# A Local Search Approach to K-Clustering

Bin Zhang, Gary Kleyner*, Meichun Hsu
Software Technology Laboratory
HP Laboratories Palo Alto
HPL-1999-119
October, 1999

local search algorithm, data clustering, K-means, clustering aggregated data, clustering large data sets, data compression, vector quantization

Data clustering is one of the common techniques used in data mining. A popular performance function for measuring goodness of the K-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. In this paper, we approach the min-MSE clustering problem by way of a Local Search (LS) algorithm, and analytically derive a clustering algorithm which we call LKM. A number of analyses of LKM are given; in particular, we prove that the set of local optima that can trap KM is a superset of those that can trap LKM. The experimental results also show that LKM converges faster and better than KM. More importantly, LKM naturally extends to an aggregated version, called A-LKM, which can be applied to the problem of clustering large data sets. A-LKM is a clustering algorithm which clusters subsets of data points, or subclusters, instead of individual data points. It can be used to cluster, for example, a large data set that has been aggregated through an algorithm such as the Phase 1 of the BIRCH algorithm ([ZRL96]), with the intention of fitting the aggregated data into the main memory to enable main memory-based clustering. We prove that A-LKM, as applied to the problem of clustering subclusters, preserve the monotone convergence property. Experimental results also show that A-LKM performs better than A-KM, in clustering aggregated data.

* ICC-Intranational Computer Consultants Inc.

## 1.0 INTRODUCTION

Data clustering is one of the common techniques used in data mining. An ideal case is to group related data into the same cluster and unrelated data into different clusters. Examples of applications of clustering include customer segmentation, document categorization, and scientific data analysis.

One popular class of data clustering algorithms is the center-based clustering algorithms. By finding $K$ centers positions (local high densities of data), $M=\{m_i \mid i=1,\ldots,K\}$, the data, $\boldsymbol{S}=\{x_i \mid i=1,\ldots,N\}$, can be put into clusters using the voronoi partition (i.e. every data item goes with the center that it is closest to). To find $K$ centers, the problem is defined as an optimization (minimization) of a performance function, $Perf(X, M)$, defined on both the data items and the center locations. A popular performance function for measuring goodness of the K-clustering is the total within-cluster variance, or the total mean-square quantization error (MSE). Formally, let

$R^D$ be the Euclidean space of dimension $D$;
$\boldsymbol{S} \subseteq R^D$ be a finite subset of $N$ data items;
$\boldsymbol{P} = (\boldsymbol{S_1},\ldots,\boldsymbol{S_K})$ – a partition of $\boldsymbol{S}$, where $\boldsymbol{S} = \bigcup_{i=1\ldots K} \boldsymbol{S_i}$ and $\boldsymbol{S_i} \cap \boldsymbol{S_j} = \varnothing$, the empty set;
$m_i = \{x \mid x \in \boldsymbol{S}_i\} / n_i$ and $n_i = |\boldsymbol{S_i}|$, the mean and size of the $i^{th}$ cluster.

The goodness of the partition is measured by the total within-cluster variance (or total mean-square quantization error):

$$MSE(P) = \sum_{i=1}^{K} \sum_{x \in S_i} (x - m_i)(x - m_i)^T$$

(We'll also use the shorter notation $|x\text{-}m_i|^2$ for $(x\text{-}m_i)\,(x\text{-}m_i)^T$).

The optimization problem is to find a clustering $\boldsymbol{P}$ that minimizes $MSE(\boldsymbol{P})$.

The K-Means (KM) algorithm is a popular algorithm which attempts to find a $K$-clustering which minimizes MSE. It is proved to converge to a local optimum [GG91]. In this paper, we approach the min-MSE clustering problem by way of a *Local Search* (LS) algorithm, and derive a data clustering algorithm which we call LKM (Local Search K-Means). LKM turns out to be strikingly similar to KM despite the fact that it is derived from an entirely different path. The major difference between LKM and KM is the criteria for moving a data item. A number of analyses of LKM are given; in particular, we prove that the set of local optimums that can trap KM is a superset of those that can trap LKM. The experimental results also show that LKM converges faster and better than KM.

One problem which some recent research has focused on is clustering large data sets ([ZRL96] [NH94] [BFR98]). In particular, the BIRCH algorithm, proposed in [ZRL96], uses aggregation of data as the mechanism for scaling up data clustering algorithms. The first phase of BIRCH aggregates the original data set into a number of locally dense subclusters such that the information about the subclusters can fit into main memory, so as to enable main memory-based clustering over the subclusters. BIRCH accomplishes its first phase by inserting data items into a tree based on a distance function and a bounding function so that the data items in each leaf nodes form the subclusters. By properly selecting the thresholds, the size of tree is controlled to fit into the main memory available. After building the tree (more precisely the aggregation of data), other data clustering algorithms, such as K-Means, can be applied to the aggregated data (i.e., the subclusters) to complete the clustering task.

LKM naturally extends to an aggregated version, which we call A-LKM. A-LKM is a clustering algorithm which clusters subsets of data points, instead of individual data points. It can be used to cluster a large data set that has been aggregated through an algorithm such as the BIRCH algorithm. The criterion for moving a subcluster of data from one cluster to another cluster is analytically derived in LKM, and every move will result in a decrease of the value of MSE, satisfying the monotone convergence property. In contrast, such property is not proven when applying KM directly to an aggregated data set, substituting every subcluster with a data point represented by the subcluster's centroid. We also experimentally demonstrate advantages of the A-LKM algorithm over A-KM (which is K-Means applied to aggregated data) for clustering aggregated data.

This paper is organized as follows. Section 2.0 develops LKM algorithm from Local Search optimization algorithm; Section 3.0 compares LKM with the K-Means; Section 4.0 develops the Aggregated LKM and presents experimental results of A-LKM and Section 5.0 concludes the paper.

## 2.0 THE LKM ALGORITHM

We derive both the LKM and A-LKM based on the Local Search optimization algorithm. We first introduce the Local Search Optimization Algorithm in the next subsection.

### 2.1 the Local Search Algorithm

Local Search (LS) algorithm is a general algorithm for finding a local optimum of a function ([GMW97]). For a performance function defined on a finite set, the optimum can be found theoretically by brute force search if it is affordable. The only reason for not doing so in practice is the cost. If the search is limited to a small region, which is defined by specifying a set of *neighborhoods* for each data item, the greedy algorithm (brute force search for the best in the neighborhood) will find a local best. By repeating this process, a local optimum of the function can be found.

Let $F: P \rightarrow R$ be a function defined on a finite set $P$. A neighborhood of a point $p \hat{I} P$ is a subset of $P$ that contains $p$. Each point in $P$ may have many neighborhoods. The sizes of the neighborhoods directly impact the cost of the brute force search and the quality of the local optimum. For each particular optimization problem, we are interested in only the kind of neighborhoods that help us to balance the cost of the brute force search, the speed of convergence and the goodness of the "local" optimum the algorithm converges to. The kind of neighborhoods we use in this paper will be clearly defined later.

To find a "local" optimum of a performance function, $F$, defined on $P$, we have

**Generic Version of Local Search Algorithm (G-LS)**

Step 1: Starting from an initial position, $current\_p = p_0 \hat{I} P$ and mark all neighborhood *unused*.

Step 2: Choose a unused neighborhood, $N(current\_p)$ of $current\_p$. If no more such neighborhoods left, STOP.

Step 3: Search in $N(current\_p)$ for $p_1 = argmin( F(p) \mid p \in N(current\_p))$; //i.e. $p_1$ gives the minimum of $F$ over $N(current\_p)$.

Step 4: If $F(p_1) < F(current\_p)$, set $current\_p = p_1$; EndIf.
Mark $N(current\_p)$ *used*. Goto Step 2.

The generic version of the algorithm is very simple and it will converge to a "local" optimum because the value of $F(current\_p)$ is monotone decreasing. The word "local" means that the converged point, $p_{final}$, gives the smallest value of $F$ among all the points in the *union* of all the neighborhoods of $p_{final}$,. How to choose the best set of neighborhoods to balance the three objectives -- low cost of the brute force search, high speed of convergence, and better "local" optimum – is still an art for most problems.

In Step 4 of **G-LS**, $F(p_1) < F(current\_p)$ gives the criterion for moving from $current\_p$ to $p_1$. If the search in the current neighborhood does not find a strictly lower value of $F$, the $current\_p$ will not be changed (no move), and another neighborhood of $current\_p$ will be tried (See Step 2). If after exhausting all the neighborhoods of $current\_p$ and still no lower value of $F$ is found, the algorithm will stop.

**2.2 Deriving the LKM Algorithm Using the Local Search Algorithm**

A tailored version of this simple algorithm is used to find a "local" optimum of the "total with-in cluster variance" performance function. We will prove that the local optimum found by this LS algorithm is a subset of the local optimums that the original K-Means algorithm can converge to. The following conclusion can be made:

The local optimums of the performance function found by the LKM, which includes all global optimums, is a subset of the local optimum that the original K-Means algorithm can converge to. The LKM is less likely to be trapped by the local optimums than K-Means.

First we will give a tailored version of the algorithm so that the notations are consistent with the performance function; then, we will define the kind of neighborhoods to be used.

The total with-in cluster variance is defined on the (voronoi) partitions of the data set, $\boldsymbol{S}$. Let $\boldsymbol{P}$ = { $P = (S_1, \ldots, S_K) \mid \boldsymbol{S} = \cup_{i=1\ldots K} S_i$ and $S_i \cap S_j = \varnothing$}, the set of partitions of $\boldsymbol{S}$. A "point" $P = (S_1, \ldots, S_K) \in \boldsymbol{P}$ is a partition of the data set $\boldsymbol{S}$. For each data item $s \in \boldsymbol{S}$, a neighborhood of $P$ is defined as

$N_s(P)$ = { All the partitions of $\boldsymbol{S}$ that can be derived by changing the membership of $s$}.

Or, in mathematical notation, if $s \in S_i$

$N_s(P) = \{P\} \cup \{ P' = (S'_1, \ldots, S'_K) \mid S'_i = S_i - \{s\}, S'_j = S_j \cup \{s\}, S'_k = S_k$ for $k \neq j$, $j=1,\ldots,K \}$.

All neighborhoods have the same size $K$, which determines the cost of the controlled brute force search. There are as many as $|\boldsymbol{S}|$ neighborhoods for each partition. After convergence, the "local" optimum under this neighborhood definition means that no single data item can be moved from one cluster to another to improve the value of the performance function. K-Means algorithm does not guarantee this. An example will be given later.

The tailored version of the criteria for making a move in Step 4 is given by the incremental change $\Delta_{P'} MSE(\boldsymbol{P}) = MSE(P') - MSE(P)$, where $MSE()$ is the total with-in cluster variance, $P = (S_1, \ldots, S_K)$ and $P' = (S'_1, \ldots, S'_K) \in N_{x0}(P)$. $x_0 \in \boldsymbol{S}$ is the data item being considered for a move.

$$\Delta_{P'} MSE(P) \equiv S_W(P') - S_W(P)$$

$$= \sum_{i=1}^{K} \sum_{x \in S'_i} |x - m'_i|^2 - \sum_{i=1}^{K} \sum_{x \in S_i} |x - m_i|^2$$

$$= \sum_{x \in S'_i} |x - m'_i|^2 + \sum_{x \in S'_j} |x - m'_j|^2 - \sum_{x \in S_i} |x - m_i|^2 + \sum_{x \in S_j} |x - m_j|^2$$

$$= [\sum_{x \in S_i} |x - m'_i|^2 - |x_0 - m'_i|^2] + [\sum_{x \in S_j} |x - m'_j|^2 + |x_0 - m'_j|^2] - \sum_{x \in S_i} |x - m_i|^2 + \sum_{x \in S_j} |x - m_j|^2$$

$$= \sum_{x \in S_i} |m_i - m'_i|^2 - |x_0 - m'_i|^2 + \sum_{x \in S_j} |m_j - m'_j|^2 + |x_0 - m'_j|^2$$

$$= \frac{-n_i}{n_i - 1} |x_0 - m_i|^2 + \frac{n_j}{n_j + 1} |x_0 - m_j|^2 .$$

The above derivation is briefly explained here. The 2nd line is derived by canceling out the terms over $S_k$'s which are the same for both $P'$ and $P$. The 3rd line is derived by rewriting the first two terms using $S_i$ and $S_j$ instead of $S'_i$ and $S'_j$. The 4th line follows from $|x - m'_i|^2 = |(x - m_i) + (m_i -$

$m'_i)|^2 = |x- m_i|^2 + 2*( x- m_i )(m_i - m'_i )^T + |m_i - m'_i|^2$. The first one cancels out with the third term and the middle one sum to zero. The same thing is done for the term indexed by $j$. The last line uses the following:

$$m'_i = \frac{n_i * m_i - x_0}{n_i - 1},$$
$$m'_j = \frac{n_j * m_j + x_0}{n_i + 1}.$$

Based on the above derivation, the criterion for moving $x_0$ is

$$\Delta_{P'} MSE(P) \equiv MSE(P') - MSE(P) < 0$$

or

$$\boxed{\frac{n_i}{n_i - 1} | x_0 - m_i |^2 > \frac{n_j}{n_j + 1} | x_0 - m_j |^2}$$

and the target cluster where $x_0$ is moved to is

$$j = \arg\min\{\frac{n_j}{n_j + 1} | x_0 - m_j |^2\}.$$

**Algorithm: LKM_One_Cycle:**

Step 1: Start with an initial partition $P = (S_1,\ldots, S_K)$; **Set** $a = 1$; Mark all neighborhoods (=data items) unused.

Step 2: **If** all data items in $S_a$ are used **then** $a$++; **endif**

**If** $a > K$ **then** STOP.

Get an unused data item, $x_0$, in $S_a$.

Step 3a: **Set** $A = n_a*(x_0-m_a)*(x_0-m_a)/(n_a-1);$

Step 3b: **for** ( $j = 1; j \neq a, j \leq K; j$ ++ ) **do**

$\delta_j MSE(P) \equiv MSE(P') - MSE(P) = n_j*(x_0 - m_j)*(x_0 - m_j)/(n_j+1) - A,$

where $P'$ is derived from $P$ by moving $x_0$ from $S_a$ to $S_j$;

Step 3c: Let $b$ = argmin{$\delta_j MSE(P) < 0$| all j != a}; // the best cluster for $x_0$ to move to.

Step 4: Mark $x_0$ as used.

**If** such $b$ exists **then**

Move $x_0$ from $S_a$ to $S_b$;

$--n_a;$ $++n_b;$

$m_a = (n_a * m_a - x_0)/(n_a - 1)$,
$m_b = (n_b * m_b - x_0)/(n_b - 1)$.
**Endif** //else do nothing.
goto Step 2

Since ***S*** is finite, the algorithm will stop after a finite number of steps.

**Algorithm: LKM**
Run **LKM_One_Cycle**
If Check(Stopping_Rule) = true **then** STOP
**End**

The Stopping_Rule can be either
1. The value of the performance function decreased less than ε in the previous run of **LKM_One_Cycle**; or
2. No data item is moved in the previous run of **LKM_One_Cycle**, which is more strict than Stopping_Rule #1.

We presented the LKM algorithm in multiple cycles primarily for comparing with the original K-means algorithm, as will be discussed in the next section. The cycles and order in which the data items in each cluster are processed is unimportant to the correctness of the algorithm. The only thing important is to check every data item to make sure that not a single data item can be moved before stopping the algorithm. A randomized version of LKM can be developed based on this observation. The data items can even be sampled randomly according to certain distribution, for example, a uniform distribution or a distribution based on importance sampling.

If we use Stopping_Rule #2, the converged partition has to be a "*local*" optimum, which means that no single data item can be moved to lower the value of the total within cluster variance. It is interesting to point out that the original K-means algorithm does not guarantee a local optimum under this neighborhood definition. An example is given later in Figure 1.

## 3.0 KM AND LKM: A COMPARISON

### 3.1 Review of the K-Means Algorithm

To prepare for the comparison of LKM with K-Means, we briefly review the K-means algorithm in this section. MacQueen [M67] is considered by many people to be the first one defined K-Means.

Starting with an initial set of points, $m_i$, $i=1, \ldots, K$, K-means algorithm has the following two phases:

1. For each data item, find the closest $m_i$ and assign the data item to the cluster $i$. The current $m_i$'s are not updated until the next phase. It is proved ([GG91]) that this phase gives the optimal partition for the given centers.
2. Recalculate, $m_i$, as the centroid of the $i^{th}$ cluster of the new partition. It is proved ([GG91]) that this phase gives the optimal center locations for the given partition of data.
3. Loop through 1 & 2 until the clusters no longer change.

After each phase, the total within cluster variance never increases (this is called the monotone convergence property of the algorithm) and the algorithm converges to a local optimum of the performance function. More precisely, the algorithm will reach a stable partition in finite number of steps because there is only a finite number of partitions of a finite data set. The cost per cycle (one run of phase 1 + one run of phase 2) is *O(K*N*D)*.

More results can be found in [GG91], [SI84] and the references there on K-Means type of algorithms with more general performance functions and their convergence properties.

### 3.2 Comparison of LKM and KM

*3.2.1 Computation costs:*

The computation cost of LKM and KM per cycle are both *O(K*N*D)*. The constant factor is also very close. Especially for high dimensional data, the difference in the constant factor is negligible. It is the convergence speed that determines the real cost in applications. (All experimental results we did show that LKM converges faster and to a lower total within cluster variance than KM, which will be discussed in the next subsection. For low dimensional data, if the number of iterations of KM and LKM are close, LKM could cost more due to a slightly bigger constant factor. For higher dimensional data, the time costs of KM and LKM are more proportional to the number of iterations they use.)

*3.2.2* Quality of the optimums:

We first show that all the local optimal partitions that trap LKM also trap KM. A data item can be moved by the LKM if $d_i \cdot MSE(\boldsymbol{P})$ is smaller than zero, which gives

**Condition A**: $(n_i /( n_i -1)) |x_0 - m_I|^2 > \text{MIN}( (n_j /( n_j + 1)) |x_0 - m_j|^2 | j=1,\ldots,K)$

KM moves a data item if the following is true:

**Condition B**: $|x_0 - m_i|^2 > \text{MIN}( |x_0 - m_j|^2 | j = 1,\ldots,K)$

The left side of Condition A is "boosted" and the right is "discounted" in comparison with those of Condition B. Therefore "Condition B is true" implies "Condition A is true", which implies

that a partition converged to by the LKM is stable under KM. If the LKM is trapped by (i.e. become stable under) a partition, KM will not be able to get out of it.

The example in Figure 1 shows that there does exist a partition that traps KM but not LKM. Considering a data set in the one-dimensional Euclidean space, *R*, let ***S*** = {0, 1.8, 3}, a set of three data points. Partition {{0, 1.8}, {3}} is stable under KM but it is not stable under LKM. LKM will converge to a better partition, {{0}, {1.8,3}}.

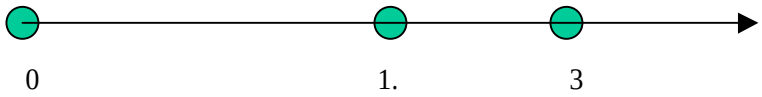


**Figure 1. A KM' stable partition can be improved by moving one data item. This will never happen to LKM.**

Combining the two arguments above, we have shown that all the local optimal partitions that trap LKM also trap KM. This conclusion is shown in Figure 2.

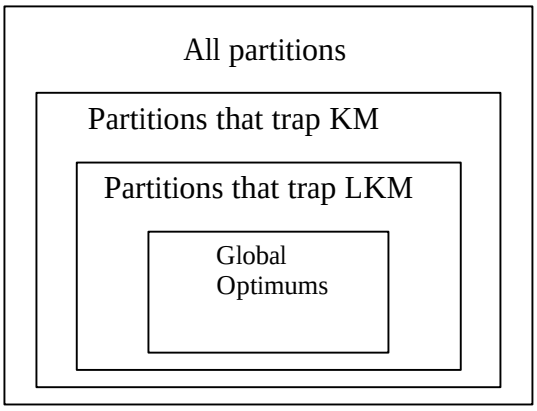


**Figure 2. The relationship between the stable partitions of the KM and that of LKM.**

Comparing how KM' **Condition B** and the LKM's **Condition A** will classify a new data item is also interesting. It reveals that there is a narrow region between two neighboring clusters and a data item falls in this regions can go to either cluster under KM.

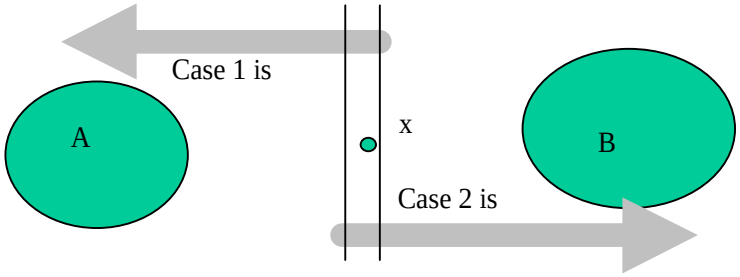


**Figure 3. The indecisive region of KM.**

Let *A* and *B* be two clusters with their centroids at $m_A$ and $m_B$, and with $n_A$ and $n_B$ data items in them. Let *A'* and *B'* be the new clusters by adding a new data item *x* into *A* or *B*, and $m'_A$ and $m'_B$, $n'_A$ and $n'_B$ defined accordingly. There are two cases:

1. $A' = A \dot{E} \{x\}$, $B' = B$ is stable under KM, i.e. $|x - m'_A|^2 < |x - m_B|^2$.
2. $A' = A$, $B' = B \dot{E} \{x\}$ is stable under KM, i.e. $|x - m_A|^2 > |x - m'_B|^2$.

If we ask for both 1. and 2. to be true, we will find a non-empty region between the two clusters where *x* can fall. The details can be carried out easily and the two hyperplanes given by the inequalities are drawn in Figure 3. Any data item that falls in the region between the two hyperplanes can join either A or B with the result of a stable partition under KM. We call this region indecisive region of KM. The indecisive region is given by the following formula:

$$((n_A + 1)/n_A)\,|x - m_B| > |x - m_A| > (n_B/(n_B+1))\,|x - m_B|$$

This indecisive region degenerates to a single hyperplane (a tie) for the LKM by design.

The analysis above can be extended to multiple clusters. Indecisive regions exist between any pair of neighboring clusters for KM. The larger the sizes of each cluster the narrower the indecisive region.

Due to the existence of indecisive regions, a single data item in this region separates (creates) two local optimums (two stable partitions) under KM. This gives an explanation why KM has more stable partitions than LKM.

*3.2.3 Convergence speed:*

Comparing with the KM, the LKM looks at the *after effect* of a move to decide if it should move (a data item). The centroids that are affected by moving a data item is updated right after the move. Therefore, each data item is processed based on the most updated information, from which we expect LKM to converge faster than KM. This expectation is supported by all experimental results we have. LKM merges the two separate phases in the original KM. As soon as a data item is moved from one cluster to another, the centroids, $m_i$ and $m_j$, of the changed clusters are updated.

*3.2.4 Empty Clusters vs Outlier Detection*:

Two more conclusions can be drawn from **Condition A** for the LKM:

1. No clusters will ever become empty under the LKM because the left of Condition A becomes zero when $\boldsymbol{S_i} = \{x_0\}$, a single data item.
2. If a single data item cluster, $\boldsymbol{S_j} = \{x_0\}$, remains that way for a full cycle, $x_0$ is an outlier in the following sense: **Condition A** is faults for all $x$, $|x - m_i|^2 \le ((n_i - 1)/n_i) * |x - x_0|^2/2 < |x - x_0|^2/2$. Every data item is at least twice of the distance away from $x_0$ than from its own cluster's centroid. $x_0$ is well isolated from all other clusters.

When this happens, we mark $x_0$ as an outlier and dropped the single data item cluster. Many ways of creating new clusters have been suggested for KM in history, which also apply to the LKM. We recommend splitting the cluster with the largest variance.

The comparisons are summarized in the following Table 1.

| | **KM** | | **LKM** |
|---|---|---|---|
| Stable Partitions | Stable partitions of KM | **É** | Stable partitions of LKM |
| Indecisive region. | Yes | | No |
| Empty cluster problem | Yes | | "No", it becomes a single point problem. Outlier detection. |
| Cost per cycle | O(N*K*D) | | O(N*K*D) |

**Table 1. Comparison of LKM with KM.**

### 3.3 Experimental Results on Comparison of LKM and KM

The LKM and KM algorithms have been implemented in C++. The experiments are run on HP UNIX C360. We have used three different sets of data:

1. UC Irvine data set,
2. BIRCH data set,
3. Hier2 – a synthetic data set.

The BIRCH data set has 100,000 points in 2 dimensions clustered around 100 (10x10) centers in a grid. Each cluster has 1000 points generated according to a normal distribution with radius sqrt(2). The distance between neighboring centers of the clusters are 4*sqrt(2). The UC Irving

data set has 20,000 points and 16 dimensions. Each point represents a rectangular pixel display of one of the 26 capital letters in the English alphabet. The Hier2 data set is generated by a hierarchical random data generator described in the next paragraph.

The design of the hierarchical random data generator is inspired by the type of clustering displayed in the Universe. Each galaxy remotely looks like a single dot in the Universe, but when you are inside of it, it has a structure with each individual "sun" like a dot; but when you are near each sun, it has satellites. The hierarchical random data generator starts with a few given points and converts each point into a local cluster of points. Then its output is used as its new input to generate more and tighter clusters. Hier2 started with four points, each one is turned into 100 points with uniform distribution and then each of the 400 points is turned into 50 points with uniform distribution.

The data sets and the number of clusters to be found are summarized in Table 2.

| Expr# | Data Set | Number of items | Number of dimensions | Number of Clusters |
|---|---|---|---|---|
| 1 | UC Irvine | 20,000 | 16 | 200 |
| 2 | BIRCH | 100,000 | 2 | 100 |
| 3 | BIRCH | 100,000 | 2 | 2000 |
| 4 | Hier2 | 20,000 | 2 | 400 |

*The shorter run amount the two.

**Table 2. The Setup of the Experiments.**

A common random partition is used to initialize both LKM and KM. After the initialization, the total with-in cluster variance is high for each data set. In all the experiments we ran, the convergence speed and the quality of the local minimum LKM converges to are consistently better than KM. We also timed all the experiments. The machine we used is a standalone HP UNIX C360 workstation. All experiments are run long enough for the convergence to reach a local optimum. In all cases, there is a significant convergence speedup being observed. (See Table 3 and Figure 5-8).

| Expr # | Number of Iterations | | Conver-gence Speed Up | Reached Local Optimal? | | Local Optimal Value | | Start Perfor mance Value | Total run time in seconds. | |
|---|---|---|---|---|---|---|---|---|---|---|
| | KM | LKM | | KM | LKM | KM | LKM | | KM | LKM |
| 1 | 6 | 4 | 33% | Yes | Yes | 2394 | 2385 | 5385 | 140 | 103 |
| 2 | 109 | 82 | 25% | Yes | Yes | 208602 | 202899 | 482046 | 319 | 341 |
| 3 | 74 | 30 | 59% | Yes | Yes | 11928 | 11678 | 24380 | 4178 | 2456 |
| 4 | 44 | 20 | 55% | Yes | Yes | 17979 | 17707 | 41265 | 101 | 66 |

**Table 3. The Summary of the Experimental Results.**

## 4.0 AGGREGATED LKM

### 4.1 Derivation of Aggregated LKM

Let $\boldsymbol{U} \subset \boldsymbol{S_i}$, $\boldsymbol{U} \neq \boldsymbol{S_i}$, be a true non-empty subset of the ith cluster. A new partition $\boldsymbol{P'}$ is derived from $\boldsymbol{P}$ by moving the subset $\boldsymbol{U}$ to the $j$th cluster of $\boldsymbol{P}$, i.e. $\boldsymbol{S'_i} = \boldsymbol{S_i} - \boldsymbol{U}$, $\boldsymbol{S'_j} = \boldsymbol{S_j} \cup \boldsymbol{U}$, and $\boldsymbol{S'_k} = \boldsymbol{S_k}$, for $k \neq i, j$. Let $n_u = |\boldsymbol{U}|$, $m_u = ( \sum_U x ) / n_u$, $s_u = \sum_U (x - u)^2$ be the size, mean and total squared error of the elements in $\boldsymbol{U}$. The means of the $i$th and the $j$th cluster after moving $\boldsymbol{U}$ from $\boldsymbol{S_i}$ to $\boldsymbol{S_j}$ (in $\boldsymbol{P'}$) are:

$$m'_i = ( n_i m_i - n_u m_u)/(n_i - n_u) \text{ and } m'_j = ( n_j m_j + n_u m_u)/(n_j + n_u).$$

The incremental change to the performance function, $\Delta P$, $MSE(\boldsymbol{P}) = MSE(\boldsymbol{P'}) - MSE(\boldsymbol{P})$, due to the move is

$$\begin{aligned}
&\Delta P, MSE(\boldsymbol{P}) = MSE(\boldsymbol{P'}) - MSE(\boldsymbol{P}) = \sum_{i=1..K} \sum_{x \in \boldsymbol{S'_i}} |x - m'_i|^2 - \sum_{i=1..K} \sum_{x \in \boldsymbol{S_i}} |x - m_i|^2 \\
&= \sum_{x \in \boldsymbol{S'_i}} |x - m'_i|^2 + \sum_{x \in \boldsymbol{S'_j}} |x - m'_j|^2 - \sum_{x \in \boldsymbol{S_i}} |x - m_i|^2 - \sum_{x \in \boldsymbol{S_j}} |x - m_j|^2 \\
&= [\sum_{x \in \boldsymbol{S_i}} |x - m'_i|^2 - \sum_{x \in \boldsymbol{U}} |x - m'_i|^2 - \sum_{x \in \boldsymbol{S_i}} |x - m_i|^2] + [\sum_{x \in \boldsymbol{S_j}} |x - m'_j|^2 + \sum_{x \in \boldsymbol{U}} |x - m'_j|^2 - \sum_{x \in \boldsymbol{S_j}} |x - m_j|^2] \\
&= [\sum_{x \in \boldsymbol{S_i}} |m_i - m'_i|^2 - \sum_{x \in \boldsymbol{U}} |x - m'_i|^2] + [\sum_{x \in \boldsymbol{S_j}} |m_j - m'_j|^2 + \sum_{x \in \boldsymbol{U}} |x - m'_j|^2] \\
&= [n_i |m_i - m'_i|^2 - \sum_{x \in \boldsymbol{U}} |x - m'_i|^2] + [n_j |m_j - m'_j|^2 + \sum_{x \in \boldsymbol{U}} |x - m'_j|^2] \\
&= [ - (n_i n_u / (n_i - n_u)) |m_u - m_I|^2 - s_u ] + [ (n_j n_u / (n_j + n_u)) |m_u - m_j|^2 + s_u] \\
&= n_u [(- n_i / (n_i - n_u)) |m_u - m_i|^2 + (n_j / (n_j + n_u)) |m_u - m_j|^2 ]
\end{aligned}$$

It is interesting to look at the terms separately: $(n_i n_u / (n_i - n_u)) |m_u - m_i|^2 - s_u$ is the amount by the variance of the $i$th cluster will decrease after taking out the subset $\boldsymbol{U}$. $(n_j n_u / (n_j + n_u)) |m_u - m_j|^2 + s_u$ is the amount by the variance of the $j$th cluster will increase after adding the subset $\boldsymbol{U}$ to it. $(n_i / (n_i - n_u)) |m_u - m_I|^2$, the first term in the brackets, measures the fitness of $\boldsymbol{U}$ in $\boldsymbol{S_i}$. $(n_j / (n_j + n_u)) |m_u - m_j|^2$, the second term in the brackets, measures the attractiveness of $\boldsymbol{S_j}$ to $\boldsymbol{U}$.

Moving subset $\boldsymbol{U}$ will result in a decrease in value of the performance function if $\Delta P$, $MSE(\boldsymbol{P}) < 0$, which is equivalent to

$$\text{Condition A': } (n_i / (n_i - n_u)) |m_u - m_I|^2 > \text{MIN}( (n_j / (n_j + n_u)) |m_u - m_j|^2 \mid j = 1, \ldots, K)$$

If $n_u = 1$, Condition A' is the same as Condition A. A subset behaves just like a BIG data item with the number of elements in it as the weight and the centroid as its location. The coefficients, $(n_i / (n_i - n_u))$ and $(n_j / (n_j + n_u))$ are more significant (boost or discount more) when $n_u > 1$.

The importance of working with subsets instead of individual data items is on clustering very large databases. Condition A' provides the theoretical criteria for an aggregated subset to be moved from one cluster to another.

When KM is applied to aggregated data, the criteria for moving a subset from one cluster to another is the same as KM applied to individual data items except that the centroid of the subset is used as the location of the subset. The subset is moved to the cluster if:

Condition B': $|m_u - m_i|^2 > \text{MIN}( |m_u - m_j|^2 \mid j=1,\ldots,K)$.

We refer to the aggregated version of the LKM algorithm as A-LKM, and that of KM as A-KM.

Comparing Condition A' and Condition B', the same statement we made for moving single data items is still true: A-LKM converges to a subset of the local optimums A-KM converges to. A-LKM is less likely to be trapped by the local optimums than A-KM, as shown in Figure 8. The proof is similar to that of the KM and LKM algorithm. To show that Partitions that trap A-KM and a superset of those trap A-LKM, we only need to compare Conditions A' and b'. To show that there exists at least one partition that traps A-KM but does not trap A-LKM, we change each data item in Figure 1 to two very close data items, we have an example to show that not every local optimum that trap A-KM will trap A-LKM: let $\boldsymbol{S}$ *={ -0.01, 0.01, 1.79, 1.81, 2.99, 3.01}, $U_1$={-0.01, 0.01}, $U_2$={1.79, 1.81}, and $U_1$={2.99, 3.01}.* Applying A-LKM and A-KM on *{$U_1$, $U_2$, $U_3$ }. {{$U_1$ ,$U_2$}, {$U_3$}}* traps A-KM but not A-LKM.

Due to aggregation of data points, the global optimums of the original performance function (defined on non-aggregated data partitions) may not be reachable after aggregation. But if some of them are reachable, they will be contained by the local optimums that trap A-LKM.

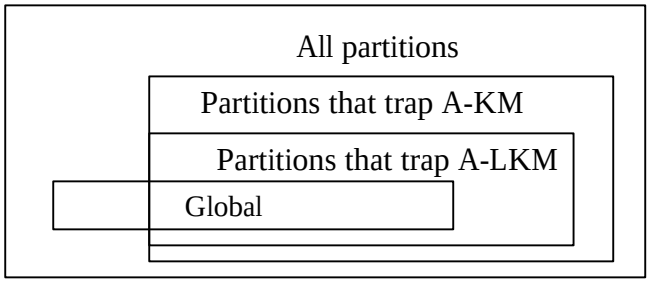


**Figure 4. The relationship between the stable partitions of the A-LKM and that of A-KM.**

When data is aggregated, like in A-KM, the performance function is still faithfully calculated as the same performance function before aggregation by keeping the sufficient statistics vector of each aggregated subset, which is (count, sum, sum of squares) (See [BRL96] and [BRF98]). The centroids of the clusters are still the true centroids of the non-aggregated data (also calculated from the sufficient statistics). Therefore the phase 2 of the convergence proof in [GG91] (p352, on Centroid Condition) is still true, the centers are still optimums for the given partition. But for the phase 1, the partition is no longer the optimal partition of the given centroids for the original (non-aggregated) data. The monotone convergence property of A-KM is, therefore, not proved. For A-LKM, the monotone convergence property still holds from our proof given at the beginning of this section.

### 4.2 Experimental Results on Aggregated Local Search Based KM (A-LKM)

We ran A-LKM and A-KM in two experiments. Both experiments are run long enough for the convergence to reach a local optimal. The first experiment takes the 2000 centers we got by running LKM on the BIRCH Data Set, we aggregated the clusters into 2000 aggregated subsets. A-KM and A-LKM are run on this 2000 aggregated subsets looking for 100 clusters. The convergence is given in Figure 9. The second experiment is performed on the 400 centers from the Hier2 data set looking for 40 clusters. The convergence of A-KM and A-LKM for this experiment is plotted in Figure 10. The convergence behavior is summarized in the table below. In both experiments, we see an improvement in convergence speed when LKM is used, one of which is very significant and the other less. We also see some incremental improvement in the quality of the optimum reached. (See Table 4.)

| Expr. | Number of Iterations | | Convergence Speed Up | Reached Local Optimal? | | Local Optimal Value | | Start Performance Value |
|---|---|---|---|---|---|---|---|---|
| | A-KM | A-LKM | | A-KM | A-LKM | A-KM | A-LKM | |
| BIRCH | 16 | 15 | 6% | Yes | Yes | 4628 | 4106 | 9048 |
| Hier2 | 13 | 6 | 54% | Yes | Yes | 7310 | 6893 | 14025 |

**Table 4. The Summary of the Experimental Results of A-KM and A-LKM.**

## 5. CONCLUSION

In this paper, we presented a K-clustering algorithm called LKM derived from the Local Search algorithm. LKM has several analytical advantages over the popular K-Means clustering algorithm (KM), and thus can be considered a refinement of the KM algorithm. LKM can offer a significant improvement in convergence speed, and some improvement in quality of the local optimum achieved. The aggregated version of LKM, which we call A-LKM, has a proven monotone convergence property; while the aggregated version of KM, which we call A-KM, does not share the original proof of monotone convergence property of KM. A-LKM, combined with an aggregation algorithm, such as the first phase of the BIRCH algorithm, is a better candidate for clustering very large data sets than such a combination with A-KM in both theoretical understanding and convergence quality and speed.

## References


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

[BB94] Leon Bottou and Yoshua Bengio, "Convergence Properties of K-Means Algorithms", Manuscipt at http://www.research.att.com/~leonb/biblio.html

[BFR98] Bradley, R.S., Fayyad, U. and Reina, C., "Scaling Clustering Algorithms to Large Databases.", Proc. The fourth International Conference on Knowledge Discovery & Data Mining, August 1998. Page 9.

[GG91] Gersho, A. and Gray, R, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers 1991.

[GMW97] Gill, P.E., Murray, W. and Wright, M.H., "Practical Optimization", Academic Press, 1997.

[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.

[NH94] Raymond T. Ng and Jiawei Han, "Efficient and Effective Clustering Methods for Spatial Data Mining", Proc., of VLDB, 1994.

[SI84] Shokri Z. Selim and M.A. Ismail, "K-Means Type of Algorithms: A Generalized Convergence Theorem and Characterization of Local Optimality", IEEE `Transactions on Pattern Analysis and Machine Intelligence, vol.PAMI-6, no.1, 1984.`

[ZRL96] Zhang, T., Ramakrishnan, R., and Livny, M., "BIRCH: an efficient data clustering method for very large databases", *ACM SIGMOD Record*, Vol. 25, No. 2 (June 1996), Pages 103-114 in: SIGMOD '96.

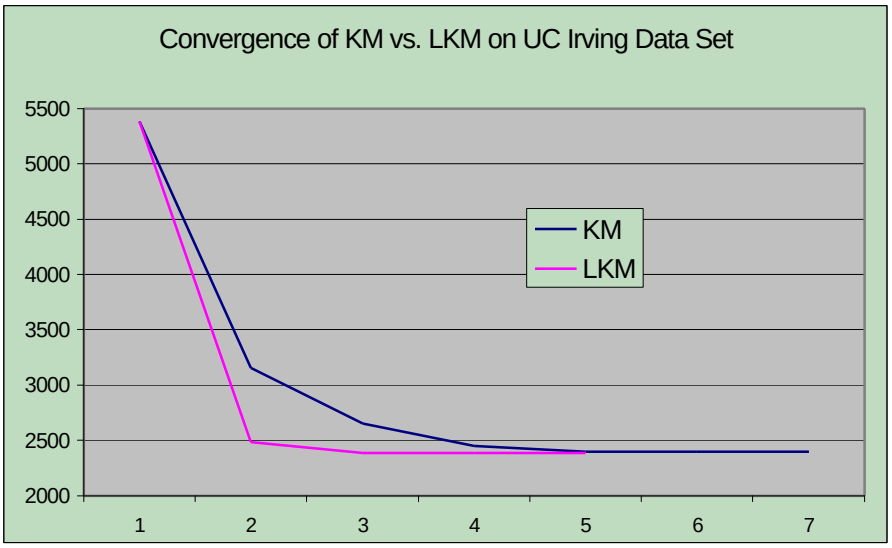


**Figure 5. Convergence of KM vs. LKM on UC Irving Data Set. 200 Centers.**

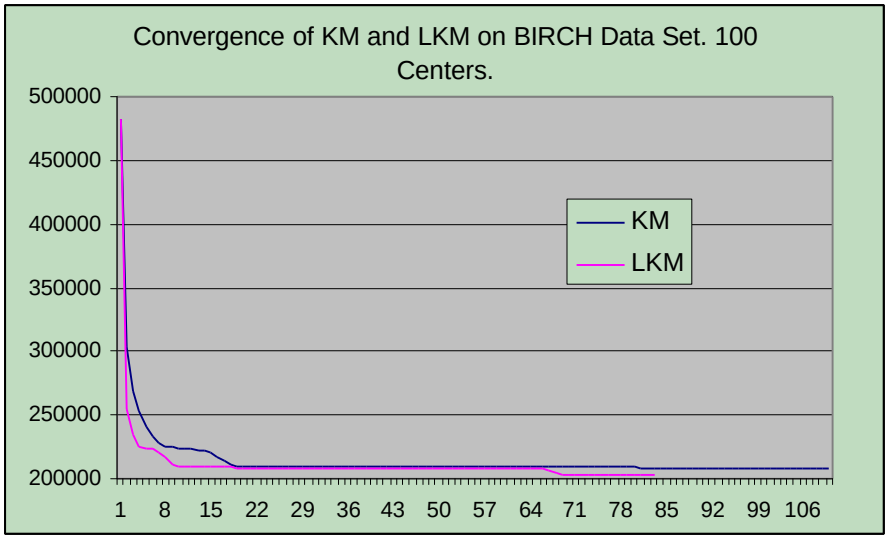


**Figure 6. Convergence of KM vs. LKM on BIRCH Data Set. 100 Centers.**

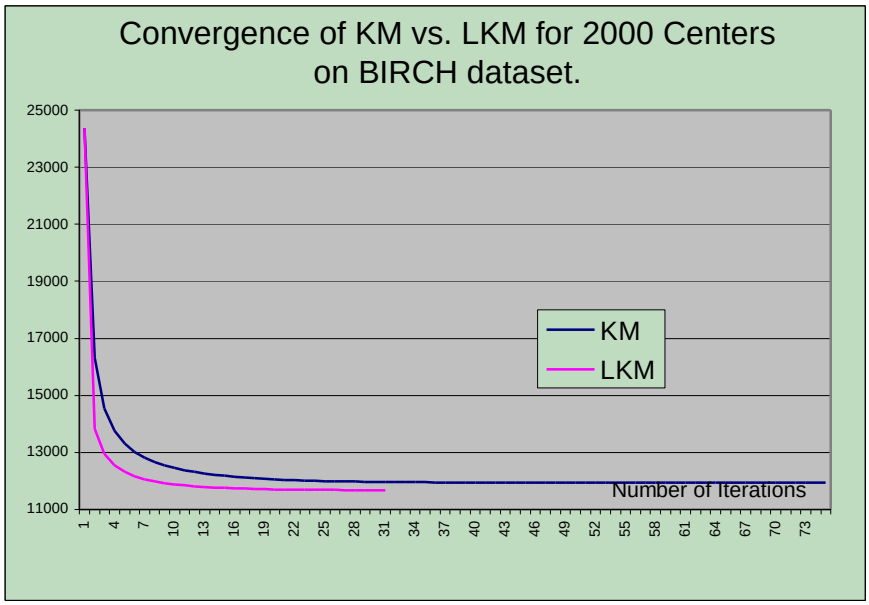


**Figure 7. Comparing the Convergence of KM and LKM on BIRCH Data Set with 2000 Centers**.

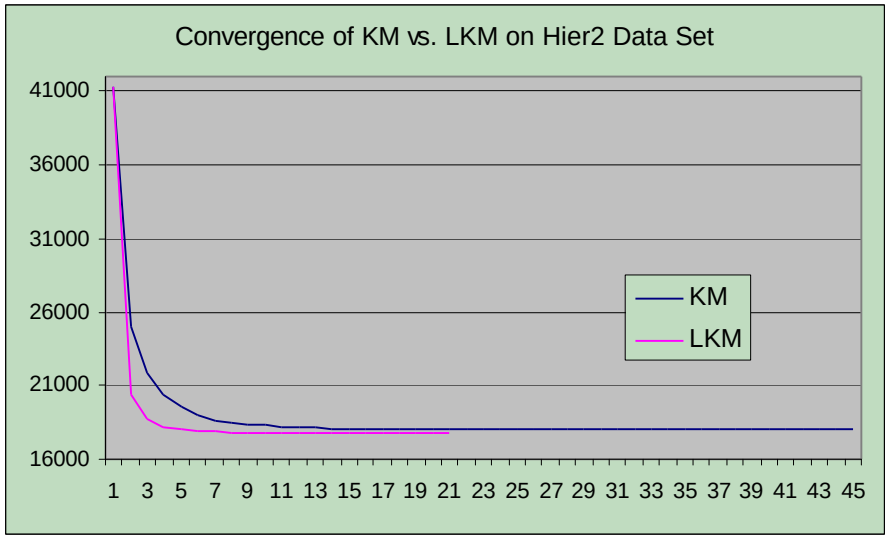


**Figure 8. Comparing the Convergence of KM and LKM on Hier2 Data Set with 400 Centers**.

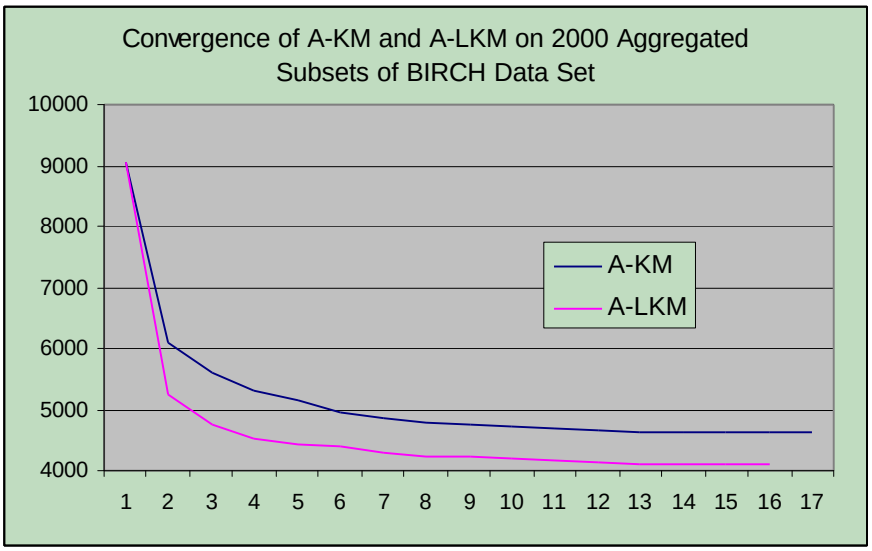


**Figure 9. Comparing the Convergence of A-KM and A-LKM on the 2000 Subsets (Aggregation) from Figure 7. 100 Clusters are Generated by Both A-KM and A-LKM.**

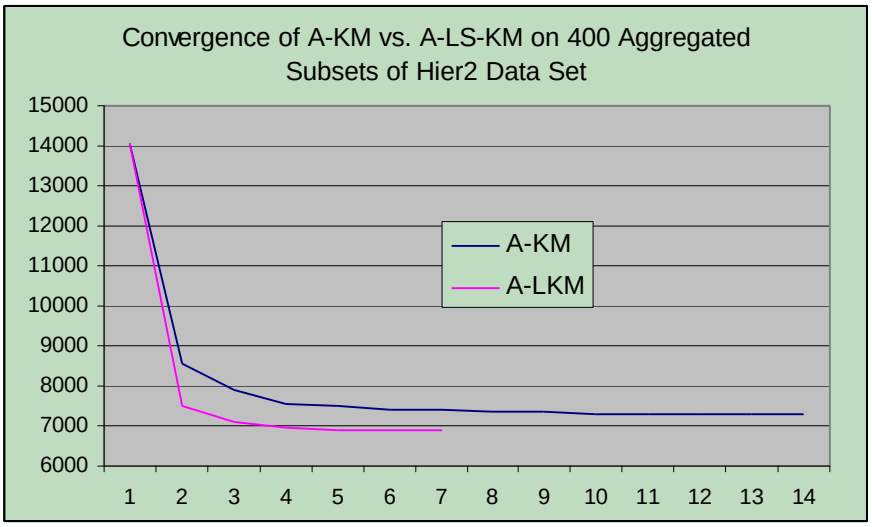


**Figure 10. Comparing the Convergence of A-KM and A-LKM on the 400 Subsets (Aggregation) from Figure 8. 40 Clusters are Generated by Both A-KM and A-LKM.**