3-2006

Inital Starting Point Analysis for K-Means Clustering: A Case Study

Amy Apon
Clemson University, aapon@clemson.edu

Frank Robinson
Vanderbilt University

Denny Brewer
Acxiom Corporation

Larry Dowdy
Vanderbilt University

Doug Hoffman
Acxiom Corporation

See next page for additional authors

Initial Starting Point Analysis for K-Means Clustering: A Case Study

Authors: Frank Robinson, Vanderbilt University
Amy Apon, University of Arkansas, Fayetteville
Denny Brewer, Acxiom Corporation
Larry Dowdy, Vanderbilt University
Doug Hoffman, Acxiom Corporation
Baochuan Lu, University of Arkansas, Fayetteville

Initial Starting Point Analysis for K-Means Clustering: A Case Study

Abstract— Workload characterization is an important part of systems performance modeling. Clustering is a method used to find classes of jobs within workloads. K-Means is one of the most popular clustering algorithms. Initial starting point values are needed as input parameters when performing k-means clustering. This paper shows that the results of the running the k-means algorithm on the same workload will vary depending on the values chosen as initial starting points. Fourteen methods of composing initial starting point values are compared in a case study. The results indicate that a synthetic method, *scrambled midpoints,* is an effective starting point method for k-means clustering.

1. Introduction

Most companies today rely on information technology (IT) to guide business decisions, direct major overseas transactions, and manage their day-to-day activities. These companies have Quality of Service (QoS) requirements that, if not met, negatively impact the business' profitability. In order to analyze IT systems and to ensure that QoS requirements are met, a good understanding of the current and/or projected workload is needed. This workload model should be representative of the actual workload and be able to imitate and accurately predict its impact on the performance of the specific IT system. Workload models, used in conjunction with other performance models, including simulation models and analytic queuing network models, can be used for capacity planning, forecasting future workloads, predicting system performance, and analyzing various "what if" performance scenarios [1].

Workload models are described by features that characterize the actual workload, and these features become parameters of performance models. Different systems are characterized by different features. Example features include the number of jobs, the job type, memory requirements, network bandwidth demands, processor needs, and software service requirements, among others. Choosing which features to include in a representative workload model depends on the nature of the specific system, what parameters are actually

available, and the purpose of the underlying performance study [2]. The features that are chosen make up the feature vector. After the features for the workload model are chosen, sample data is collected. The actual workload is then described by the set of points that include all of the selected features for each of the jobs in the system. That is, each job is described by its feature vector.

Because the number of jobs in a collected data set may be quite large, it is often necessary to perform clustering analysis on the sample data. Clustering analysis is a technique used to find representative classes, or homogenous groups, within the raw workload job data set. These classes of jobs determine the workload model.

There are several clustering techniques, including hierarchical clustering, minimum spanning tree, and k-means clustering [3]. The k-means clustering algorithm is used in the case study described in this paper. This is one of the simplest and most popular clustering algorithms. The algorithm finds *k* distinct classes of jobs, or clusters, when given a specific workload data set. The algorithm finds the midpoint, or centroid, of each cluster and assigns each job to its nearest centroid. The algorithm is initialized by providing: 1) the number of desired clusters, *k*, and 2) initial starting point estimates of the *k* centroids. There is no commonly accepted or standard "best" way to determine either the number of clusters or the initial starting point values. The resulting set of clusters, both their number and their centroids, depends on the specified choice of initial starting point values [4]. This case study shows that the number of clusters and the final cluster centroid values differ based on the initial starting values used when performing k-means clustering.

One common way of choosing the initial starting point values is to randomly choose *k* of the actual sample data points and use these as the initial starting values. This method, along with 13 other techniques, are proposed and analyzed in this study. For brevity, six of these techniques are presented and analyzed in this paper. These methods fall into two categories: actual sample starting points and synthetic starting points. In the case study reported here, the results indicate that the best synthetic method performs better than the best actual sample method.

The results of this case study indicate that the selection of initial starting point values impacts the result of the k-means clustering algorithm. The results also provide reasonable, rule-of-thumb heuristics for selecting initial starting point values for the clustering algorithm.

The remainder of this paper is organized as follows. Section 2 overviews the k-means clustering algorithm. Section 3 describes starting point selection. Section 4 presents the

details of the experimental case study. Finally, Section 5 summarizes the impact of the results.

2. Overview of the K-Means Clustering Algorithm

Given a data set of workload samples, a desired number of clusters, *k*, and a set of *k* initial starting points, the k-means clustering algorithm finds the desired number of distinct clusters and their centroids. A centroid is defined as the point whose coordinates are obtained by computing the average of each of the coordinates (i.e., feature values) of the points of the jobs assigned to the cluster [2]. Formally, the k-means clustering algorithm follows the following steps.

1. Choose a number of desired clusters, *k*.
2. Choose *k* starting points to be used as initial estimates of the cluster centroids. These are the initial starting values.
3. Examine each point (i.e., job) in the workload data set and assign it to the cluster whose centroid is nearest to it.
4. When each point is assigned to a cluster, recalculate the new *k* centroids.
5. Repeat steps 3 and 4 until no point changes its cluster assignment, or until a maximum number of passes through the data set is performed.

Before the clustering algorithm can be applied, actual data samples (i.e., jobs) are collected from observed workloads. The features that describe each data sample in the workload are required *a priori*. The values of these features make up a feature vector (F_{i1}, F_{i2}, … , F_{iM}), where F_{im} is the value of the m^{th} feature of the i^{th} job. Each job is described by its M features. For example, if job 1 requires 3MB of storage and 20 seconds of CPU time, then (F_{11}, F_{12}) = (3, 20). The feature vector can be thought of as a point in M-dimensional space. Like other clustering algorithms, k-means requires that a distance metric between points be defined [2]. This distance metric is used in step 3 of the algorithm given above. A common distance metric is the Euclidean distance. Given two sample points, p_i and p_j, each described by their feature vectors, p_i = (F_{i1}, F_{i2}, … , F_{iM}) and p_j = (F_{j1}, F_{j2}, … , F_{jM}), the distance, d_{ij}, between p_i and p_j is given by:

$$d_{ij} = \sqrt{\sum_{m=1}^{M} (F_{im} - F_{jm})^2} \qquad (1)$$

If the different features being used in the feature vector have different relative values and ranges, the distance computation may be distorted since features with large absolute values

tend to dominate the computation [2]. To mitigate this, it is common for the feature values to be first scaled in order to minimize distortion. There are several different methods that can be used to scale data. The method used in this paper is z-score scaling. Z-score scaling uses the number of standard deviations away from the mean that the data point resides [5]. The z-score equation is

$$F^{*}_{im} = \frac{F_{im} - \mu_m}{\sigma_m} \qquad (2)$$

where F_{im} is the value of the m^{th} feature of the i^{th} job (i.e., the data point), μ_m is the mean value of the m^{th} feature, and σ_m is the standard deviation of the m^{th} feature. Thus, before the algorithm is applied, the original data set is scaled, using the z-score scaling technique, where the feature mean is subtracted from the feature value and then divided by the standard deviation of that feature (i.e., F_{im} is replaced by its scaled value F^{*}_{im}). This technique has the effect of normalizing the workload features so that no single feature dominates in the clustering algorithm.

The number of clusters to be found, along with the initial starting point values are specified as input parameters to the clustering algorithm. Given the initial starting values, the distance from each (z-scored scaled) sample data point to each initial starting value is found using equation (1). Each data point is then placed in the cluster associated with the nearest starting point. New cluster centroids are calculated after all data points have been assigned to a cluster. Suppose that C_{im} represents the centroid of the m^{th} feature of the i^{th} cluster. Then,

$$C_{im} = \frac{\sum_{j=1}^{n_i} F^{*}_{i,jm}}{n_i} \qquad (3)$$

where $F^{*}_{i,jm}$ is the m^{th} (scaled) feature value of the j^{th} job assigned to the i^{th} cluster and where n_i is the number of data points in cluster i. The new centroid value is calculated for each feature in each cluster. These new cluster centroids are then treated as the new initial starting values and steps 3-4 of the algorithm are repeated. This continues until no data point changes clusters or until a maximum number of passes through the data set is performed.

Given a clustered workload, an error function can be defined that describes how much error (or difference) there is between all of the jobs in the workload and their respective cluster centroids. In this study, the error function used is the sum of the distances that each point is from its cluster's centroid. Assuming that there are k clusters, the error function (E_k) is defined as:

$$E_k = \sum_{i=1}^{k} \sum_{j=1}^{n_i} \sum_{m=1}^{M} (F^*_{i,jm} - C_{im})^2 \qquad (4)$$

where $(F^*_{i,jm}-C_{im})^2$ is the distance measure between a data point and the cluster centroid to which it is assigned.

Typically, to determine the most descriptive number of clusters in a particular workload, the k-means clustering algorithm is run on the sample data set for a range of possible clusters, say, from k =1 to 10. At the end of each run, the error value E_k is calculated using equation (4). These error values are then plotted on a graph against the possible number of clusters. As the number of clusters k increases, it is reasonable to expect that the value of Ek decreases, since with more possible cluster centroids to choose from, it is more likely that any given job will be assigned to a potentially closer cluster. The final number of clusters in the data set is chosen by examining and selecting the "knee" of the E_k curve. For example, consider the generic E_k curve shown in Figure 1. Here the "knee" occurs when k = 4. Significant reduction in E_k occurs when going from k=1 to k=4. However, reducing E_k further by increasing the number of clusters beyond 4 is minimal. To summarize, the algorithm is executed with different numbers of potential clusters, k. The E_k curve is constructed and the value of k at the curve's knee determines the number of final clusters found in the data set.

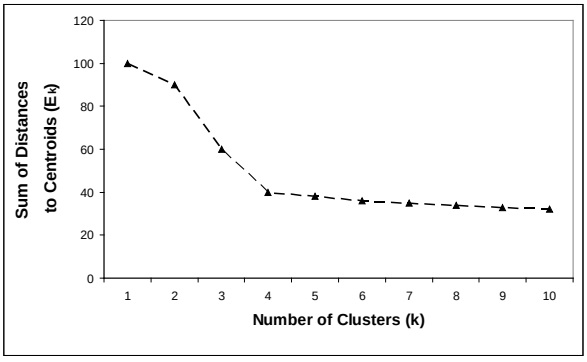

Figure 1. Generic E_k Curve

3. Overview of Starting Point Selection

In this section, two motivational examples illustrate the fact that choosing different starting point values lead to different clusters with different error values. First, consider a

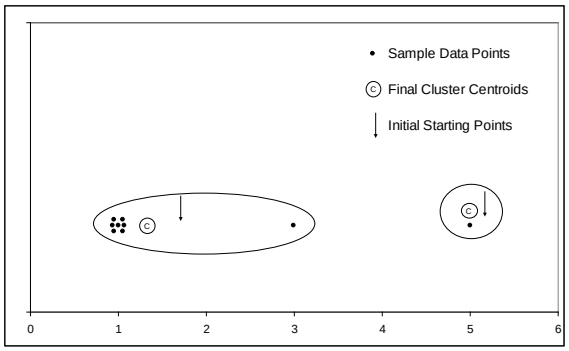

Figure 2. Starting Points Example 1a

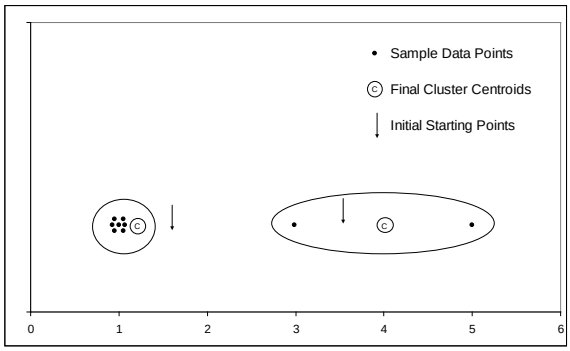

Figure 3. Starting Points Example 1b

workload with a single feature. Suppose that the feature values range from 1.0 to 5.0 and that there are seven samples at 1.0, one sample at 3.0, and one at 5.0. Figures 2 and 3 show the results of running the clustering algorithm on the data set when trying to find two clusters. Figure 2 shows the two clusters that are found when the initial starting points are 1.7 and 5.1. Eight data points are placed in the first cluster, with a centroid value of 1.25. The second cluster consists of one data point, with a centroid value of 5.0. The error function value is E_1=3.5. Figure 3 shows the two clusters that are found when the initial starting points are 1.5 and 3.5. With these starting points, one cluster consists of the seven data points with a centroid value of 1.0 and the other cluster consists of two data points with a centroid value of 4.0. This clustering leads to an error function value or E_2=2.0. This simple example illustrates that the selection of starting point values is important when trying to determine the best representation of a workload. In comparing figures 1 and 2 to determine which of the two clusterings is "better", a case can be made that a tight cluster is centered on 1.0, with the other two data points being outliers. This is captured in the second (i.e., Figure 3) clustering and also results in a lower E_2 value. Thus, lower values of E_k are taken as better clustering representations.

A second motivational example is illustrated in Figures 4-9, which show the clusters resulting from clustering on a workload with two features and 11 sample data points. When finding a single cluster, the initial starting point value does not matter because all data points will be assigned to the same cluster. Figure 4 shows the result of clustering when determining one cluster. The error function value is E_1=15.2. There are an infinite number of alternatives for initial starting values when trying to find more than one cluster. Figure 5 shows the best (i.e., lowest E_2 value) result for this example when determining two clusters. Nine data points are in one cluster and two data points in the other cluster. The resulting

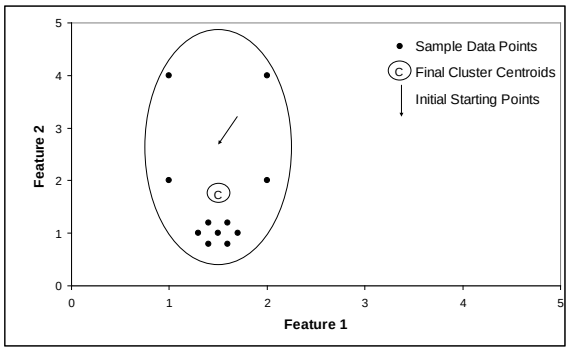

Figure 4. Starting Points Example 2: One Cluster

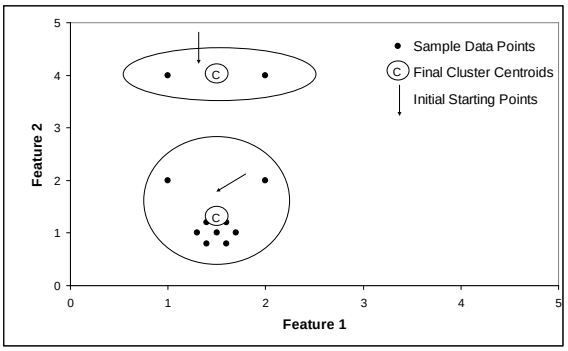

Figure 5. Starting Points Example 2: Two Clusters

error function value is E_2=2.6. When finding three clusters, two starting point alternatives are evaluated. The results from these two alternatives are shown in figures 6 and 7. In Figure 6, the starting points are: (1.8, 1.0), (1.0, 2.7), and (2.0, 2.7). The figure shows that the algorithm places seven data points in one cluster and two data points in each of the other clusters. The error function value is $E_{3,alt1}$=4.0. In Figure 7, the starting points are: (1.8, 1.0), (1.4, 2.0), and (1.4, 4.0). The algorithm again places seven data points in one cluster and two data points in each of the other clusters, but the error function value is $E_{3,alt2}$=1.0, indicating an improved clustering. Figure 8 shows the clustering result when finding four clusters, using (1.8, 1.0), (0.8, 2.0), (2.3, 2.0), and (1.3, 4.0) as the starting values. In this case, the error function value E_4=0.5.

Figure 9 shows the plot of the error function value E_k as a function of the number of clusters, *k*. As shown, the two starting point alternatives give different error function values. Intuitively, as more clusters are added, the error function value should decrease. Alternative 1 shows anomalous behavior, where $E_2 < E_{3,alt1}$, indicating that adding an extra cluster actually hurts the error measure. Identifying the knee of the curve for Alternative 1 in Figure 9, one could conclude that two clusters (i.e., Figure 5) best represents this data set. Better, more reasonable, methods for determining the initial starting points should not exhibit such anomalous behavior.

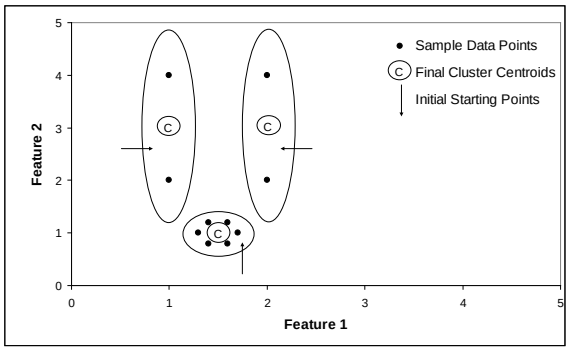

Figure 6. Starting Points Example 2: Three Clusters Alt 1

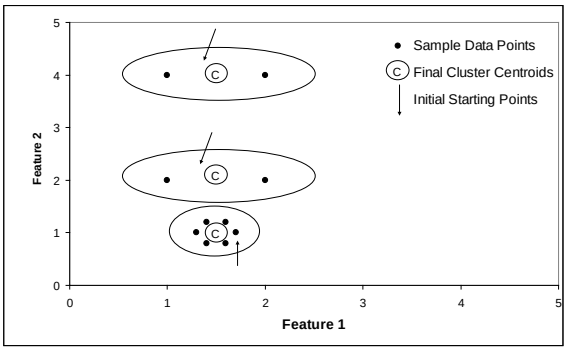

Figure 7. Starting Points Example 2: Three Clusters Alt 2

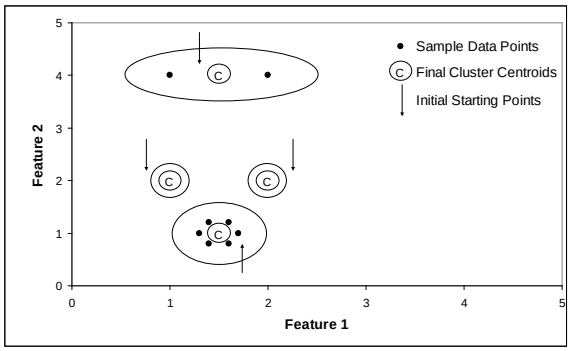

Figure 8. Starting Points Example 2: Four Clusters

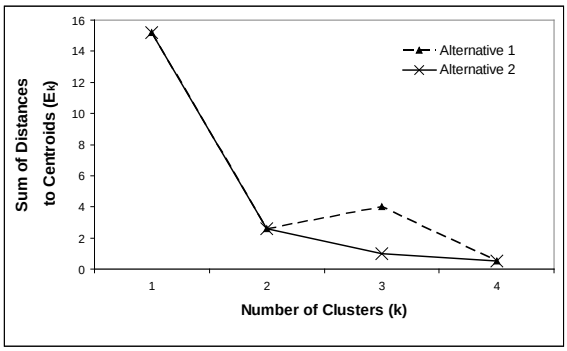

Figure 9. Starting Points Example 2: E_k Analysis

4. Experimental Case Study

For this case study, over 4200 data points, spanning the workload observed across several days, were collected and analyzed. The data collected was from a large data processing center from one of their grid computing environments. Five features were identified as being descriptive of each job (i.e., data point): number of records processed, number of processors used, response time, processor time consumed, and job resource type. Each of these features for each of the jobs was scaled using the z-score method.

A total of fourteen methods for choosing initial starting point values for the k-means clustering algorithm were evaluated. These methods fall into two main categories: actual sample starting points and synthetic starting points. Ten synthetic methods and four actual sample methods were compared.

Synthetic Initial Starting Points

A synthetic initial starting point is a point (i.e., a synthetic feature vector) that does not correlate to any specific observation in the actual workload. There are several different ways of coming up with these synthetic starting point values. A total of ten were identified and

evaluated. However, for brevity, only three synthetic methods are presented in detail: *scrambled midpoints, unscrambled midpoints,* and *scrambled medians*.

Several choices can be made when choosing synthetic initial starting values. This study considers: 1) dividing the range of feature values into different numbers of partitions, 2) picking initial starting points for each feature within these partitions, and 3) combining the different starting points of the features to create initial starting points. Before giving the detailed algorithms of the three synthetic starting point methods mentioned above, a discussion of each of these choices follows.

In nine of the ten synthetic methods studied, the range of feature values is divided into partitions. The number of partitions is k-1, k, or k+1, depending on the specific method. It was found that the number of partitions selected does not significantly affect the results of the k-means clustering algorithm.

After dividing the workload into partitions, values for each feature in each partition are chosen as initial starting values. Both the median and the midpoint values for each feature in the partitions are evaluated. For example consider a workload with a single feature. Suppose there are five sample data points in a particular partition: 1.0, 1.0, 1.0, 4.0, and 5.0. If using the median value, 1.0 would be chosen as the initial starting value for this partition. If using the midpoint value, 2.5 would be chosen as the initial starting value for this partition. Without presenting all the details, it was found that using the midpoint values is more effective in general than using median values when choosing synthetic initial starting points.

After coming up with the initial starting point values for each feature, a choice is made of how to combine the different feature values to construct an initial starting point value. Two methods are considered: scrambled and unscrambled. The unscrambled method selects partition #1 of each feature to construct starting point #1, selects partition #2 of each feature to construct starting point #2, … , and selects partition #k of each feature to construct starting point #k. For example, if a workload with three features is divided into two partitions (i.e., k=2), the values of the three features in the first partition forms one initial starting point, and the three feature values in the second partition forms the other initial starting point. The scrambled method combines the different partitions of the feature values randomly to create the initial starting points. For example, consider a workload with three features, (f_1, f_2, f_3), that is divided into two partitions. Since the features are combined randomly, one initial starting point might be the first partition's f_1 value, the second partition's f_2 value, and the second partition's f_3 value. The other start point might be the second partition's f_1 value, the

second partition's f_2 value, and the first partition's f_3 value. Again, without presenting all the details, it was found that using scrambled methods is more effective in general than using unscrambled methods.

One method for composing synthetic initial starting points is *scrambled midpoints*. The algorithm for scrambled midpoints follows:

1. Divide the range of (scaled) feature values into *k* equal sized partitions. For example, if a feature takes on values from –1.0 to 5.0 and k=3, then the three partitions are [-1.0, 1.0], [1.0, 3.0] and [3.0, 5.0]. This partitioning is done for each feature.
2. Take the midpoint of each partition for each feature as a potential initial starting feature value.
3. For each feature, randomly select one of the partition's midpoints to include in the starting point. Do this *k* times to construct the needed *k* starting points.

Since different combinations of the different feature values lead to a different error values, the k-means clustering algorithm is run multiple times for each *k*. The lowest error value for each *k* is used when constructing the E_k curve. For the results reported below, the algorithm was run 25 times for each *k*.

A second method of composing synthetic initial starting points is *unscrambled midpoints.* The unscrambled midpoints method is the same as the scrambled midpoints method with one exception. In step 3, features are combined based on the partition they are in. All of the feature values in the same partition are combined to create *k* sample data points. That is, the midpoints in partition #1 of each feature form starting point #1, … , and the midpoints in partition #k of each feature form starting point #k. There is no need to run the algorithm multiple times when using this method.

The third method of composing synthetic initial starting points is *scrambled medians*. The scrambled medians algorithm is the same as the scrambled midpoints algorithm with one exception. In step 2, the median value of each partition for each feature is used as a potential initial starting feature value. As in the scrambled midpoints method, the clustering algorithm is run multiple times for each *k* and the smallest error value for each *k* should be used when constructing the E_k curve. For the results reported below, the algorithm was run 25 times for each *k*.

Seven other synthetic methods were evaluated, but their results are not reported here. None of these seven offer any particular novel insight nor do they outperform the three considered here.

Figure 10 shows the E_k curve for the three synthetic methods. This figure indicates that the scrambled midpoints method outperforms the other two. It consistently gives low error values. In general, from Figure 10 and from other unreported results, scrambled methods outperform unscrambled methods and using midpoints outperforms using medians. From viewing the scrambled midpoints curve, the knee of the curve indicates 6 clusters are present in the workload data. This case study shows that the scrambled midpoints method is the preferred synthetic method.

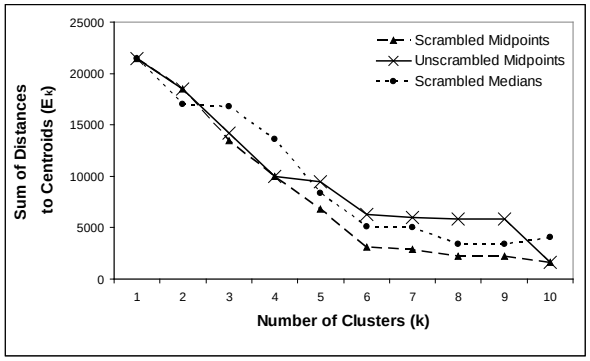

Figure 10. Synthetic Starting Point Analysis

Actual Sample Data Initial Starting Points

In contrast to constructing a synthetic starting point, an actual sample starting point is one that comes directly from the workload data. There are several different methods of selecting actual samples and using them as starting point values. A total of four were examined, three of which are reported here: *random*, *breakup*, and *feature value sums*.

The *random* method is a popular way of choosing the initial starting point values. It randomly chooses *k* of the sample data points and uses these as initial starting values. There are variations in the way that random sample data points are chosen. The algorithm for the random method follows:

1. Determine the maximum number of clusters, *n*, to be found by the k-means clustering algorithm. A value of *n*=10 is used here.
2. Choose *n* random sample data points.

3. For k=1 to k=n, run the k-means clustering algorithm, using k of the n sample data points chosen in step 2 as initial starting values. Each time k is increased, another one of the n random sample starting points is added as an initial starting point.

To prevent the anomaly described in Section 3, the set of data points that are used when running the algorithm to find k clusters is a subset of the data points that are used when running the algorithm to find k+1 clusters. For example, if k=2, using the same sample data point that that is used when k=1 (and simply adding another one of the pre-selected random sample data points as the other initial starting value) prevents anomalous behavior. Also, since combinations of the different jobs in the workload lead to a different error values, the k-means clustering algorithm is run multiple times, selecting different random jobs as starting points. The lowest error values are used in the E_k curve. For the results reported below, the clustering algorithm was run 25 times.

The second method for selecting actual sample data points as starting points is *breakup*. The algorithm for breakup follows:

1. Determine the maximum number of clusters, n, to be found by the k-means clustering algorithm.
2. Run the k-means clustering algorithm for k=1. (When finding a single cluster the initial starting point does not matter.)
3. For k=*2* to k=n
    a. From running the algorithm for k-1, determine the most populous cluster.
    b. Within this most populous cluster, find the two data samples that are the farthest away from each other.
    c. Using these two points, together with the k-2 other data points that formed the clusters that were not most populous, the set of k starting points are determined.

This method is reasonable since it seeks to break up the most populous cluster into two smaller clusters. It is hoped that a decreased error value E_k results.

The third method for selecting actual sample data points is *feature value sums*. The algorithm for feature value sums follows:

1. Each data sample's feature vector is added together to give a single value for each job.
2. The entire workload is sorted based on these feature vector sums.
3. For each k, the workload is divided into k equal sized regions.

4. The set of *k* median sample data points, one from each region, is used as the *k* initial starting points.

Figure 11 shows the E_k curve for the three actual data sample methods. This figure indicates that the breakup method generally outperforms the other two. It consistently gives low error values. From viewing Figure 11, it again appears that 6 clusters are present in the workload data. Thus, the synthetic methods and the actual sample methods reinforce each other. This case study indicates that the breakup method is the preferred actual sample method.

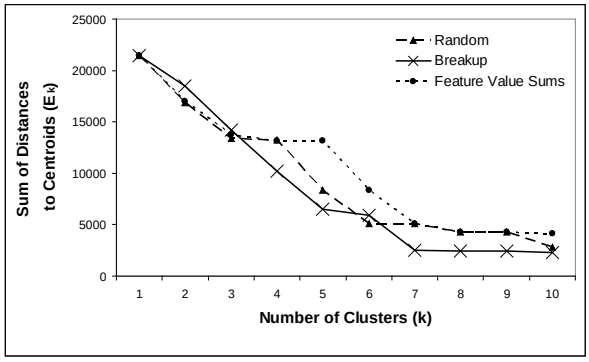

Figure 11. Actual Sample Starting Point Analysis

Figure 12 presents a summary comparison between the preferred actual sample method (breakup), the preferred synthetic method (scrambled midpoints), and the "best" overall error values found at each *k* when all fourteen different methods are considered. For example, scrambled medians could have found the lowest error value for k=3 clusters, breakup could have found the lowest error value for k=4 clusters, and random could have found the lowest error value for k=5 clusters. Based on Figure 12, scrambled midpoints (marginally) outperforms breakup and never seems to deviate far from the best curve. Therefore, the overall conclusion of this case study is that scrambled midpoints method is the preferred method to find initial starting point values.

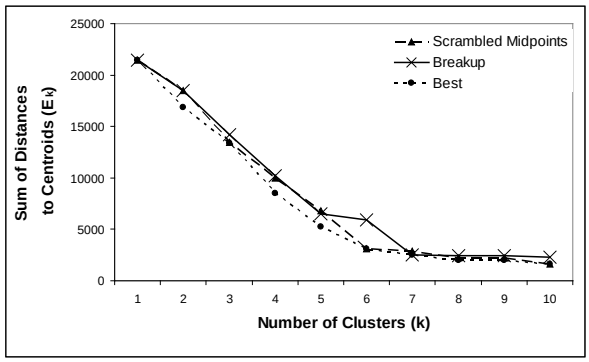

Figure 12. Synthetic/Actual Sample Comparison

5. Summary

This paper presents an overview of the k-means clustering algorithm. K-means clustering is a common way to define classes of jobs within a workload. Simple examples show that the initial starting point selection may have a significant effect on the results of the algorithm, both in the number of clusters found and their centroids. A case study is described where different methods of choosing initial starting points are evaluated.

A total of fourteen methods for choosing initial starting points are investigated for the case study. These methods fall into two categories: synthetic starting points and actual sample starting points. Six of these methods, three from each category, are presented. The conclusion is that the best synthetic method, scrambled midpoints, performs better than the best actual sample method, breakup. Thus, for the case study presented in this paper, scrambled midpoints is the preferred method for choosing initial starting values for the k-means clustering algorithm.

References

[1] Pentakalos, O., Menasce, D., and Yesha, Y., "Automated clustering-based workload characterization," *Joint Sixth NASA Goddard Space Flight Center Conference on Mass Storage Systems and Technologies and Fifth IEEE Symposium on Mass Storage Systems*, College Park, MD, March 23-26, 1998.

[2] D. Menasce, V. Almeida, and L. Dowdy. "Performance by Design," Prentice Hall, 2004.

[3] Jain, A., Murty, M., Flynn, P., "Data Clustering: A Review," http://scgwiki.iam.unibe.ch:8080/SCG/uploads/596/p264-jain.pdf.

[4] Eisen, Michael. "Cluster 3.0 Manual." http://bonsai.ims.u-tokyo.ac.jp/~mdehoon/software/cluster/cluster3.pdf.

[5] Hale. "Introductory Statistics." http://espse.ed.psu.edu/statistics/Investigating.htm.