}$'s (combination of (6.1) and (6.2)), where the difficulties occur, are done as follows:

$$q_{i,k} = \frac{d_{i,\min}^{\ \ 4}}{d_{i,k}^{\ 3}[1 + \sum_{l \neq \min} (\frac{d_{i,\min}}{d_{i,l}})^2]^2} = \frac{(\frac{d_{i,\min}}{d_{i,k}})^3 * d_{i,\min}}{[1 + \sum_{l \neq \min} (\frac{d_{i,\min}}{d_{i,l}})^2]^2} \tag{7}$$

where

$$d_{i,\min} = MIN(d_{i,l} \mid l = 1,...,K).$$

All the ratios ($d_{i,min}/d_{i,l}$) are in [0,1]. The procedure is given below:

```
**Calculate_q_vector**($x_i$,$M$): /*  Index $i$ is fixed inside this function.
                                        This function is called for each data point. */
Step 1: calculate $d_{i,k}= ||x_i-m_k||$, for $k=1,…,K$.
Step 2: search for $d_{i,min}=min\{ d_{i,k} | k=1,…,K\}$.
Step 3: form vector < $d_{i,min}/ d_{i,k}$ | $k=1,…,K$> ($d_{i,min}/d_{i,min}$ is always set to 1. If $d_{i,min}=0$, all other
        components are set to zero.)
Step 4: calculate the q vector from (7).
```

The rest of the implementation, (6.3) to (6.5), is straightforward.

7. Computational Costs of *KHM* in Each Iteration

In each iteration, calculating all the pairwise distances from N data points to K centers (of D dimensional vectors) costs $O(N*K*D)$. *KM* and EM (linear mixing) share the same cost on this part. After getting the coefficients $p_{i,k}$, calculating the linear combinations, $m_k = \quad p_{i,k}*x_i$, costs another $O(N*K*D)$. *EM* costs the same on this part. *KM* costs less ($O(N*D)$) on this due to the partitioning but an additional $O(N*K)$ comparison and assignment (marking) operations are used to do the partitioning. After calculating the distances, all quantities used in the algorithm no longer depend on the dimension and all other costs are $O(N*K)$. The leading asymptotic term for all three algorithms are the same, $O(N*K*D)$.

The asymptotic computational complexity *per iteration* for *KM*, *KHM* and *EM* (linear mixing model) are all $O(N*K*D)$. It is the convergence rate and the convergence quality (dependency on the initialization) which differentiate them in real world applications.

(Note: due to the partitioning nature, faster algorithms/implementations have been designed for *KM* using trees to do spatial partition of either the centers or the data [GG92],[PM99].)

Space complexity of KHM is NxD for data points, KxD for the K centers and $KxD+2*K$ for temporary storage. The temporary storage requirement tend to be lower than *KM* because the later needs a $O(N)$ temporary storage to keep the membership information and $N>>K$ in real problems.

8. The *EM* Clustering Algorithm Based on Linear Mixing of Gaussian Distributions

We briefly review a limited version of the *EM* algorithm needed later for comparison with *KHM* and *KM* later. We limit ourselves to the *EM* algorithm with linear mixing of K identical spherical bell-shape (Gaussian distribution) functions because this limited version of *EM* matches the version of *KM* and *KHM* (see Section 9 and 10 for more details). Using general form (non-identical, non-spherical) bell shape functions in the three algorithms is beyond the scope of this paper.

Let

$$Perf_{EM}(X,M) = -\log(\prod_{i=1}^{N}[\sum_{l=1}^{K} p_l * \frac{1}{\sqrt{(2p)^D}} EXP(-\| x-m \|^2)]).$$

(8)

a linear mixing of *K* identical spherical bell-shape functions. *EM* algorithm is a recursive algorithm with the following two steps:

E-Step:

$$p(m_l \mid x_i) = \frac{p(x_i \mid m_l) * p(m_l)}{\sum_{i=1}^{N} p(x_i \mid m_l) * p(m_l)},$$

(9)

where *p(x|m)* is the prior probability with Gaussian distribution, *p(m$_l$)* is the mixing probability.

M-Step:

$$m_l = \frac{\sum_{i=1}^{N} p(m_l \mid x_i) * x_i}{\sum_{i=1}^{N} p(m_l \mid x_i)},$$

(10)

$$p(m_l) = \frac{1}{N}\sum_{i=1}^{N} p(m_l \mid x_i),$$

(11)

where *N* is the size of the whole data set.

For more details, see [MK97] and the references there.

9. A Unified View of The Three Performance Functions

Without introducing any change, applying the identity mapping *-log(EXP(-( )))* to the performance functions of *KM* and *KHM*, we get

$$Perf_{KM}(X,M) = -\log(\prod_{i=1}^{N} EXP(-MIN\{\| x-m \|^2 \mid m \in M\}));$$

(12)

$$Perf_{KHM}(X,M) = -\log(\prod_{i=1}^{N} EXP(-HA\{\| x - m \|^2 \mid m \in M\})).$$
(13)

Now they share the same form of the *EM*'s performance function:

$$Perf_{EM}(X,M) = -\log(\prod_{i=1}^{N}[\sum_{l=1}^{K} p_l * \frac{1}{\sqrt{(2p)^D}} EXP(-\| x - m \|^2)]).$$
(14)

The quantity inside the brackets "[]" in (14) is the linear mixing of the bell-shape functions – the *EXP()*'s. Comparing with EM's performance function, we can look at the performance functions of *KM* and *KHM* also as mixings of bell-shape functions. (The extra factor, 1/sqrt[(2)^D], does not have any real impact because it only change the performance function by adding a constant to it, which does not change the locations of the optima of the function. It is OK to either add the factor to *KM* and *KHM's* performance functions, or remove the factor from the *EM*'s. We choose not to change any of them.)

10. MIN Mixing, Harmonic Mixing and Linear Mixing of Bell-shape Functions

The performance functions of *KM* and *KHM* can be viewed as a (log of) mixing of bell-shape functions. Define

MIN Mixing*:*

$$EXP(-MIN\{\| x - m_l \|^2 \mid l = 1,...,K\})$$
(15)

Min-Mixing can also be called Max-Mixing because *EXP(-x)* is monotone decreasing and

$$EXP(-MIN\{\| x - m_l \|^2 \mid l = 1,...,K\}) = MAX\{EXP(-\| x - m \|^2) \mid l = 1,...,K\}.$$
(16)

Harmonic Mixing:

$$EXP(-HA\{\| x - m_l \|^2 \mid l = 1,...,K\})$$
(17)

Linear Mixing:

$$\sum_{l=1}^{K} p_l * \frac{1}{\sqrt{(2p)^D}} EXP(-\| x - m_l \|^2)$$
(18)

A 2-dimensional and four-center harmonic mixing is plotted in **Figure 2**.

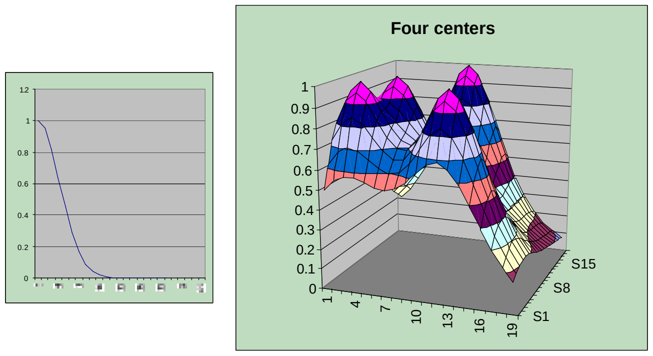


Figure 2. The left picture is half of the bell-shape curve.
The right is the 2-D Harmonic Mixing with four centers.

One quick observation is that *EM*'s linear mixing has more parameters -- the mixing probabilities, p_l, to be estimated (See (18)). This is one reason that the *EM* algorithm converges slowly. (See [MK97] p105, for an explanation of the slow convergence of *EM* algorithm.)

For both MIN-Mixing and Harmonic Mixing, the maximums of the components (the individual bells) match the maximums of the mixture. This can be proved easily by setting the gradient of the mixing functions to zero. It is also shown in Figure 4 for a particular example. But for linear mixing, they do not match -- the maximums of the mixture shift towards the centroid of the maximums of the components as shown in Figure 3. When Linear Mixing *EM* is used, the centers of the bell shape functions are biased from the true centers of the local densities of data (because the maximums of the mixture is intended to match the local densities of data).

Three different mixings of two bell shape functions in one-dimensional space are plotted in each plot in Figure 3. Four different distances between the centers of the bell shape functions are shown. As the two centers move closer, the difference among the three mixtures increase. The two peaks in linear mixing even merge into a single peak (see the plot in upper-left corner). For the other two mixings, however, they remain to have two peaks no matter how close the centers are (analytical proof is straightforward – just calculate the Hessian matrix and show it is negative definite). When the centers are far apart, all three mixtures are very close. This explains that when the number of centers matches the number of naturally well separated clusters, all three algorithms tend to give very similar results (See Experiments in Group 1 on BIRCH Data Set).

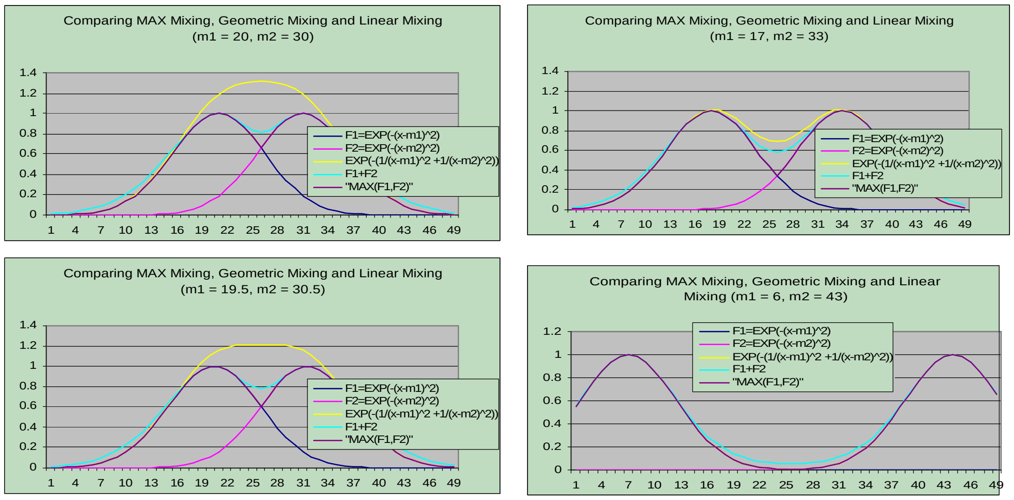


Figure 3. Comparing Three Different Mixings of Two Bell-shape Functions in One Dimension. Color coding: brown – *MIN Mixing in KM* (hard to see because it perfectly overlap with the individual bells), light blue – Harmonic Mixing in *KHM*, and yellow – Linear Mixing in *EM.* Two individual bells are in pink and dark blue. As the centers move from far to near, the difference among the three mixings increase.

For finding clusters, linear mixing (the limited version with fixed covariance matrix of the bells) does not behave properly when centers get too close to each other. As two or more local peaks merge into a single peak, the maximum of that single peak behaves like a ghost center and the individual centers loose their identity. This is clearly shown in our experimental results (See the experimental results from *EM* in Group 2 and Group 4).

(Note: In the *EM* with a general covariance matrix as parameters, this problem seems to be less severe because the diameter of the bells can shrink as the centers get closer, if not trapped by local optimums. But the number of parameters to be estimated grow at a quadratic rate with the number of centers, which creates a much more severe problem.)

11. Experimental Results

Two data sets are used: the BIRCH data set is from UC Irvine [B99]; and another data set, Hier, is generated by a Hierarchical Data Generator (HDG).

The HDG maps each data point in a given data set into a cluster of points with either uniform or normal distribution. We call this mapping Zoom-In. The variance of the clusters can be set by user. When this mapping is repeatedly applied, the clustering of the final resulted data set shows a hierarchical structure. (Comparing it with the clustering of matter in the Universe.)

The detailed information on these data sets is given in Table 1.

Name	Size (points)	Structure
BIRCH	100,000	100 clusters in a 10x10 grid with 1000 points in each cluster in normal distribution. The radius of each cluster is sqrt(2) and the

		neighboring clusters are 4*sqrt(2) apart.
Hier	20,000	Started with 4 vertices of a square, each one is mapped into 100 points uniformly distributed, then each of the 400 points is mapped into 50 points uniformly distributed.

Table 1. The Data Sets.

We choose 2-dimensional data in this presentation for the power of visualization. It has been experienced widely that a single performance value does not provide enough information for judging the quality of an algorithm. We have applied *KHM* on 40-dimensional data with 150,000 points. It also showed better convergence (less sensitive to initialization).

Twelve experiments conducted are organized in four groups:
Group 1 shows the quality of the three algorithms on the BIRCH data set when the "correct" number of centers are used. All three algorithms started from the same random initialization. The centers under EM is "more mobile" than under *KM*. They are most mobile under *KHM*. From the plots (given at the end of this paper), eight pairs of centers are trapped under *KM* (two centers found in one cluster) after it has already converged to a local optimum. Only one pair of centers is trapped under *KHM*; and 4 pairs under *EM*. The results are comparable under many different random initializations used (only one set is presented here).

Group 2 shows that when the initialization is very bad, both *KM* and *EM* converges very slowly (if converge to anything meaningful at all). All three algorithms started from the same bad initialization. *KM* moves the centers out "layer-by-layer" like peeling an onion. *KHM* converges very fast and reached a configuration that is close to the global optimum in about 40 iterations. *EM* does not work well when the centers are close to each other (the reason was given earlier – the individual bell shape functions loose their identity and are merged into a big bell shape function with a ghost center). Most centers out of the 2000 centers merged with others. The problem could be fixed by allowing the variance ("diameter") of the bell-shape functions to change. But that will introduce a lot more parameter(s), which cause new convergence problems.

Group 3 shows that *KHM* can start from really bad initializations. In the Bad Initialization, 400 centers are linearly spread out on a small line segment out the region occupied by data. *KHM* converged nicely. *KM* and *EM* do not work under this bad initialization. A random initialization is used for them.

Group 4 repeat the same experiment as Group 1 but with a bad initialization instead of the random initialization. In the Bad Initialization, 100 centers are linearly spread out on a small line segment in the center of all data (See the first plot in Group 4). *KHM* converged nicely in 90 iterations with only two pairs trapped. *KM* and *EM* do not converge well even after 300 iterations.

Exp. Group#	Algorithm	Data Set	#of Centers	#of Iterations of the snapshots	Local Opt.?	Initia-lization	Conver-gence
1	*KM*	BIRCH	*100*	*50*	Yes	Random	Stopped
	KHM			*50,100,120*	No*		Stabilized
	EM			*50,100*	No*		Stabilized
2	*KM*	BIRCH	*2000*	*20,60,100,200*	No	Bad Init.	
	KHM			*5,10,40,200*	No*		Stabilized
	EM			*20,60,100,200*	No		

3	*KM*	Hier	*400*	*45*	Yes	Random	Stopped
	KHM			*15,25,50*	No*	Bad Init.	Stabilized
	EM			*100*	No*	Random	Stabilized
4	*KM*	BIRCH	*100*	*10,25,100,300*	No	Bad Init.	
	KHM			*10,25,40,90*	No*		Stabilized
	EM			*10,25,100,300*	No		

No* -- See the last column.

Table 2. The Setup of the Experiments.

Overlay is used in all plots. The given data set is plotted in light yellow as the background (The BIRCH data set is reduced to *20,000* data points, or *1/5* of the original, before plotting for better visibility of the clusters). The centers are plotted in red.

Animation was done with all the convergence paths. The number of iterations listed in Table 2 are the maximum number of iterations. Due to the limitation of printing on paper, we provide only a few snapshots for each experiment.

Initializations of the centers for all experiments are also plotted.

The title bar of each figure is encoded as:
Group#: Algorithm, Data Set, Number of Centers, Number of Iterations Done, Initialization.

12. Experiment Using High Dimensional Data

High dimensionality creates new challenges to data clustering. We conducted an experiment using a 40 dimensional data set with 150,000 data points. Starting from the same random initialization of 100 centers, two experiments are run: 1) KM – stopped at a local optimum after 8 iterations; 2) KHM first with 20 iterations and then switch to KM which stopped after 7 iterations at a local optimum. The results are shown in the following table. The results show that *KHM* helped *KM* to avoid a bad local optimum.

#	Algorithm	Number of Iterations	Final Performance Value (KM's)	Note
1	KM	8	96152	Local optimum
2	KHM/KM	20/7	85485	Local optimum

13. Future Research

Geometrically, *K*-Harmonic Means' performance function is intuitive. Is there a statistical interpretation of the performance function like the one *K*-Means has?

K-Harmonic Means is a new clustering algorithm with a new performance function. Does it provide new opportunities for a scale-up of K-Harmonic Means for very large data sets?

In a number of experiments, it is consistently shown that *K*-Harmonic Means is very insensitive to the initialization and *K*-Harmonic Means converge faster than *K*-Means when the initialization is far from a local optimal (of *K*-Means). But experience also show that *K*-Means converge very fast when the initialization is close to a local optimum. *K*-Means performance function is popular and with clear interpretation. Putting this information together, it suggests combining *K*-Harmonic Means with *K*-Means to take advantage of both. How many iterations should be run using *KHM* before switching to *KM*?

References

[A73] Anderberg, M. R. 1973. Cluster analysis for applications. Academic Press, New York. xiii + 35p.

[B99] Bay, S. D. (1999). The UCI KDD Archive [http://kdd.ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science.

[BFR98] Bradley, P., Fayyad, U. M., and Reina, C.A., "Scaling EM Clustering to Large Databases," MS Technical Report, 1998.

[BF98] Bradley, P., Fayyad, U. M., C.A., "Refining Initial Points for KM Clustering", MS Technical Report MSR-TR-98-36, May 1998.

[BFR98a] Bradley, P., Fayyad, U. M., and Reina, C.A., "Scaling Clustering to Large Databases", KDD98, 1998.

[DH72] Duda, R., Hart, P., "Pattern Classification and Scene Analysis", John Wiley & Sons, 1972.

[DLR77] Dempster, A. P., Laird, N.M., and Rubin, D.B., "Miximum Likelyhood from Incomplete Data via the EM Algorithm", Journal of the Royal Statistical Society, Series B, 39(1):1-38, 1977.

[FPU96] Fayyad, U. M., Piatetsky-Shapiro, G. Smyth, P. and Uthurusamy, R., "Advances in Knowledge Discovery and Data Mining", AAAI Press 1996

[GG92] Gersho & Gray, "Vector Quantization and Signal Compression", KAP, 1992

[GMW85] Gill, P.E., Murray, W. and Wright, H.M., "Practical Optimization", Academic Press, 1981.

[G85] Gonzales, T.F., "Clustering to Minimize the Maximum Intercluster Distance", Theo. Comp. Sci. 38, p293-306, 1985.

[KR90] Kaufman, L. and Rousseeuw, P. J., "Finding Groups in Data : An Introduction to Cluster Analysis", John Wiley & Sons, 1990.

[M67] MacQueen, J. 1967. Some methods for classification and analysis of multivariate observations. Pp. 281-297 *in*: L. M. Le Cam & J. Neyman [eds.] Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, Vol. 1. University of California Press, Berkeley. xvii + 666 p.

[MA] McKenzie, P. and Alder, M., "Initializing the EM Algorithm for Use in Gaussian Mixture Modeling", The Univ. of Western Australia, Center for Information Processing Systems, Manuscript.

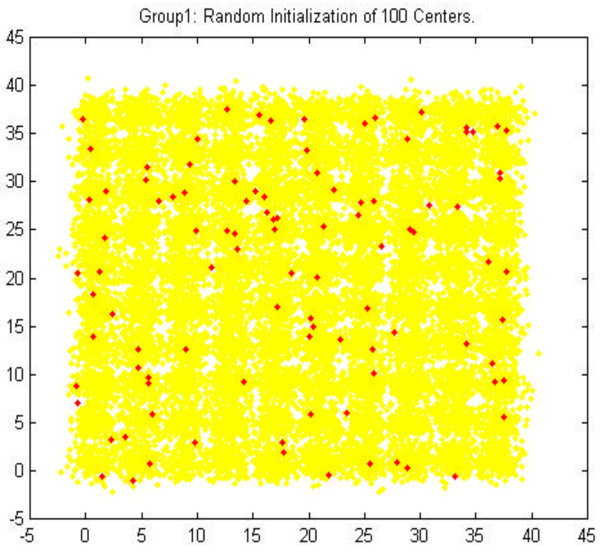
[MK97] McLachlan, G. J. and Krishnan, T., "The EM Algorithm and Extensions.", John Wiley & Sons, Inc., 1997

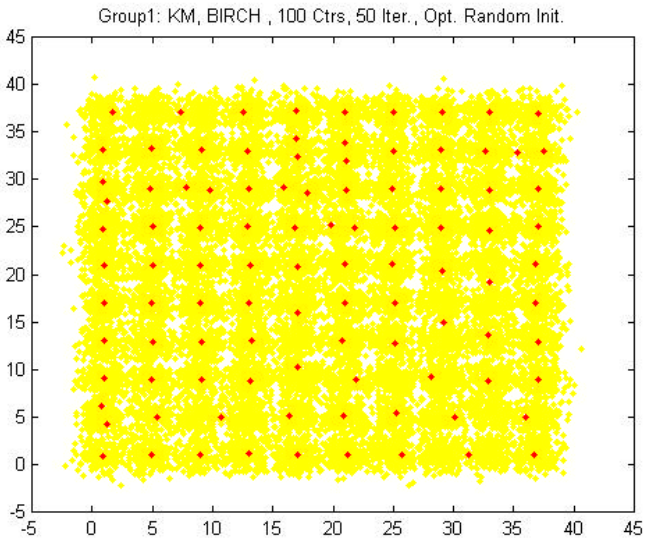
[PM99] Pelleg, D. and Moore, A, "Accelerating Exact *k*-means Algorithms with Geometric Reasoning", KDD-99, Proc. of the Fifth ACM SIGKDD Intern. Conf. On Knowledge Discovery and Data Mining, page 277-281.

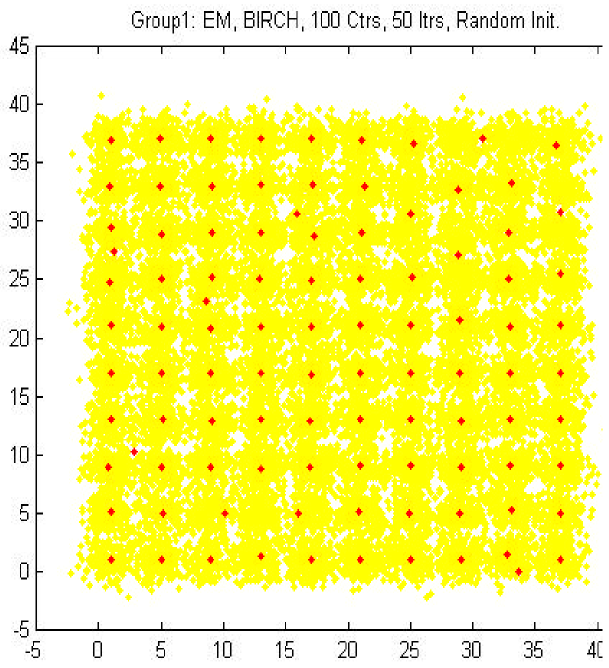
[RW84] Rendner, R.A. and Walker, H.F., "Mixture Densities, Maximum Likelihood and The EM Algorithm", SIAM Review, vol. 26 #2, 1984.

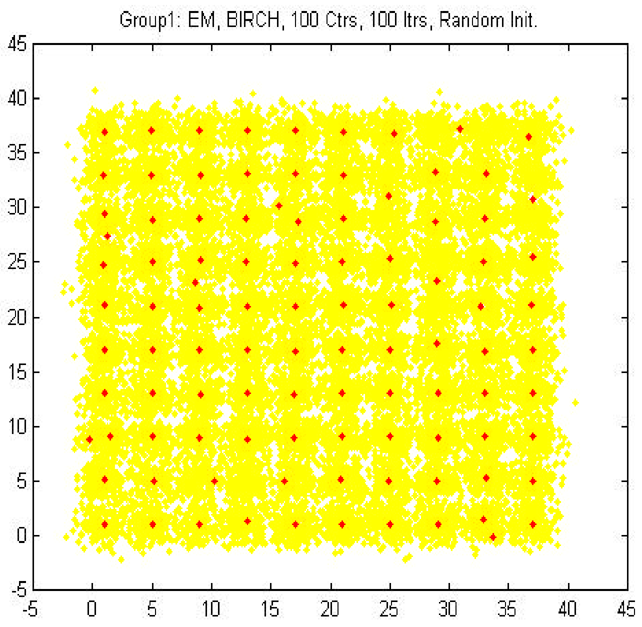
[SI84] Selim, S.Z. and Ismail, M.A., "*K*-Means-Type Algorithms: A Generalized Convergence Theorem and Characterization of Local Optimality", IEEE Trans. On PAMI-6, #1, 1984.

Group 1: #1 - Random Initialization used for all three experiments in Group 1.
#2 - KM converged to a local optimal in 50 iterations and stoppped by itself.
#3 - EM after 50 iterations.
#4 - EM after 100 iterations.

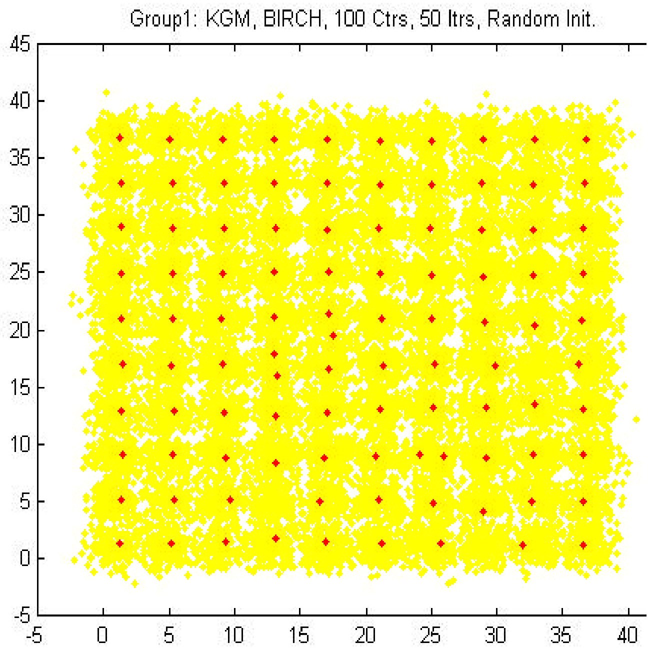


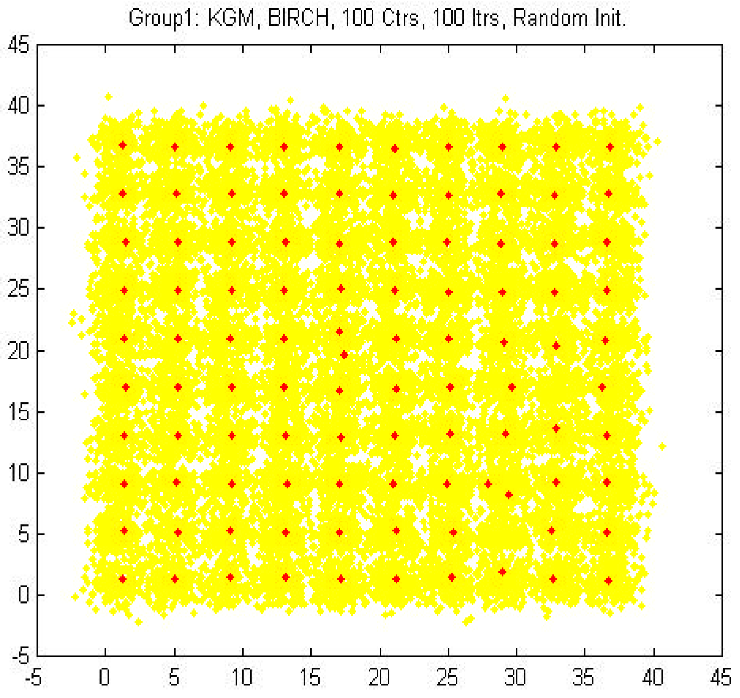


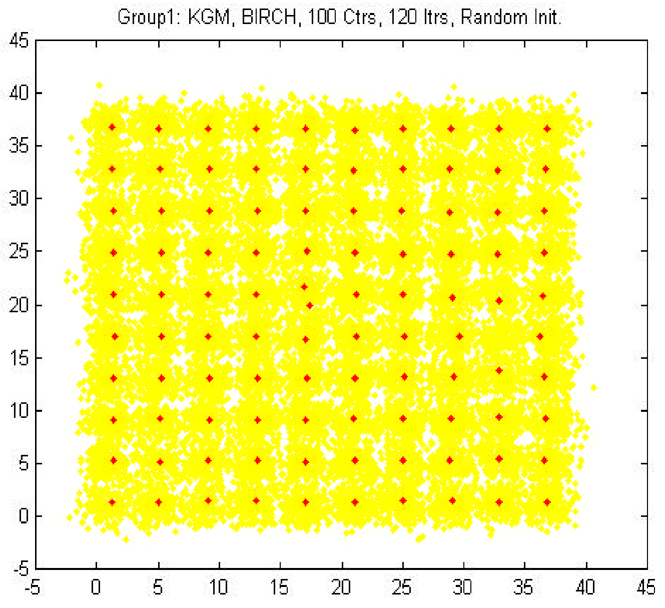




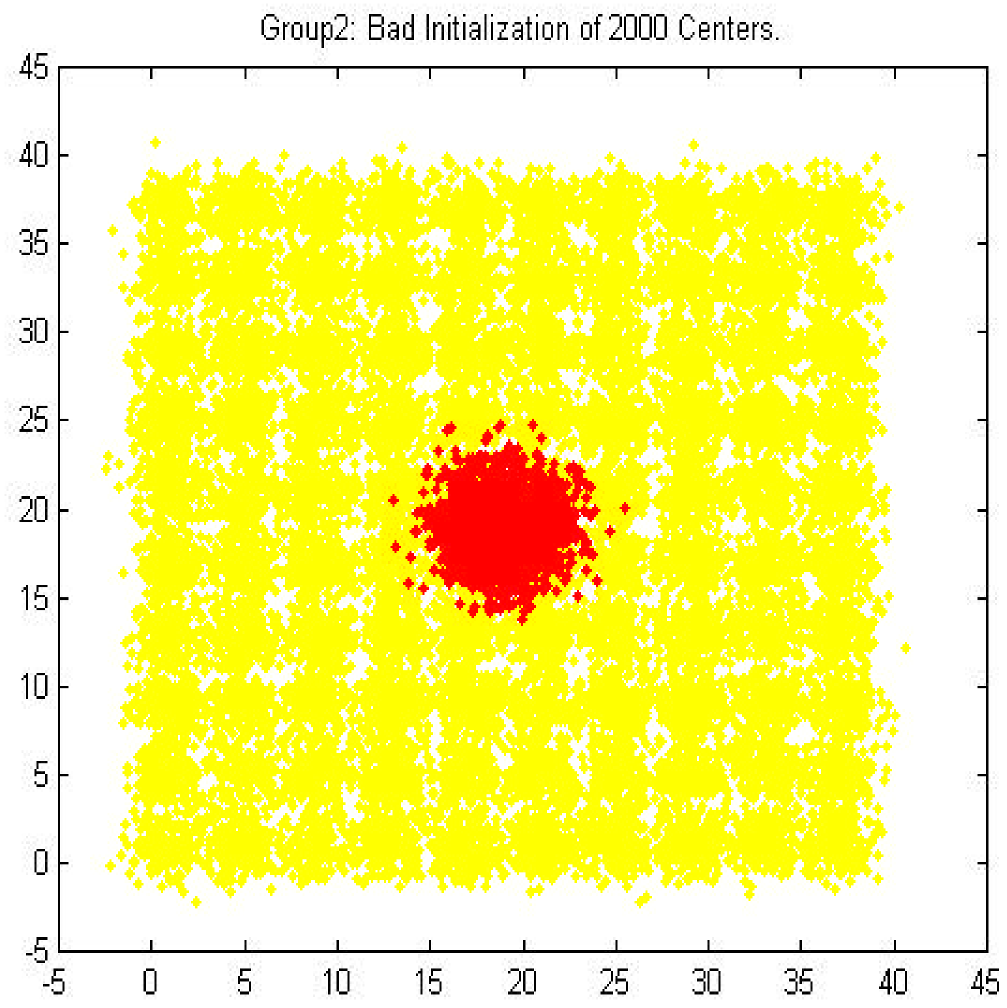
Group 1: #1 - KGM after 50 iterations.
#2 - KGM after 100 iterations.
#3 - KGM after 120 iterations.



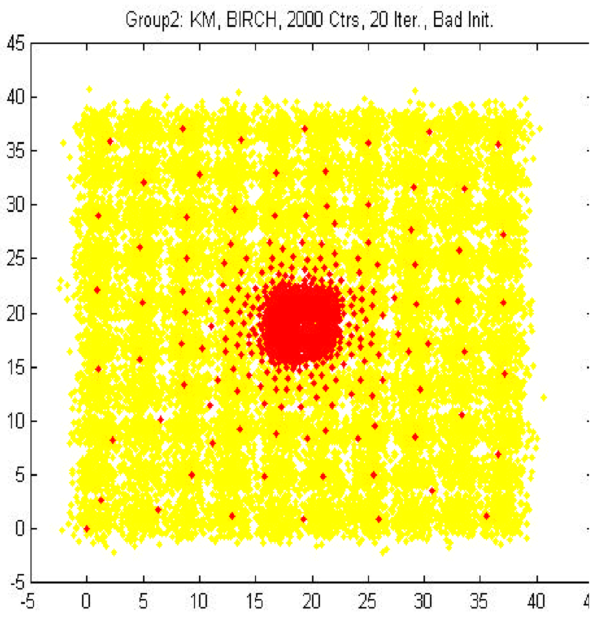


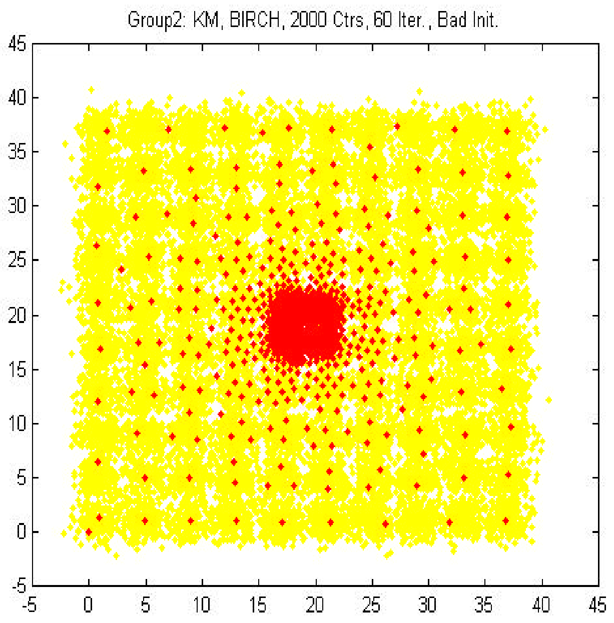


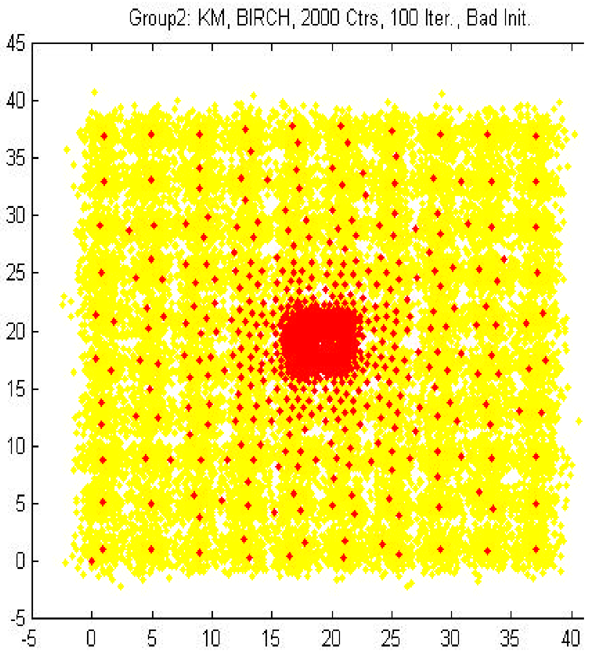
Group 2: The random initialization used for all experiments in Group 2.

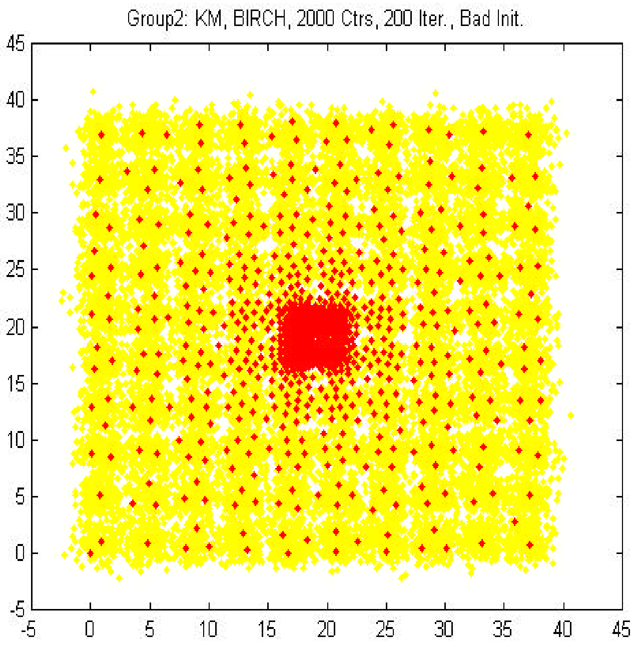


Group 2: #1 - KM after 20 iterations.
#2 - KM after 60 iterations.
#3 - KM after 100 iterations.
#4 - KM after 300 iterations.

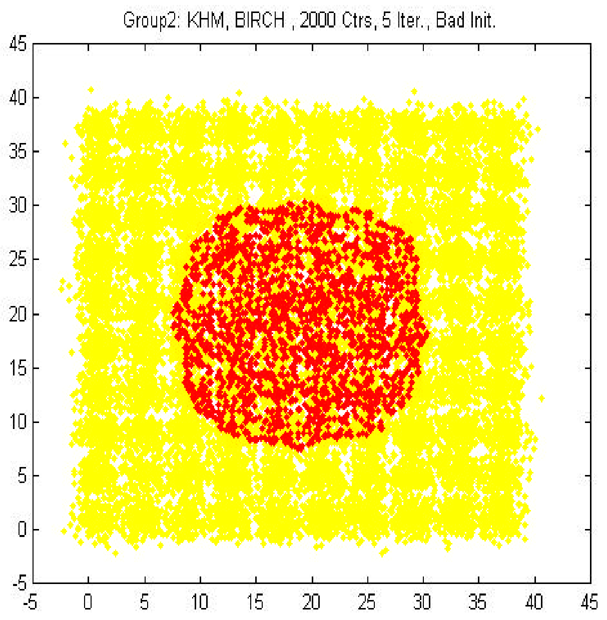


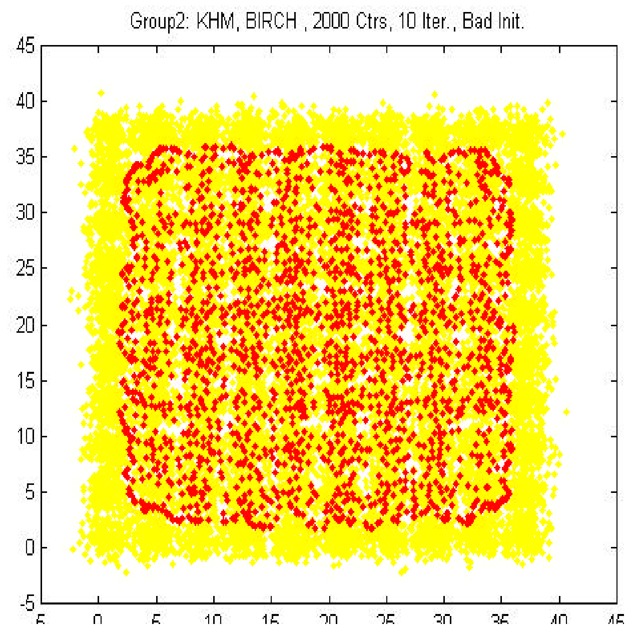


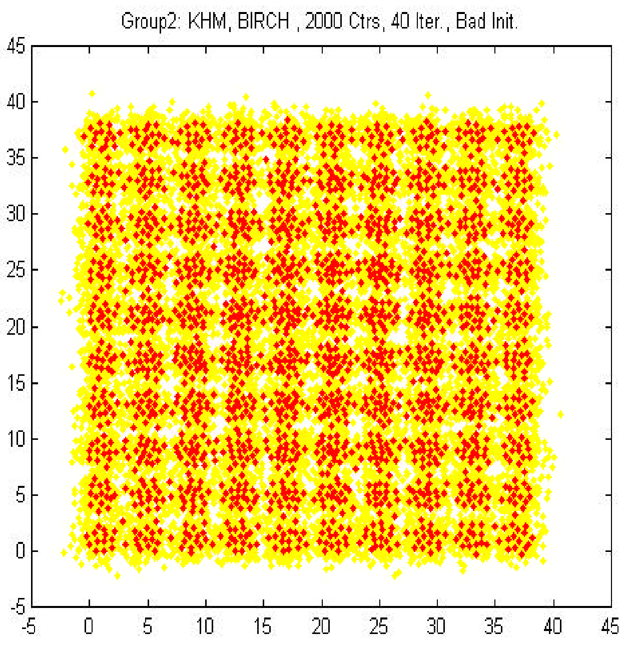


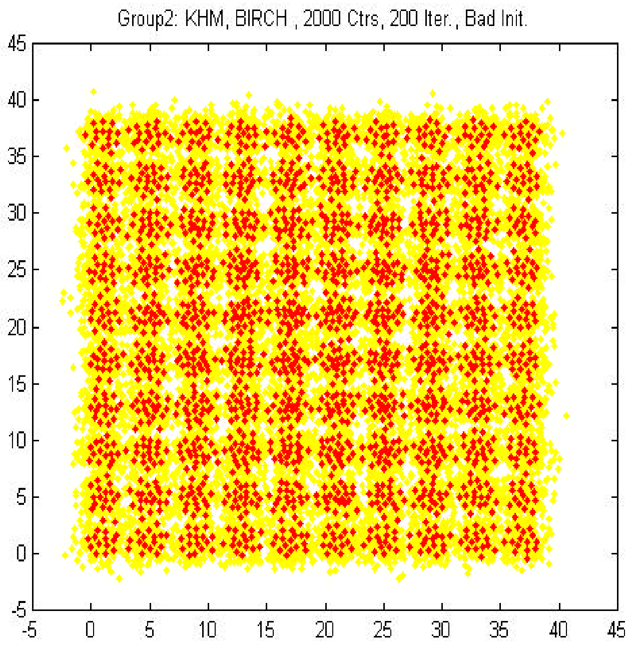


Group 2: #1 - KGM after 5 iterations.
#2 - KGM after 10 iterations.
#3 - KGM after 40 iterations.
#4 - KGM after 200 iterations.

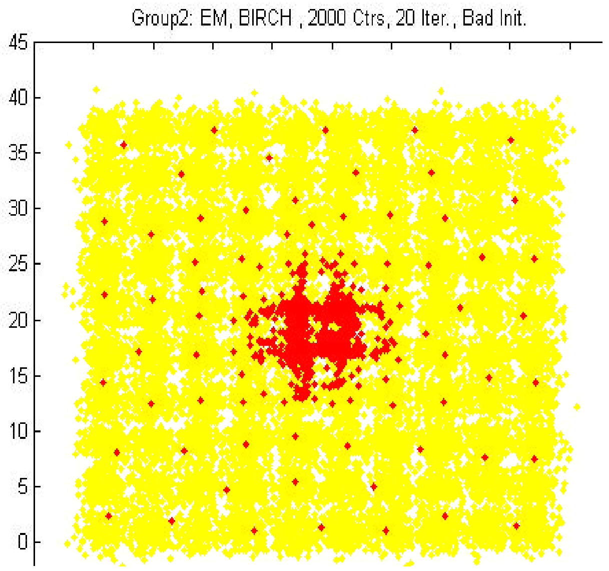


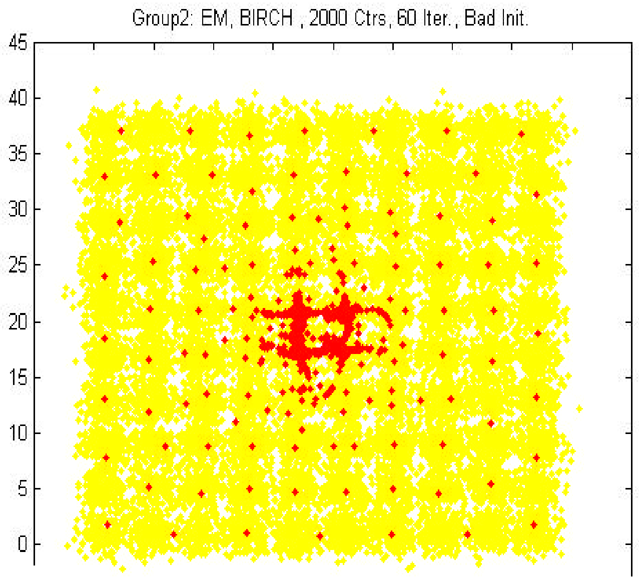


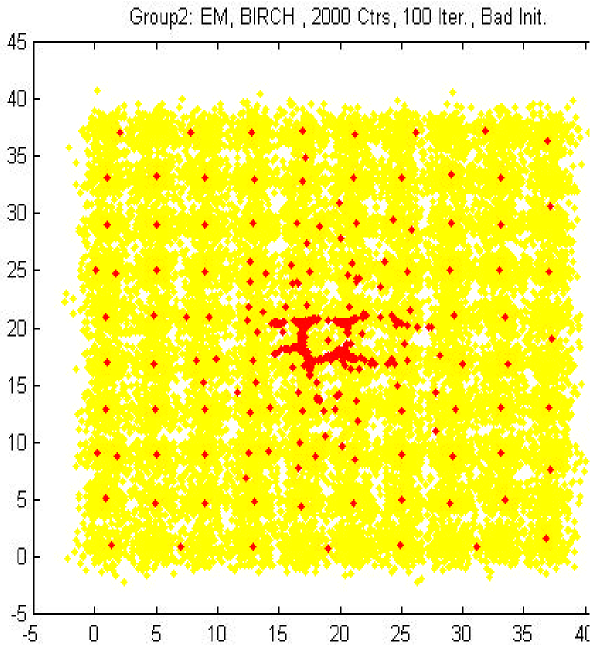


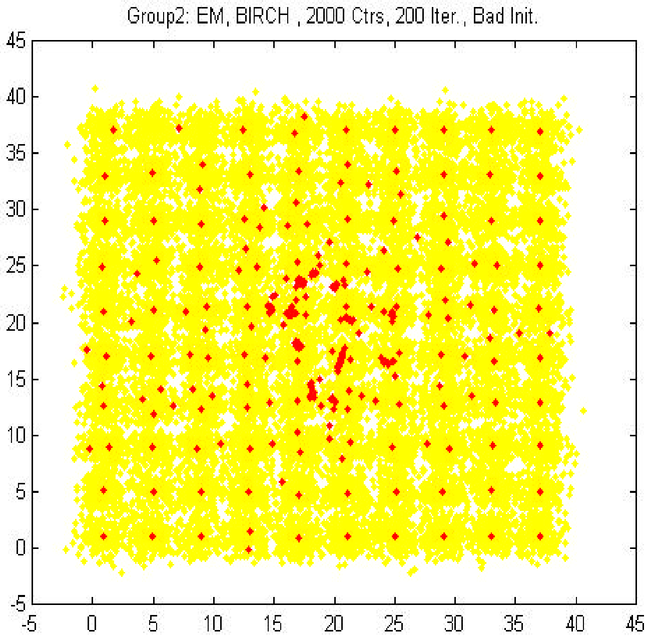


Group 2: #1 - EM after 20 iterations.
#2 - EM after 60 iterations.
#3 - EM after 100 iterations.
#4 - EM after 300 iterations.

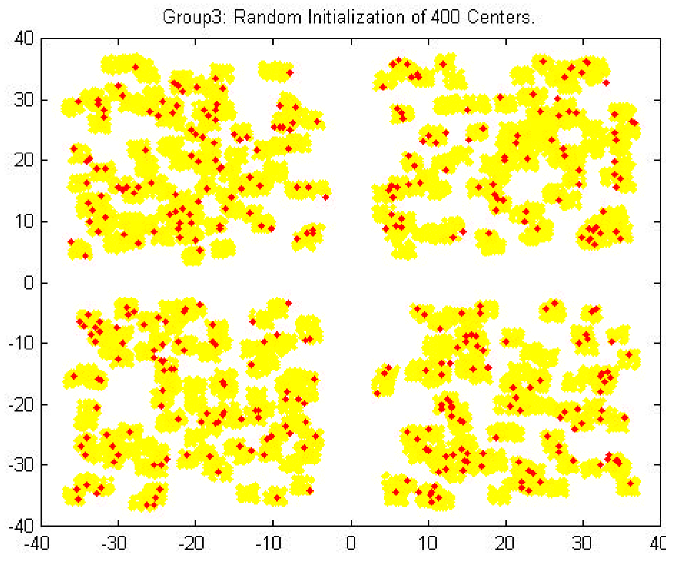


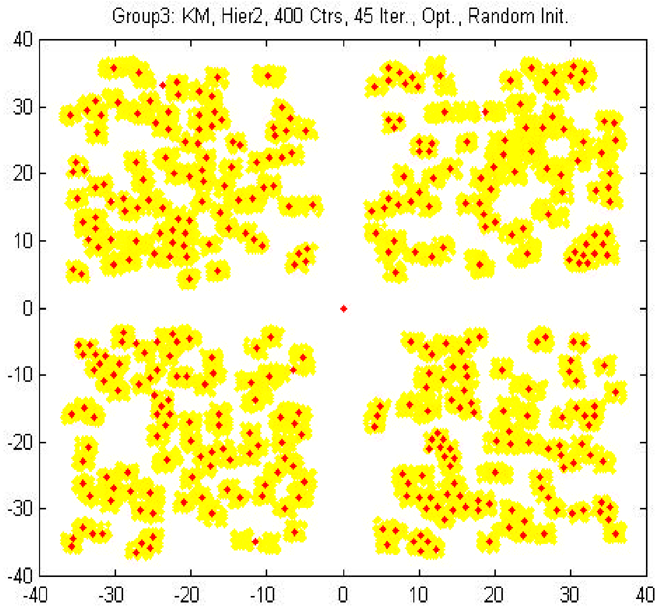


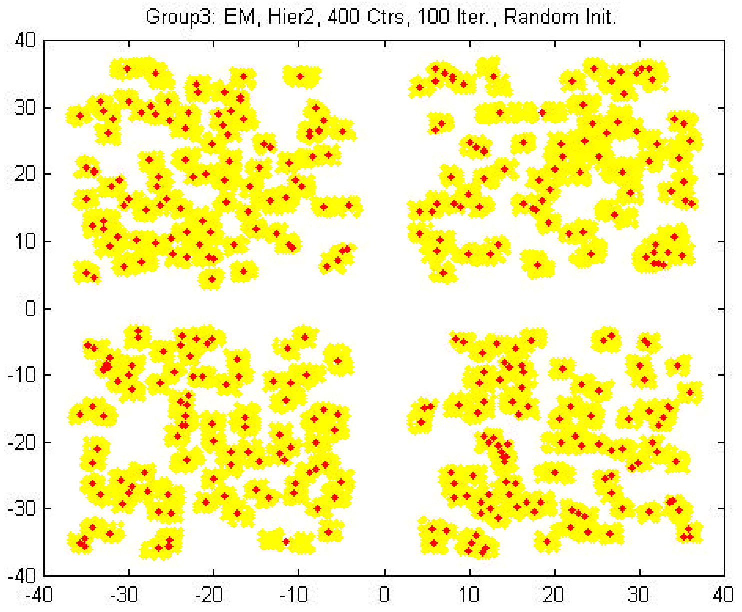




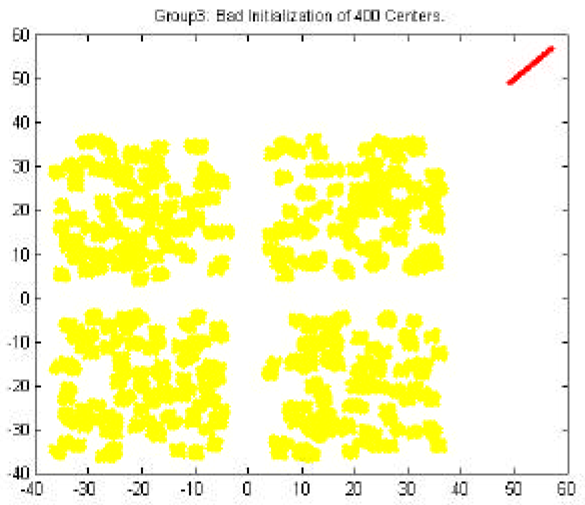
Group 3: #1 - Random initialization of 400 centers used for both KM and EM.
#2 - KM reached a local optimal after 45 iterations and stopped.
#3 - EM after 100 iterations.

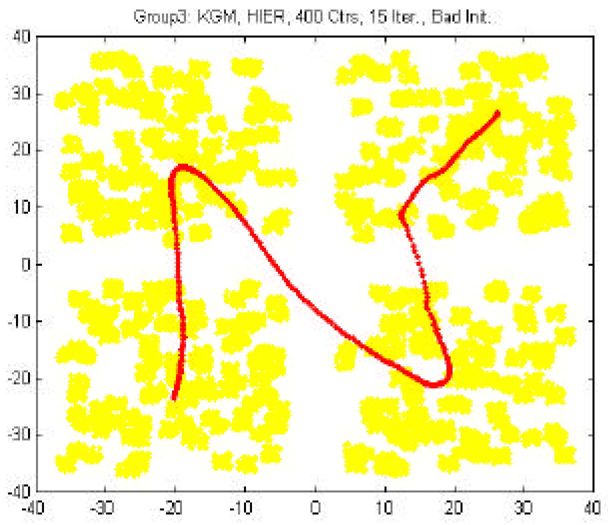


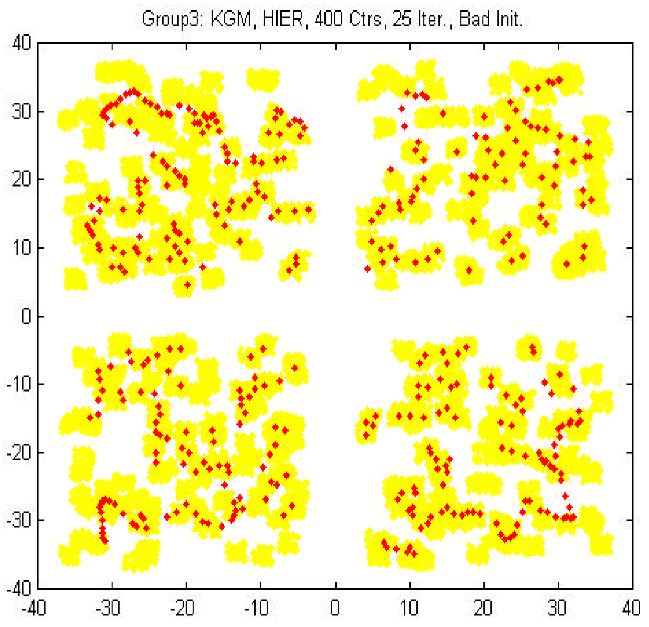


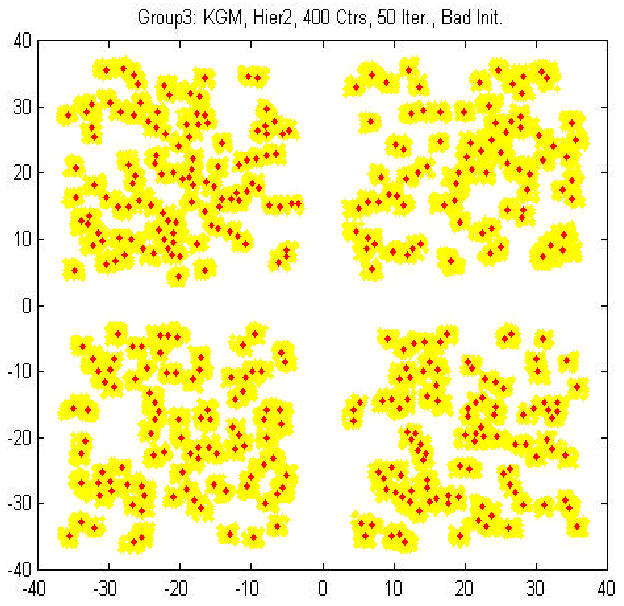


Group 3: #1 - The bad initialization used for KGM in this experiment.
#2 - KGM after 15 iterations.
#3 - KGM after 25 iterations.
#4 - KGM after 50 iterations.

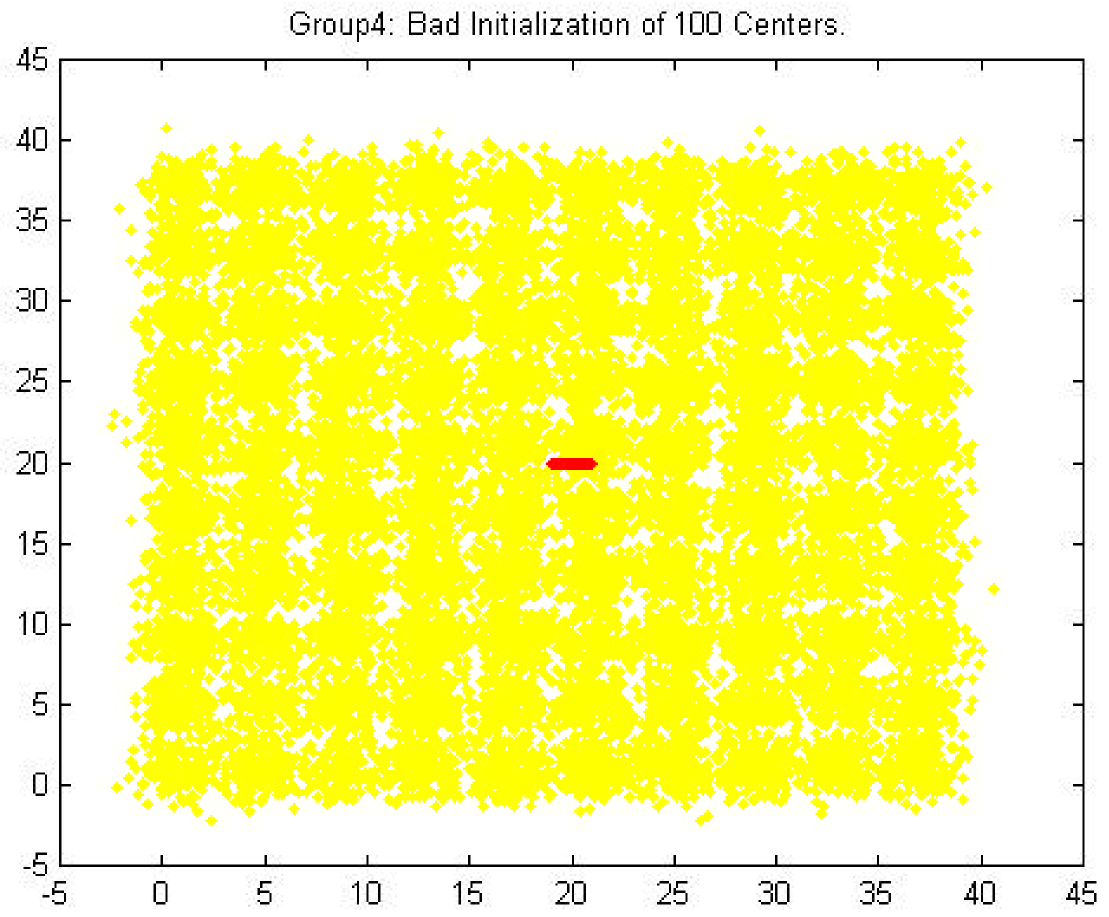




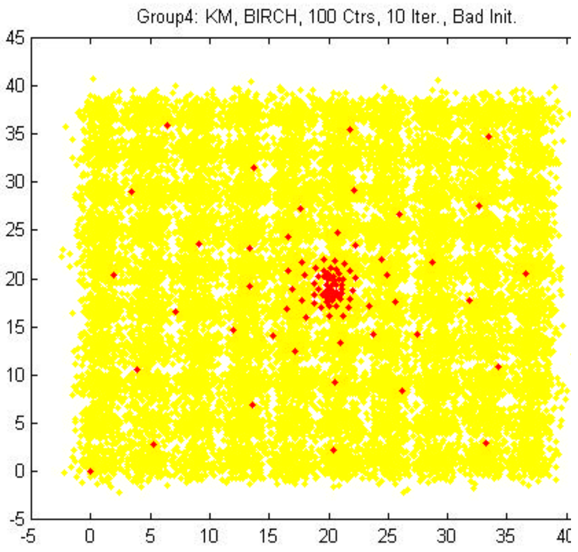


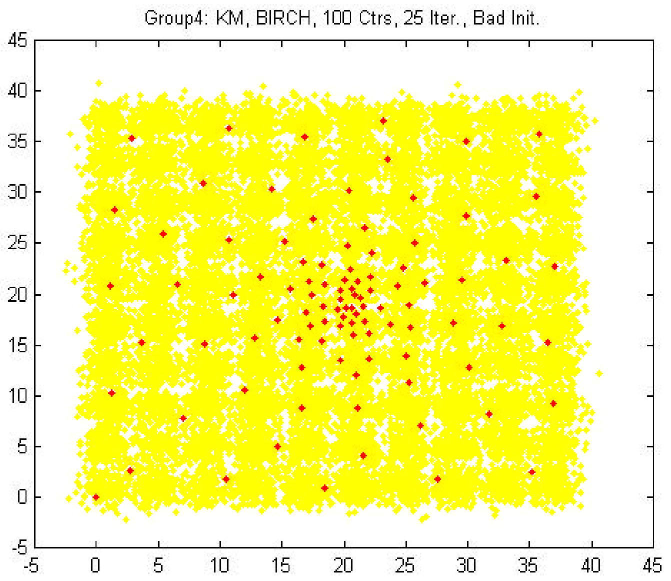


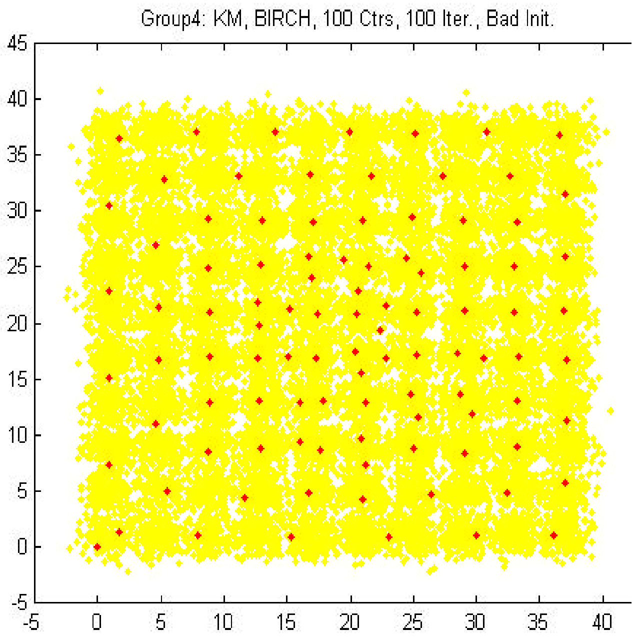
Group 4:	The bad initialization used for all three experiments in this group.

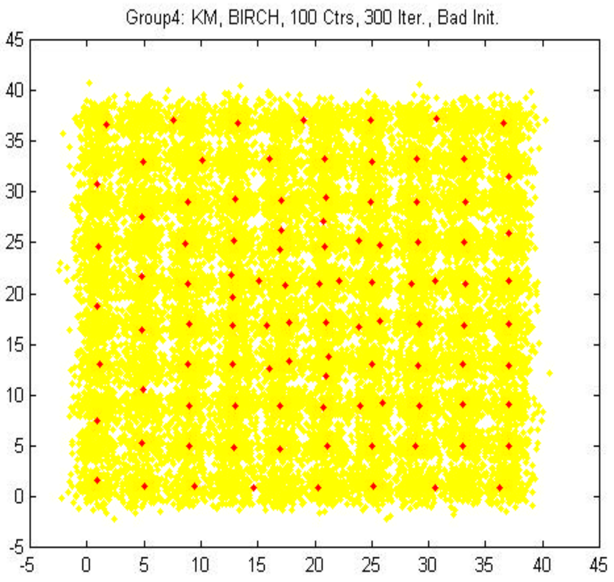


Group 4: #1 - KM after 10 iterations.
#2 - KM after 25 iterations.
#3 - KM after 100 iterations.
#4 - KM after 300 iterations.

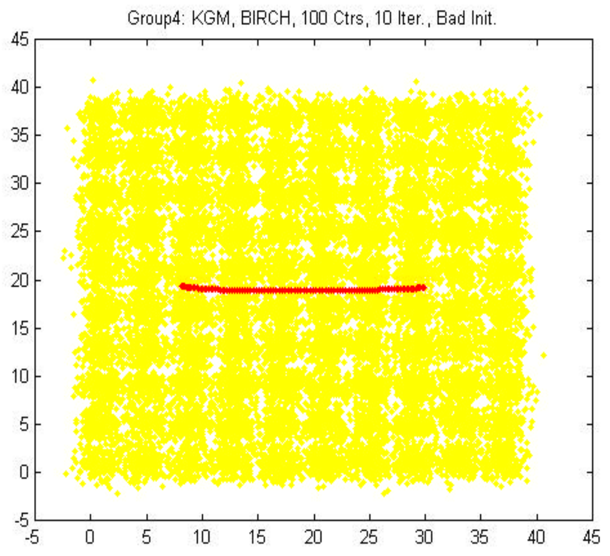


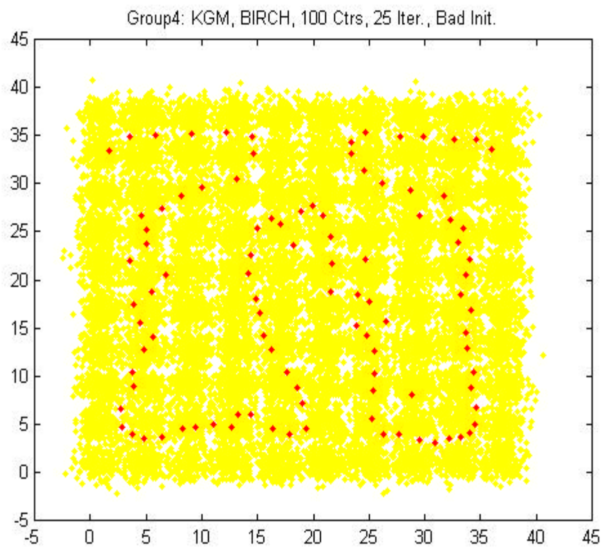


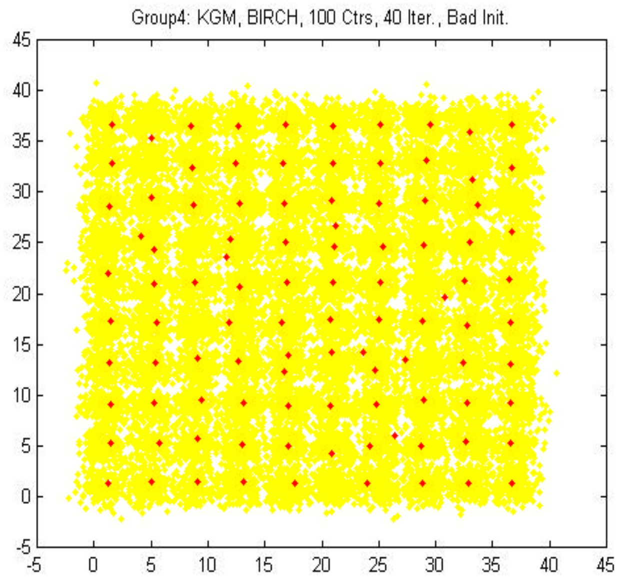


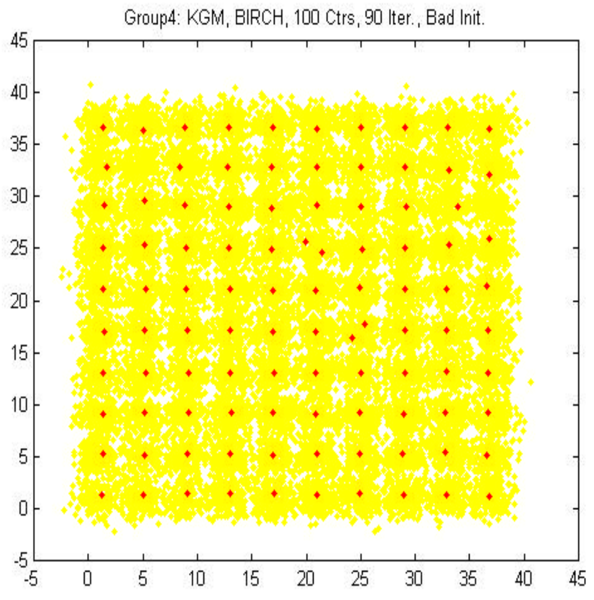


Group 4: #1 - KGM after 10 iterations.
#2 - KGM after 25 iterations.
#3 - KGM after 40 iterations.
#4 - KGM after 90 iterations.

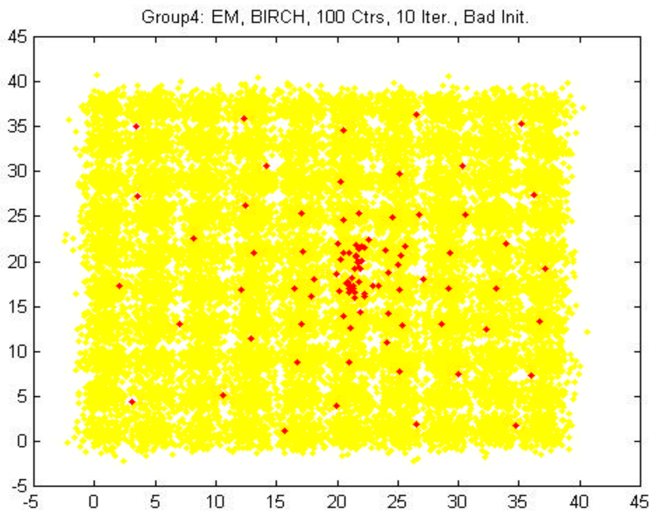


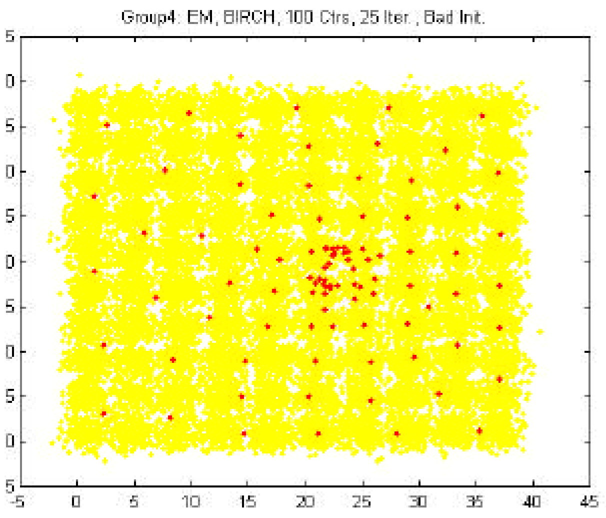


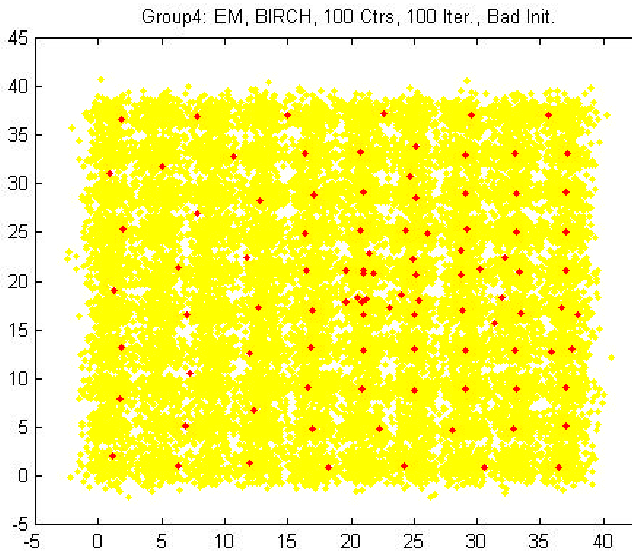


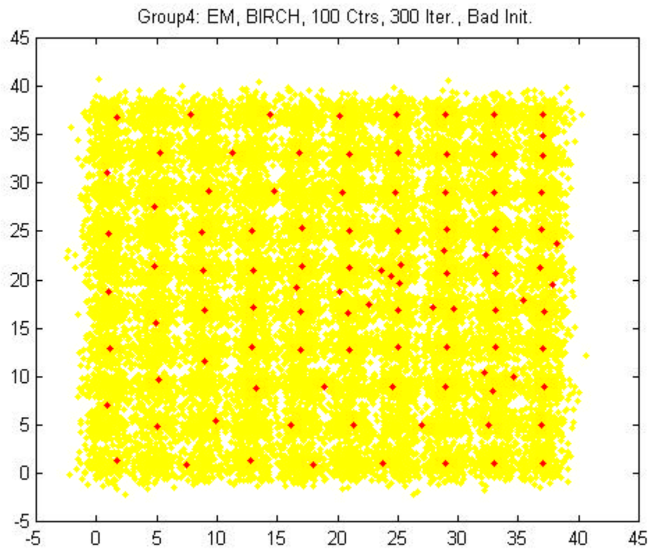


Group 4: #1 - EM after 10 iterations.
#2 - EM after 25 iterations.
#3 - EM after 100 iterations.
#4 - EM after 300 iterations.


Group4: EM, BIRCH, 100 Ctrs, 10 Iter., Bad Init.


Group4: EM, BIRCH, 100 Ctrs, 25 Iter., Bad Init.


Group4: EM, BIRCH, 100 Ctrs, 100 Iter., Bad Init.


Group4: EM, BIRCH, 100 Ctrs, 300 Iter., Bad Init.